

**METHODS FOR PARAMETERIZING AND EXPLORING PARETO
FRONTIERS USING BARYCENTRIC COORDINATES**

A Thesis
Presented to
The Academic Faculty

by

Matthew J. Daskilewicz

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology
May 2013

Copyright © 2013 by Matthew J. Daskilewicz

METHODS FOR PARAMETERIZING AND EXPLORING PARETO FRONTIERS USING BARYCENTRIC COORDINATES

Approved by:

Prof. Brian J. German, Co-Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Dimitri N. Mavris, Co-Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Christiaan J. J. Paredis
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Sumeet Parashar
Manager of Technical Services
ESTECO North America

Prof. Marcus J. Holzinger
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: 3 April 2013

To my mom and dad, who taught me that I can do anything.

ACKNOWLEDGEMENTS

This has been a long journey, and I am grateful for the many people who supported me along the way. I am especially thankful for my two wonderful advisors, Dr. Brian German, my great friend and mentor, and Dr. Dimitri Mavris, who believed in me from the very beginning. I am grateful to Dr. Chris Paredis, Dr. Sumeet Parashar, and Dr. Marcus Holzinger for their feedback and guidance. And I owe a special thanks to Dr. Rob McDonald and Dr. Tim Takahashi for their advice, inspiration, ideas, and support.

I would also like to thank the members of the “German Research Group,” past and present, with whom I have had the great pleasure of working, especially David Pate for driving me crazy and Collin Heller for keeping me sane. I thank William “Spam” Engler, who has commiserated with me on this journey and who has worked tirelessly to make our lab a better place. Finally I would like to thank the many friends who made my grad school experience fun, rewarding, and probably longer than it should have been, including but certainly not limited to James Arruda, Dane Freeman, Curtis Iwata, Chung Lee, Jonathan Murphy, Olivia Pinon Fischer, and Elizabeth Tang.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF SYMBOLS	xv
SUMMARY	xix
I INTRODUCTION AND MOTIVATION	1
1.1 Multi-Attribute Design Problem Terminology and Notation	5
1.1.1 Pareto Optimality	6
1.1.2 Definition of Subfrontiers	7
1.1.3 Pareto Frontier Regularity Condition	8
1.2 Approach to Multi-Attribute Design Problems and Need for a Coordinate System on the Pareto Frontier	9
1.3 Simplicies and Barycentric Coordinates	15
1.4 Outline	16
II LITERATURE SURVEY OF MULTI-OBJECTIVE OPTIMIZATION AND DECISION SUPPORT TECHNIQUES	18
2.1 Multi-Objective Optimization	18
2.1.1 Distinction Between Single- and Multi-Objective Optimization . . .	18
2.1.2 Multi-Objective Optimization Algorithms	20
2.2 Multivariate Data Visualization	25
2.2.1 The Visualization Process	26
2.2.2 Dimensionality Reduction for Visualization	27
2.2.3 Discrete and Continuous Visualization	30
2.2.4 Basic Visualization Techniques	31
2.2.5 Multivariate Visualization Techniques	37
2.2.6 Visualization Techniques Developed and Implemented by the Design Community	50
2.2.7 Visualization-Enabled Decision Support Software	54
2.3 Multi-Criteria Decision Making	56

2.3.1	Value-Function Based MCDM Methods	58
2.3.2	Proposed Value Function Forms	60
III	EXPERIMENTAL OBSERVATIONS OF PARETO FRONTIERS AND THEIR IMPLICATIONS	62
3.1	Two-Attribute Problem	62
3.2	Four-Attribute Problem	68
3.3	Formalization of Pareto Frontier Geometry	70
3.3.1	Relation Between Simplex Geometry and Pareto Frontiers	71
3.3.2	Structural Similarities Between Pareto Frontiers and Simplices	71
3.3.3	Local Dimensionality of the Pareto Frontier	73
3.3.4	Graphical Interpretation of Multi-Objective Optimality in 2-D	74
3.3.5	Considerations in more than Two Attributes	77
3.3.6	Deviations from Simplex-Like Geometry	78
IV	METHODOLOGY	81
4.1	Assumptions about the Design Problem	84
4.2	Reformulating the Design Analysis on the Pareto frontier	88
4.3	Pareto Simplex Exploration	92
4.3.1	Step 1: Sample the Pareto Frontier	93
4.3.2	Step 2: Assign Coordinates to Sampled Outcome Vectors	95
4.3.3	Step 3: Construct Interpolant for Coordinates	97
4.3.4	Step 4-5: Explore Objective Space using Sampled Data and Interpolated Coordinates and Form Value Function / Make Decision	98
4.4	Multi-Objective Optimization Test Problems	98
V	A NON-DOMINATION LEVEL COORDINATE SYSTEM	100
5.1	Non-Domination Levels	101
5.2	Non-Domination Levels as the Basis for a Coordinate System	101
5.3	Robust Non-Domination Levels	103
5.4	Optional Preprocessing	107
5.5	Example Coordinate Calculations	108
5.6	Application to Example Problems	109

VI A MODIFIED NORMAL-BOUNDARY INTERSECTION COORDINATE SYSTEM	113
6.1 Introduction to Normal Boundary Intersection	114
6.2 Modified NBI Approaches by Motta et al. and Mueller-Gritschneider et al. .	119
6.2.1 Successive Sampling of Subfrontiers	120
6.2.2 Basepoint Redistribution	121
6.2.3 Results	123
6.3 Further Modifications to Improve Sampled Frontier Spacing	124
6.3.1 Generalization to p -Norms	124
6.3.2 Considering Distances to Additional Points	124
6.4 Application to Example Problems	130
VII A SELF-ORGANIZING MAP COORDINATE SYSTEM	137
7.1 Applications of Self-Organizing Maps	138
7.2 A Coordinate System Based On Self-Organizing Maps	139
7.3 Classical Approach to Creating a Self-Organizing Map	141
7.3.1 Self-Organizing Map Topologies	141
7.3.2 Training the Self-Organizing Map	143
7.4 Modified Training for Pareto Simplex Exploration	147
7.4.1 The Simplex Topology	148
7.4.2 Initializing the Codebook Vectors	149
7.4.3 Sequential Training of Subfrontiers	150
7.4.4 Evenly Distributing the Codebook Vectors on the Frontier	151
7.4.5 Summary of SOM Training Process	155
7.4.6 Complications of Identifying Sampled Subfrontiers	156
7.5 Associating SOM Coordinates with the Sampled Pareto Frontier	157
7.6 Application to Example Problems	161
VIII RESULTS	169
8.1 Modeling the Pareto Frontier with Radial Basis Functions	169
8.2 Coordinate-Enabled Visualizations of the Pareto Frontier	172
8.2.1 Prediction Profiler	172
8.2.2 Ternary Plots	183

8.3	Wing Design Problem Case Study	194
8.3.1	Sampled Frontiers and Coordinate Systems	195
8.3.2	Radial Basis Functions and Continuous Visualization with Prediction Profilers	197
8.3.3	Continuous Visualization with Prediction Profilers	198
8.4	Effect of Sampling Size on Coordinate System Quality	202
IX	CONCLUSIONS	215
9.1	Summary and Discussion of Results	215
9.2	Directions for Future Work	222
APPENDIX A	— SOURCES FOR FIGURE 22	224
APPENDIX B	— SOURCE CODE FOR NON-DOMINATION LEVEL COORDINATES	226
APPENDIX C	— SOURCE CODE FOR MODIFIED NORMAL BOUNDARY INTERSECTION	229
APPENDIX D	— SOURCE CODE FOR PARETO SIMPLEX SELF ORGANIZING MAPS	233
APPENDIX E	— WING DESIGN ANALYSIS FUNCTION	243
REFERENCES	248

LIST OF TABLES

1	Categorization of Multivariate Visualization Techniques	31
2	Simple Visualization Techniques	32
3	Multivariate Visualization Techniques	38
4	Capabilities and features of some design support tools	55
5	Examples of Value Functions used in MCDM Methods	60
6	Problem Statements for Exemplar Pareto Frontiers	99
7	Example coordinate calculations for a point on a 3-attribute Pareto frontier	109
8	Independent Variables for Wing Design Problem	195
9	Cases Evaluated in Study of Coordinate System Quality vs. Sample Size . .	203
10	Constant Parameters for Wing Design Problem	243
11	Independent Variables for Wing Design Problem	244
12	Coefficients for Form Factor Regression	246

LIST OF FIGURES

1	Example dominance relations	7
2	3-attribute Pareto frontiers, all objectives are “maximize” a) regular b) irregular due to non-unique $\max(y_3)$	8
3	Notional P - v and T - s diagrams of an ideal Brayton cycle.	13
4	A notional compressor map	14
5	Progression toward a better understanding of Pareto frontiers	15
6	The four lowest-dimensional regular simplices	16
7	A two-dimensional barycentric coordinate grid	16
8	Extent of objective space sampled by different sizes of Monte Carlo simulation	21
9	Comparison of visualization process taxonomies	27
10	A 2-D graph (at right) created by projecting a multidimensional data set (at left) onto a plane	29
11	A 2-D graph (at right) created by slicing a continuous function (at left) at a fixed value of the unplotsed variables	29
12	Two types of line plots	35
13	Three views of the same function. a) Line plot. b) Surface plot. c) Contour plot.	36
14	Notional multivariate visualistics plot	42
15	Notional dimensional stacking plot	43
16	Two notional plots from a trellis display	44
17	Example glyph styles.	45
18	Prediction profiler showing six independent variables and three dependent variables. Horizontal and vertical lines show current slice values.	47
19	Notional HyperSlice showing four independent variables.	48
20	Notional worlds-within-worlds visualization.	49
21	Comparison of bin ordering for two variables, x and y , using hierarchical counting and hyperspace diagonal counting	52
22	Examples of Pareto frontier illustrations that have appeared in journals and conference papers. (Sources are listed in Appendix A)	63
23	Two-attribute NSGA-II results. a) 300 population, 100 generations. b) 30,000 population, 100 generations	64
24	Two-attribute frontier found using a path-tracing algorithm	65

25	Graphical representation of the Jacobian (calculated numerically) at one NSGA-II population member	68
26	The 4-attribute Pareto frontier of Fan Diameter, Range, TOFL, and Weight projected into 3 dimensions	69
27	Relations between subfrontiers of a four-attribute optimization problem, showing similarity to 3-D simplex	73
28	Pareto frontier (darkened) of the 3-attribute problem (4) demonstrating varying Pareto frontier dimensionality	74
29	A baseline design $\tilde{\mathbf{x}}$ and four possible changes to $\tilde{\mathbf{x}}$ that correspond to small increases in four design variables.	75
30	Two possible changes to a baseline design $\tilde{\mathbf{x}}$ due to increasing/decreasing two unconstrained design variables	76
31	The three spaces relevant to Pareto Simplex Exploration, design space, objective space, and coordinate space, and mappings between them.	92
32	Steps in proposed Pareto Simplex Exploration approach	94
33	Notional mapping of a barycentric coordinate grid to a spherical Pareto frontier	96
34	Example Pareto frontier shapes: a) Sphere, b) Truncated Sphere, c) Concave, d) Teardrop, e) Bumpy Sphere (dashed lines enclose dominated regions) . .	99
35	Coordinate curves of a Pareto frontier as defined by the mapping in Figure 33. a) All coordinates b) α_1 coordinates c) α_2 coordinates d) α_3 coordinates . .	102
36	A sampled population, with points of the same non-domination level connected with lines	104
37	Comparison of point distribution a) without hyperplane projection and b) with projection	108
38	Sampled Pareto frontier colored according to: a) Non-domination levels b) Robust non-domination levels c) Non-domination level coordinates	109
39	Example problem boundaries and sampled frontiers; sampled by NSGA-II, colored by corresponding NDLCs	112
40	Major steps of normal boundary intersection	115
41	Normal boundary intersection for two attributes	117
42	Three-attribute spherical Pareto frontier with portion searched by NBI lightened	118
43	Optimization of 2-attribute subfrontiers by a) original NBI b) modification by Motta et al. and Mueller-Gritschneider et al.	121
44	Comparison of p -norms on sampled frontier of sphere example problem . . .	125
45	Basepoints (gray) to be redistributed based on sampled subfrontier (black) .	126

46	Steps for ordering weights for sequentially solving concentric rings	128
47	Three-attribute, five-level weight grid, labeled by order in which corresponding subproblems are solved	129
48	Comparison of improvements to Mueller-Gritschneider et al.'s approach . . .	130
49	Comparison of normal boundary intersection variants applied to test problems	133
50	Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding mod-NBI weights	135
51	Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding NDLCs	136
52	Comparison of data binning by 10×10 rectangular heatmap grid and 10×10 SOM	140
53	a) "Classical" SOM fit to teardrop Pareto frontier. b) Pareto simplex SOM fit to teardrop Pareto frontier	141
54	A 1-D, 20-node SOM trained to an S-shaped data distribution.	142
55	Example 2-D self-organizing map topologies: a) hexagonal, b) square, c) simplex	143
56	The topological distance of nodes relative to node c	145
57	Nodes and edges considered at each stage in Pareto simplex SOM training. a) 1st stage, b) 2nd stage, c) 3rd stage	150
58	SOM trained to teardrop test problem using hierarchical training of subfrontiers.	151
59	Notional codebook vector redistribution, showing iterations from $\mathbf{m}_{initial}$ to \mathbf{m}_{final} , at which the edges have equal length	154
60	Detail of Figure 57 with arrows indicating edges that are never included in N_i	155
61	Visualization of Pareto simplex SOM training process	156
62	a) 1-D subfrontiers found by applying dominance criteria to projected frontier b) Effect on SOM training, cf. Figure 61c	157
63	SOM manifold. Point \mathbf{y}_0 can be expressed as a weighted sum of \mathbf{m}_1 , \mathbf{m}_2 , \mathbf{m}_3	158
64	Upper (d_u) and lower (d_l) limits of distance from a point, y , to the SOM cell $\triangle abc$	161
65	Final trained SOMs for exemplar Pareto frontiers sampled with NSGA-II . .	165
66	Final trained SOMs for exemplar Pareto frontiers sampled with NBI	165
67	Example problem boundaries and sampled frontiers; sampled by NSGA-II, colored by corresponding PSSOM coordinates	166

68	Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding PSSOM coordinates	167
69	Distribution of mod-NBI sampled frontiers in PSSOM coordinate spaces . .	168
70	Comparison of radial basis functions' residual vs. predicted values for y_3 . .	171
71	Example prediction profiler paths. a) With independent variables; b) With barycentric coordinates	174
72	Prediction profiler for sphere example problem	178
73	Prediction profiler for truncated sphere example problem	179
74	Prediction profiler for concave example problem	180
75	Prediction profiler for teardrop example problem	181
76	Prediction profiler for bumpy sphere example problem	182
77	Grid of 1326 weights/coordinates used to assess methods for example problems	183
78	Comparison of coordinate systems for sphere example problem	187
79	Comparison of coordinate systems for truncated sphere example problem . .	188
80	Comparison of coordinate systems for concave example problem	189
81	Comparison of coordinate systems for teardrop example problem	190
82	Comparison of coordinate systems for bumpy sphere example problem	191
83	Effect of a weighted-sum coordinate system on example problems	192
84	Effect of a Chebyshev norm coordinate system on example problems	193
85	Sampled frontiers for wing design problem, plotted in the objective space. a) 5000 point NSGA-II b) 1326 point mod-NBI	196
86	Sampled frontiers for wing design problem, colored by coordinates	206
87	Sampled frontiers for wing design problem, colored by attribute value	207
88	Final trained SOMs for wing design problem	207
89	Residual vs. predicted plots for wing design problem	208
90	Distribution of percent error for RBF validation study	209
91	Prediction profiler for wing design problem, showing four coordinate systems, using coordinates as independent variables	210
92	Augmented prediction profiler for wing design problem using coordinates as independent variables	211
93	Augmented prediction profiler for wing design problem using design variables as independent variables	212

94	Comparison of resampled Pareto frontiers based on PSSOM coordinates created from sparsely sampled frontiers	213
95	Error histograms for RBF validation based on PSSOM coordinates created from sparsely sampled frontiers	214
96	Spline interpolation curve for $e = f(RBMR)$	245

LIST OF SYMBOLS

α	Response metric coefficients (weights) used to scalarize the multi-objective Lagrangian
β	Barycentric coordinates of a self-organizing map; or the specific coordinates of a Pareto simplex SOM
γ	Scale factor used in SOM node redistribution
λ	Lagrange multiplier
\mathcal{L}	Lagrangian function
$\mathcal{P}(A)$	The strong Pareto (sub-)frontier with respect to the attributes in A
ψ	Non-domination level coordinates, or any coordinates for the Pareto frontier
A	A subset of the attributes/objectives, or a generic matrix
D	The design space, i.e. the set of feasible vectors of design variables
f	A design analysis function from the design space to the response space, or a generic function
g	An inverse-like function of the coordinates, or a generic function or constraint
g_x	An inverse-like function of the coordinates that calculates the value of one or more design variables
g_y	An inverse-like function of the coordinates that calculates the value of one or more attributes
h	A generic equality constraint
H_f	Hessian matrix of the design analysis
J_f	Jacobian matrix of the design analysis
k	Number of attributes/objectives; dimensionality of response/objective space
n	Number of design variables; dimensionality of design space
O	The obtainable response/objective space, i.e. the image of D under f
q	Number of inequality constraints
w	A mod-NBI weight coordinate; or a generic weight used in multi-objective optimization
Y	The set of attributes/objectives (cardinality = k)
AOF	Abbreviation for aggregate objective function
KKT	Abbreviation for Karush-Kuhn-Tucker

NBI	Abbreviation for Normal Boundary Intersection
NDLC	Abbreviation for non-domination level coordinates
PSSOM	Abbreviation for Pareto simplex self-organizing map
\square^*	(Superscript) Indicates optimality or satisfaction of KKT conditions for a vector, or invertibility for a set
$a \asymp b$	a and b are equally preferable (indifferent)
$a \succ b$	a is preferred to b
$a \iff b$	a if and only if b
$a \Rightarrow b$	a implies b ; b if a
$a \setminus b$	Set difference; the subset of a that excludes b
$a \succ b$	a dominates b
$a \wedge b$	a and b

LIST OF SYMBOLS

α	Response metric coefficients (weights) used to scalarize the multi-objective Lagrangian
β	Barycentric coordinates of a self-organizing map; or the specific coordinates of a Pareto simplex SOM
γ	Scale factor used in SOM node redistribution
λ	Lagrange multiplier
\mathcal{L}	Lagrangian function
$\mathcal{P}(A)$	The strong Pareto (sub-)frontier with respect to the attributes in A
ψ	Non-domination level coordinates, or any coordinates for the Pareto frontier
A	A subset of the attributes/objectives, or a generic matrix
D	The design space, i.e. the set of feasible vectors of design variables
f	A design analysis function from the design space to the response space, or a generic function
g	An inverse-like function of the coordinates, or a generic function or constraint
g_x	An inverse-like function of the coordinates that calculates the value of one or more design variables
g_y	An inverse-like function of the coordinates that calculates the value of one or more attributes
h	A generic equality constraint
H_f	Hessian matrix of the design analysis
J_f	Jacobian matrix of the design analysis
k	Number of attributes/objectives; dimensionality of response/objective space

n	Number of design variables; dimensionality of design space
O	The obtainable response/objective space, i.e. the image of D under f
q	Number of inequality constraints
w	A mod-NBI weight coordinate; or a generic weight used in multi-objective optimization
Y	The set of attributes/objectives (cardinality = k)
AOF	Abbreviation for aggregate objective function
KKT	Abbreviation for Karush-Kuhn-Tucker
NBI	Abbreviation for Normal Boundary Intersection
NDLC	Abbreviation for non-domination level coordinates
PSSOM	Abbreviation for Pareto simplex self-organizing map
\square^*	(Superscript) Indicates optimality or satisfaction of KKT conditions for a vector, or invertibility for a set
$a \asymp b$	a and b are equally preferable (indifferent)
$a \succ b$	a is preferred to b
$a \iff b$	a if and only if b
$a \Rightarrow b$	a implies b ; b if a
$a \setminus b$	Set difference; the subset of a that excludes b
$a \succ b$	a dominates b
$a \wedge b$	a and b

SUMMARY

The research objective of this dissertation is to develop and demonstrate methods for parameterizing the Pareto frontiers of continuous multi-attribute design problems using barycentric coordinates and in doing so to enable intuitive exploration of optimal trade spaces. This work is enabled by two observations about Pareto frontiers that have not been previously addressed in the engineering design literature. First, the observation that the mapping between Pareto efficient designs and Pareto efficient response vectors is one-to-one almost everywhere suggests that points on the Pareto frontier can be inverted to find their corresponding design variable vectors. Second, the observation that certain common classes of Pareto frontiers are topologically equivalent to simplices suggests that a barycentric coordinate system will be more useful for parameterizing the frontier than a Cartesian coordinate system.

By defining such a coordinate system, the design problem may be reformulated from $\mathbf{y} = f(\mathbf{x})$ to $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ where \mathbf{x} is a vector of design variables, \mathbf{y} is a vector of attributes, and $\boldsymbol{\psi}$ is a vector of barycentric coordinates. Exploration of the design problem using $\boldsymbol{\psi}$ as the independent variables has the following desirable properties: 1) Every vector $\boldsymbol{\psi}$ corresponds to a Pareto efficient design, and every Pareto efficient design has a corresponding $\boldsymbol{\psi}$. 2) The number of $\boldsymbol{\psi}$ -coordinates is equal to the number of attributes, regardless of the number of design variables. 3) Each attribute y_i has a corresponding coordinate ψ_i such that the sign of $\partial y_i / \partial \psi_i$ is positive if the objective is to maximize y_i and negative if the objective is to minimize y_i , i.e. the attributes improve monotonically as their corresponding coordinates increase.

The primary contribution of this work is the development of three methods for forming a barycentric coordinate system on the Pareto frontier, two of which are entirely original to

this work. The first method, named “non-domination level coordinates,” constructs a coordinate system based on the $(k - 1)$ -attribute non-domination levels of a discretely sampled Pareto frontier. This method is easily implemented, but it requires a large sampling of the frontier and the resulting coordinates are locally noisy. The second method is a modified normal boundary intersection multi-objective optimizer that adaptively redistributes its search basepoints in order to sample from the entire frontier fairly uniformly. The weights associated with each basepoint can then serve as a coordinate system on the frontier. This method is based on a modification to an existing NBI-based technique, and is very effective when the design analysis is compatible with gradient-based optimization. The third method, named the “Pareto simplex self-organizing map” uses a modified a self-organizing map training algorithm with a barycentric-grid node topology to iteratively conform a coordinate grid to the sampled Pareto frontier. This method is compatible with any sampled Pareto frontier, does not require excessively many sampled points, and has been found to yield the most consistent coordinate system for most of the example problems that were tested.

In addition to the three methods of constructing a coordinate system, the following contributions will be pursued: (1) derivation of conditions for which constant-dimension Pareto frontiers are homeomorphic to simplices; (2) description of how a Pareto frontier parameterized with barycentric coordinates can be utilized in a design-problem exploration methodology; and (3) demonstration of innovative visualizations that leverage the parameterized Pareto frontier.

The applicability of this work is limited to design problems whose Pareto frontier satisfies certain regularity conditions described in this work. The construction of the non-domination level coordinate system requires a sufficiently large sampling of the Pareto frontier; therefore its applicability is further limited to design problems for which such a sampling can be feasibly obtained. The NBI and self-organizing map based methods are suitable for more sparsely sampled Pareto frontiers. Two additional assumptions about the design problem are made in order to manage the scope of the proposed work. Lastly, only Pareto frontiers

of constant dimensionality are considered, with the implications of this work for variable-dimension frontiers considered briefly in the conclusions.

CHAPTER I

INTRODUCTION AND MOTIVATION

Engineered systems are designed through consideration of a set of multiple requirements and attributes. Within the bounds defined by the requirements, engineers may seek to optimize one attribute directly, or they may attempt to achieve a balance of performance, cost, and other attributes. As discussed by Hayes, design is a learning process in which engineers iteratively employ information seeking, comparison, and selection of design alternatives as their understanding of the problem evolves [56].

The decision support methods that may be used to facilitate this learning include multi-objective optimization for generating potentially optimal designs, multivariate visualization for evaluating the results of trade studies, and multi-criteria decision making (MCDM) methods for defining value. These methods are typically implemented in such a way that the design problem is represented from the perspective of either the k attributes or the n design variables being “free variables.” In many cases, however, design decisions can be simplified by reducing the set of alternatives to its non-dominated subset – the Pareto frontier – which is a portion of the $(k - 1)$ -dimensional boundary of the attainable objective space. The implication is that, when consideration is limited to non-dominated alternatives, the design problem actually has only $k - 1$ degrees of freedom.

The presence of excess degrees of freedom in the existing decision support methods may seem inconsequential, but in practice it can significantly increase the difficulty of understanding the tradeoffs that are implicit in the shape of the attainable objective space. An analogous case is the problem of solving a system of equations that has exactly one solution when the number of equations equals the number of unknown, but an infinite number of solutions when there is just one excess unknown. Moreover, the choice of either the design variables or attributes as free variables for exploring a decision problem leads to its own inherent difficulties. Exploring with the design variables is difficult because many

combinations of design variables will be suboptimal and because the mapping of design variables to attributes can only be understood through a potentially complicated design analysis function. Exploring with the attributes is difficult because they cannot truly be chosen independently, so the decision maker will eventually need to reconcile the freely chosen attribute values with the actual attainable values.

An alternative approach is to treat the *relative weights* of each attribute in determining the overall value as the degrees of freedom. By constraining these weights to sum to one, the k weights have only $k - 1$ degrees of freedom. This approach seems to have potential: it provides the necessary number of degrees of freedom and ensures that the free variables are intuitively meaningful. Indeed, elements of this approach can be seen in many MCDM methods in which the terms of the value function, one term per attribute, are weighted by coefficients that sum to 1.

These value function weights enable exploration to the extent that they can be used as tuning parameters to adjust the attribution of value to outcomes, but they lack many desirable properties that limits their practicality for exploring decision problems. A shortcoming of this MCDM-based approach for exploration is that the weights are inextricably linked to the corresponding value function, so that exploring the objective space using these weights requires the decision maker to first agree that the value function's form is correct. Additionally, the mapping from weights to non-dominated designs for many value function forms is neither injective (one-to-one) nor surjective (onto). This shortcoming severely limits the usefulness of these weights for exploration, as it implies that many different weights will map to the same alternative, while some alternatives will not be mapped to by any weights.

I believe that there is an opportunity for enabling new design space exploration methods, which can reduce the time and number of exploration cycles needed to converge on a value function or ranking of the alternatives, by creating a method of defining a $(k - 1)$ -dimensional coordinate system that parameterizes the Pareto frontier such that the mapping from coordinates to non-dominated alternatives is both injective and surjective. The overarching research objective of this dissertation is to define a method for constructing such a

coordinate system.

This work is enabled by two properties of Pareto frontiers that have not been previously addressed in the engineering design or optimization literature. First, the set of “subfrontiers,” a term used here to describe the Pareto frontiers calculated by considering only a subset of the attributes, is topologically equivalent to the set of faces of a simplex of matching dimensionality. This suggests that a barycentric coordinate system, most commonly used for parameterizing simplices, may also be useful for parameterizing Pareto frontiers. Second, the mapping between Pareto efficient designs in the design space and the objective space is generally one-to-one. This enables exploration of the Pareto frontier in the objective space to be mapped back to the design space, which in turn enables the coordinates to be used for simultaneous exploration of the objective space and the design space.

The particular approach taken here and its implications are summarized in the following hypothesis:

Hypothesis: If the set of alternatives in a continuous, deterministic decision problem is filtered to the set of non-dominated alternatives, the resulting $(k - 1)$ -dimensional Pareto frontier of these alternatives’ image in the objective space can be parameterized using a barycentric coordinate system with k coordinates. By defining such a coordinate system, the design problem may be reformulated from $\mathbf{y} = f(\mathbf{x})$ to $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ where $\boldsymbol{\psi}$ represents the barycentric coordinates. Exploration of the design problem using $\boldsymbol{\psi}$ as the independent variables will have the following advantages over exploration using \mathbf{x} as the independent variables: 1) Every vector $\boldsymbol{\psi}$ corresponds to a Pareto efficient design. 2) The number of $\boldsymbol{\psi}$ -coordinates is equal to the number of attributes regardless of the number of design variables. 3) Each attribute y_i has a corresponding coordinate ψ_i such that the sign of $\partial y_i / \partial \psi_i$ is positive if the objective is to maximize y_i and negative if the objective is to minimize y_i , i.e. each attribute improves monotonically as its corresponding coordinate increases.

Three methods for defining such a coordinate system are presented in this dissertation.

The first method, non-domination level coordinates, bases each coordinate on the non-domination levels of a sampled Pareto frontier that are calculated after the frontier has been projected into a $(k - 1)$ -dimensional space. The second method, mod-NBI, bases the coordinates on the weights in a modified normal boundary intersection algorithm that is able to evenly sample from the entire Pareto frontier. The third method, Pareto simplex self-organizing maps, iteratively trains a coordinate system to uniformly cover a sampled Pareto frontier.

Each of these coordinate systems has the following desirable properties that no existing method of exploring decision problems provides:

- Construction and use of the coordinate systems is not limited by the Pareto frontier's dimensionality, or convexity, except to the extent that the dimensionality must be the same everywhere.
- The coordinates are intuitively meaningful in the sense that increasing a coordinate corresponds to a motion along the Pareto frontier that improves the associated attribute. Because the barycentric coordinates always sum to 1, increasing one coordinate necessitates a decrease in other coordinates with corresponding degradations of their associated attributes as governed by the curvature of the Pareto frontier.
- The coordinate systems are not based on a value-function or aggregate-objective function. Their purpose is to parameterize the Pareto frontier in a way that is independent of any subjective judgment of value.
- The mapping from coordinates to the Pareto frontier is bijective. This ensures that every point on the Pareto frontier has a corresponding tuple of coordinates, and every tuple of coordinates has a corresponding point on the Pareto frontier.
- The coordinate systems can be constructed from any sufficiently dense sampling of the Pareto frontier, or in the case of mod-NBI the coordinate system is defined first and the mod-NBI algorithm creates a corresponding sampling of the frontier.

1.1 Multi-Attribute Design Problem Terminology and Notation

This dissertation concerns design problems in which the system or object being designed is completely characterized by a set of scalar-valued quantities, so that all properties of the system that are not described by these quantities are considered to be either fixed or irrelevant. Some of these quantities can be chosen independently by the designer, while the remainder are calculated as functions of the independent quantities. The independently chosen quantities will be called *design variables*, the calculated quantities *response metrics*, and the function that maps design variables to response metrics the *design analysis*.

Let n denote the number of design variables and x_i denote the i -th design variable. Then a *design* is an n -tuple of values, with each value corresponding to one of the design variables. For simplicity, we consider as the only constraints a set of closed intervals $[l_i, u_i]$, one interval associated with each design variable x_i , that denotes the variables' ranges of feasible values. A design for which each design variable takes values within its feasible interval is called a *feasible design*, and the set of all feasible designs is called the *feasible design space* or simply the *design space* and is denoted by D . In order to make use of intuitive concepts such as nearness and similarity of designs, we consider the design space to be a subset of the vector space \mathbb{R}^n . Thus each n -tuple of design variables that defines a design will be represented by a vector \mathbf{x} . Correspondingly, the feasible design space can be defined as $D = \{\mathbf{x} \in \mathbb{R}^n | x_i \in [l_i, u_i] \text{ for } i \in 1, \dots, n\}$.

Let k denote the number of response metrics, and let y_j denote the j -th response metric. As in the case of the design space, we consider a k -tuple of response metric values to be a vector, called the *outcome* of its corresponding design, and we give it the symbol \mathbf{y} .

The design analysis is denoted by f , defined as

$$f : \begin{cases} \mathbb{R}^n \rightarrow \mathbb{R}^k \\ \mathbf{x} \Rightarrow \mathbf{y} \end{cases}$$

where the notation indicates that f is a mapping from the field \mathbb{R}^n to the field \mathbb{R}^k that maps a particular vector \mathbf{x} to \mathbf{y} ; equivalently, $\mathbf{y} = f(\mathbf{x})$.

The *objective space*, $O \subset \mathbb{R}^k$, is the image of the feasible design space under the design

analysis' mapping, i.e. $O = \{\mathbf{y} \in \mathbb{R}^k | \mathbf{y} = f(\mathbf{x}), \mathbf{x} \in D\}$. Whereas the feasible design space D was easily defined as the subset of \mathbb{R}^n that satisfies the upper and lower bounds on the values of the design variables, no similarly convenient description of the extent of the objective space can be given. Note that while D is defined to be a hypercube, O is generally irregularly shaped but bounded.

The *design problem* is the problem of choosing a single, i.e. the “best,” design based on the designer’s interpretation of the value of some *attributes* of the design. Without loss of generality, it is here presumed that the attributes comprise all of the response metrics and none of the independent variables. (Consequently the term “attribute” will be used synonymously with “response metric” in the remainder of this dissertation.) It is assumed that each attribute has a corresponding objective, maximize or minimize, that defines a preference ordering within that attribute, i.e. other attributes being equal. No assumption about how designer’s preferences weigh multiple conflicting objectives is made.

1.1.1 Pareto Optimality

A *multi-objective optimization problem* has the form $\min_{\mathbf{x}} \mathbf{y}$ s.t. $\mathbf{x} \in D$. The minimization indicated in multi-objective optimization is defined by a partial order, most often the Pareto dominance order. Pareto dominance may be defined as being *strong* or *weak*. A vector of attributes \mathbf{y} weakly dominates another vector $\hat{\mathbf{y}}$ if $\mathbf{y} \neq \hat{\mathbf{y}}$ and $\mathbf{y}_i \leq \hat{\mathbf{y}}_i$ ($\mathbf{y}_i \geq \hat{\mathbf{y}}_i$) when the i -th attribute’s objective is to minimize (maximize). A vector of attributes \mathbf{y} strongly dominates another vector $\hat{\mathbf{y}}$ if $\mathbf{y}_i < \hat{\mathbf{y}}_i$ ($\mathbf{y}_i > \hat{\mathbf{y}}_i$) when the i -th objective is to minimize (maximize).

A vector \mathbf{y} is called *weakly non-dominated* if no other vector $\hat{\mathbf{y}} \in f(D)$ strongly dominates it and is called *strongly non-dominated* if no other vector weakly dominates it. A vector that is strongly non-dominated is necessarily weakly non-dominated, but the converse is not true. The set of all strongly non-dominated attribute vectors is called the Pareto frontier, a discrete sampling of which constitutes a solution to a multi-objective optimization problem.

The dominance relations of a set of points are illustrated in Figure 1 for a two-attribute problem in which both objectives, y_1 and y_2 , are to be minimized. Point a is strongly dominated by the points that lie within the cone indicated by the dashed lines and is

weakly dominated by point b , which lies on the boundary of the cone. The gray points at the bottom of the set are weakly non-dominated because the black point to their left weakly dominates them (as do any gray points to each point's left), but no points strongly dominate them.

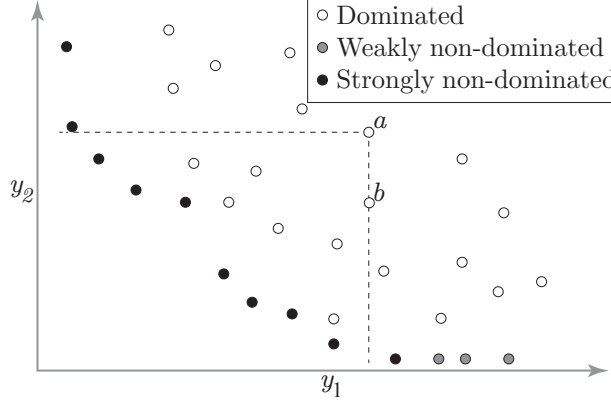


Figure 1: Example dominance relations

1.1.2 Definition of Subfrontiers

The concept of a “subfrontier” of the Pareto frontier will be used in subsequent chapters to describe the geometry of the Pareto frontier. Let Y denote the set of response metrics. Y is not to be confused with O , the feasible objective space; rather, Y is a set with k elements, which are the k attributes themselves. For example, in a decision to buy a car, Y may be the set {price, fuel economy, horsepower, safety rating}. Let A denote a subset of Y , such that A can be used to describe problems in which some of the attributes in Y are ignored, i.e. the attributes in $Y \setminus A$ do not affect the value.

The symbol $\mathcal{P}(A)$ denotes the Pareto frontier found by solving the multi-objective optimization problem whose objectives are the set of attributes A . $\mathcal{P}(A)$ will be called an n -subfrontier of $\mathcal{P}(Y)$, where n is the number of attributes in A . The 1-subfrontiers are thus simply the individual optima of the attributes in Y , and the k -subfrontier is the “full” Pareto frontier $\mathcal{P}(Y)$.

1.1.3 Pareto Frontier Regularity Condition

Multi-objective optimization for design typically focuses on strongly non-dominated designs since any weakly non-dominated design can be improved to a strongly non-dominated design without tradeoff. However, certain complications arise in the discussion of subfrontiers when attention is restricted to strongly non-dominated designs. In particular, designs that are strongly non-dominated with respect to the full set of attributes, Y , may be only weakly non-dominated with respect to a subset of the attributes, A . To resolve this issue, we limit consideration to “regular” frontiers, which we define by the following condition: a Pareto frontier is regular if no vector of $\mathcal{P}(Y)$ is strictly weakly non-dominated with respect to any subset of attributes $A \subseteq Y$.

Figure 2a shows a regular Pareto frontier, and Figure 2b shows an irregular Pareto frontier. The irregularity in Figure 2b arises from the non-unique optima for y_3 . When the Pareto frontier is projected into y_3 , i.e. when considering $\mathcal{P}(\{y_3\})$, the non-unique optima of $\max(y_3)$ become indistinguishable. These points are in both $\mathcal{P}(\{y_1, y_2, y_3\})$ and $\mathcal{P}(\{y_3\})$ in violation of the aforementioned condition, thus the Pareto frontier is irregular.

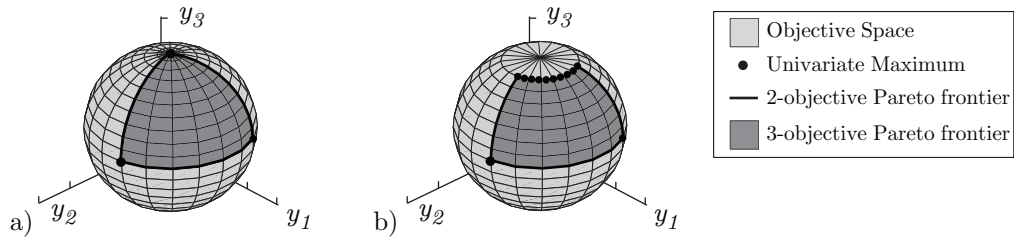


Figure 2: 3-attribute Pareto frontiers, all objectives are “maximize” a) regular b) irregular due to non-unique $\max(y_3)$

As a practical rule to check if a particular Pareto frontier is regular, it is usually sufficient to check if each optimum of the individual attributes in $\mathcal{P}(Y)$ is unique. If so, it is unlikely that strictly weakly non-dominated points will arise in the higher subfrontiers. While irregularity does not preclude the use of the barycentric coordinate system described here, it indicates that the structure of the Pareto frontier is in some sense not simplex-like, so that barycentric coordinate system may be less meaningful than it would be for a regular

frontier.

An irregular example problem whose frontier is similar to the one shown in Figure 2b will be considered in Chapters 5–7 in order to demonstrate the applicability of barycentric coordinates to irregular frontiers. The results generally show that any problems that arise due to the irregularity tend to be localized within the area around the irregularity.

1.2 Approach to Multi-Attribute Design Problems and Need for a Coordinate System on the Pareto Frontier

As stipulated in the previous section, the eventual selection of a particular design as the “solution” to the multi-attribute design problem is based only on consideration of the *attributes*. From the perspective of decision theory, this implies the existence of a *value function* of the attributes, $value = f(y_1, \dots, y_k)$, that has the property that designs whose attributes yield a greater value are preferred. Design optimization may then be viewed as the process of seeking the design whose attributes maximize the value.

Many real engineering design decisions, however, are made without explicitly formulating a value function. Consider for example the countless “low-level” design decisions that are made in the course of a program that are too far removed from the ultimate value function, e.g. return on investment, to feasibly calculate, let alone optimize, the value. Instead, these decisions may be based on lower-level attributes that are known to contribute to the highest-level value function.¹ For example, in the design of an aircraft engine it is known that, other things being equal, engines with low specific fuel consumption (SFC) and low weight are preferred, as lower values of these attributes are known to improve the specific range (i.e. miles flown per pound of fuel consumed) of the aircraft in which the engine will eventually be installed. It is often more practical for engineers designing the engine to base decisions on SFC and engine weight than on specific range because, although the specific range is more indicative of the engine’s true value, it cannot be calculated without detailed knowledge of the airframe in which the engine will be installed. The highest-level value

¹In these cases it could be argued that value gained by explicitly calculating and maximizing the value function would be offset by the value lost as a result of investing resources to conduct this optimization. Thus the highest-value programmatic decision is to make the design-decisions based on lower-level attributes.

function would, of course, not be based solely on specific range but would also include the effects of other intermediate-level attributes such as development and manufacturing costs and reliability. It is assumed here that the set of (low-level) attributes being considered is complete in the sense that all such intermediate-level attributes and the highest-level value are functions of only the attributes being considered.

The difficulty of making decisions based on multiple low-level attributes instead of a single value function is that the attributes almost certainly conflict; i.e. there is no single design that simultaneously optimizes each attribute. Continuing the engine design example above, it is unlikely that a single design exists that simultaneously minimizes both the SFC and weight. More likely, the components or technologies that might be added to the engine to improve the SFC will also increase the engine's weight. It is the responsibility of the designer to find a design whose attributes are balanced such that they maximize the highest level value, which, as described above, is not explicitly calculated, but can only be approximated by the designer.

One of the greatest challenges related to seeking such balanced designs is the lack of a convenient description of the set of alternative designs. As described by deterministic decision theory, the “best” design can be selected by a series of pairwise comparisons in which the designer is asked “Would you prefer design A or design B ?” The practicality of such a process is strongly related to the ability to identify a series of designs that quickly converges to the highest-value design.

When a closed-form value function is known, it can be used as the objective function in a numerical optimization algorithm to automate this alternative-identification and comparison process. The rate at which the process converges (i.e. the number of optimization iterations required) will then be a direct result of the suitability of the particular optimization algorithm to the particular problem being solved. For the case being considered here, however, in which the value function is not explicitly known, the search for potentially better alternatives cannot be automated for two reasons. First, the decision maker must consider each pair of alternatives and make a subjective judgment as to which is better. Second, because the value cannot be calculated numerically, it is difficult to identify a direction of

motion in the objective space, from a baseline design, which corresponds to the steepest increase in value or another path believed to quickly converge to the highest-value designs.

For real design problems, the attainable attribute values are typically known only as discrete points in the objective space – the attributes corresponding to the particular point designs that have been evaluated with the design analysis – and the gradients of the individual attributes can only be calculated numerically by additional sampling. The Jacobian of the design analysis may be calculated numerically at a baseline vector of attributes to obtain a local linearization of the design analysis. From this linearization, it can be determined if there is a feasible “search direction” from the baseline design along which every attribute improves. In this case the baseline design is known to be suboptimal and the designer would prefer to move along the search direction to a higher-value design.

Once all such moves have been exhausted, and some attributes can only be improved at the expense of others, then by definition the current design lies on the Pareto frontier. At this point, the designer must decide if the balance among the attributes at the current design has acceptably high value to stop searching for better alternatives. In other words, does the designer believe that other alternatives exist that would yield sufficiently higher value to justify the expense of continuing to search? If the search is to be continued, it must then be decided how to continue the search. If the current design seems nearly optimal, then a local search along the Pareto frontier may be conducted by again numerically calculating derivatives at the current design and assessing if the tradeoffs implied by these derivatives are favorable, e.g. is the degradation in y_2 required to achieve a specified improvement in y_1 sufficiently low? On the other hand, if the current design has a very unfavorable balance of the attributes, the designer may wish to begin a new search in a different region of the design space, with the hopes that this will eventually lead to sampling from higher-value regions of the Pareto frontier.

The process described in the previous paragraphs can be difficult and time consuming. A significant drawback of this process is that the designer must constantly be “in the loop,” driving the search toward higher-value attribute vectors. When the design analysis is slow, this becomes an inefficient process of alternately steering the search and waiting for the

next design’s attributes to be calculated. The number of search iterations required is driven by the designer’s ability to make purposeful changes to the current design such that the series quickly converges to high-value designs. This ability may loosely be equated with the designer’s “engineering expertise,” related to the particular problem being considered; however, for problems with many design variables, attributes, and constraints, even the most experienced engineers may have difficulty inferring the location of potentially optimal regions of the design space based on examining the sampled attribute vectors.

This process of inferring potentially optimal regions of the design space may be aided by examining the sampled attribute vectors (for example, by using data visualization) to learn something about the objective space and the mapping from design variables to attributes. As described above, the attribute vectors are generally known only as a set of discrete points in the objective space, with some subset of these points comprising a sampled Pareto frontier. Based on these sampled points, the designer must infer the location (or absence) of potentially higher-value attribute vectors in the objective space, and the locations of their corresponding design variable vectors in the design space.

Attempts to infer the locations of potentially better designs are complicated by two properties of the objective space and the design analysis. First, since it is known that the Pareto frontier lies on the boundary of the feasible objective space, by definition the attributes cannot be improved independently; rather, each attribute can only be improved at the expense of others as dictated by the local curvature of the Pareto frontier. When the Pareto frontier is only known as a set of sampled point designs, the local curvature can only be determined by numerically calculating derivatives around a baseline point. Second, even if the location of a potentially desirable point in the objective space can be inferred, a corresponding point in the design space must be found. Again, this requires an understanding of the derivatives of the attributes with respect to the design variables.

The work presented in this dissertation has been motivated by the belief that the search for high value designs in a multi-attribute design problem can be aided by defining a coordinate system that parameterizes a sampled Pareto frontier. This belief is in turn based on the belief that the difficulty of identifying potentially high-value designs is largely related

to the difficulty of *describing the Pareto frontier* in a way that is intuitively meaningful to the designer.

Coordinate systems have many applications in science and engineering for describing relations between variables that are constrained to vary along a manifold whose dimensionality is lower than the number of variables. A well known example is the case of a closed thermodynamic system involving a perfect gas. Although many state variables can be defined, including temperature, pressure, volume, entropy, density, internal energy, enthalpy, and others, this system is known to be two-dimensional in the sense that it is fully defined by the choice of values for any two of the state variables. By representing the system with different pairs of the state variables as coordinates, different processes can be more easily illustrated. For example, Figure 3 shows P - v and T - s diagrams of an ideal Brayton cycle. Depending on whether P and v or T and s are used as coordinates, different segments of the cycle are plotted as straight lines. Consequently, the choice of either P and v or T and s as coordinates affects the ease of representing and understanding certain types of thermodynamic processes.

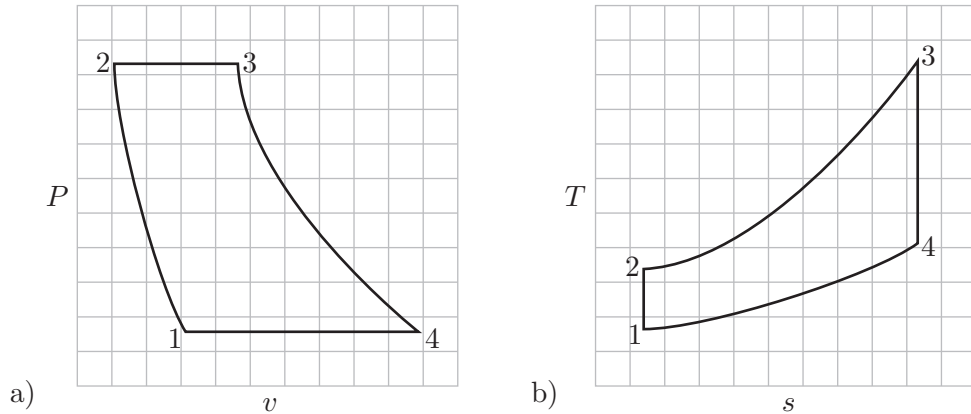


Figure 3: Notional P - v and T - s diagrams of an ideal Brayton cycle.

A second example application, illustrated in Figure 4, relates to the use of compressor maps in gas turbine design. These maps relate four quantities in a two-dimensional manifold: the pressure ratio, corrected mass flow, rotational speed (ω), and efficiency (η). As shown in Figure 4, the compressor map does not fill the entire pressure ratio vs. corrected mass

flow space. Consequently, using these two quantities as independent variables can lead to the specification of unattainable efficiencies or rotational speeds. The iso-efficiency curves generally form ellipses and are thus unsuitable for use as coordinates. Instead, a coordinate system based on the rotational speed and “R-lines” is used. The R lines are constructed to be roughly parallel to the surge line; they generally lack a meaning outside of their use in this coordinate system [96]. This example demonstrates the use of a custom coordinate system, in which one coordinate is not directly derived from a physical quantity, to aid in understanding a complex manifold.

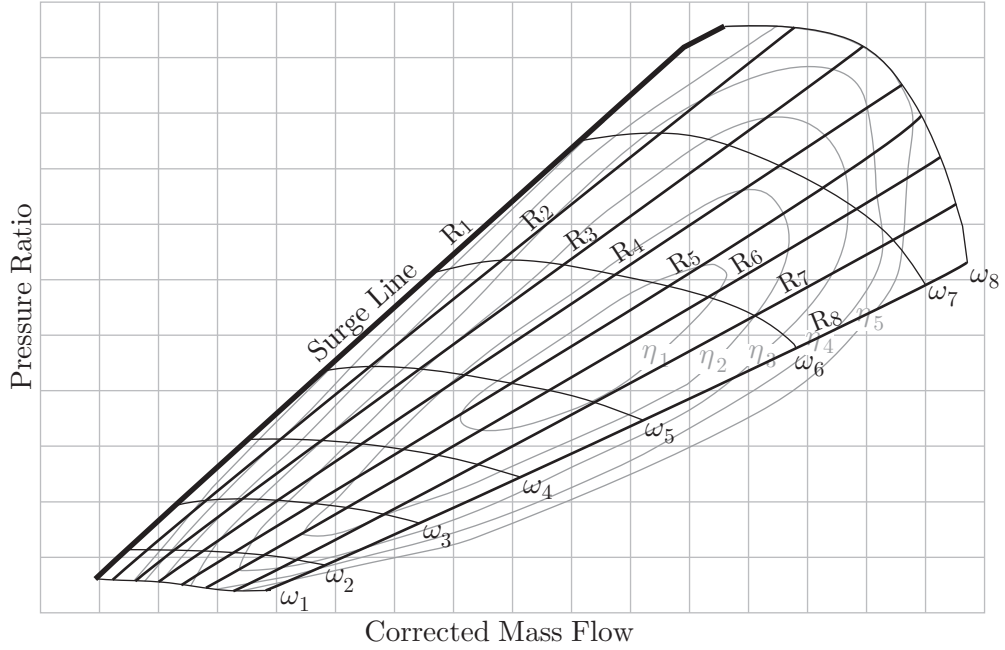


Figure 4: A notional compressor map

The goal of this dissertation is to create a method for defining a coordinate system on a general Pareto frontier that enables fast, intuitive exploration in the same way that the compressor map enables exploration of compressor performance. In order to achieve this goal, our ability to represent the Pareto frontier must evolve from the “cloud of points” representation to a representation of the frontier as a continuous manifold. The coordinate system then provides a convenient mechanism for describing this manifold. These concepts are illustrated in Figure 5.

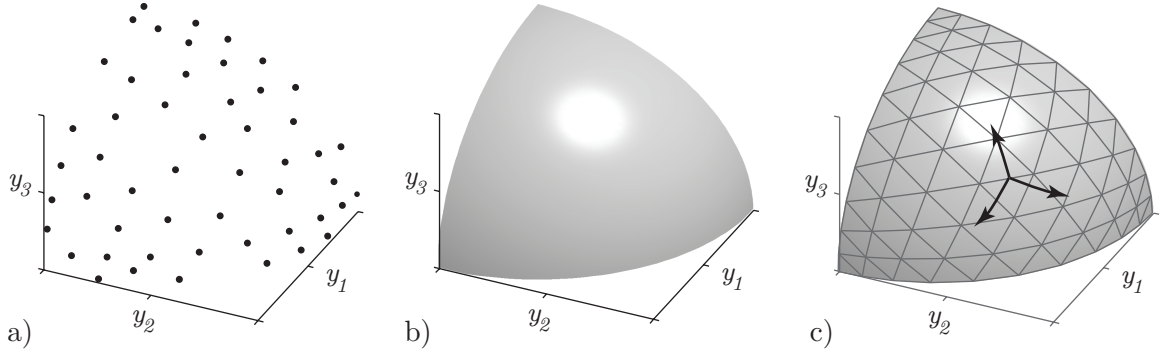


Figure 5: Progression toward a better understanding of Pareto frontiers: a) representation as discrete points, b) representation as a continuous manifold, c) representation as a manifold parameterized by coordinates

An important property of the coordinate system is that it is confined to parameterize only the (generally $k - 1$ -dimensional) Pareto frontier, which is a subset of a larger k -dimensional objective space. This property distinguishes the coordinates from the attributes, which as described above cannot be independently changed without “falling off” the Pareto frontier into either the suboptimal interior of the feasible objective space or the infeasible/unattainable region of the objective space.

1.3 *Simplices and Barycentric Coordinates*

A primary goal of this dissertation is the development of coordinate systems that can be used to parameterize arbitrary Pareto frontiers. As will be described in Chapter 3, a regular Pareto frontier is topologically equivalent to a simplex of matching dimensionality. Consequently, a *barycentric coordinate system* is particularly useful for parameterizing the frontier.

A simplex is a p -dimensional polytope that is the convex hull of $p + 1$ vertices. Simplices are therefore analogous to triangles in arbitrary dimensions. The four lowest-dimensional simplices are shown in Figure 6.

A barycentric coordinate system represents a point on a simplex as a weighted sum of its vertices: $\mathbf{y} = \alpha_1 \mathbf{y}_1 + \dots + \alpha_k \mathbf{y}_k$, $\sum \alpha_i = 1$ where the α_i are the barycentric coordinates and $\mathbf{y}_1, \dots, \mathbf{y}_k$ are the vertices. The name “barycentric” refers to the fact that a point

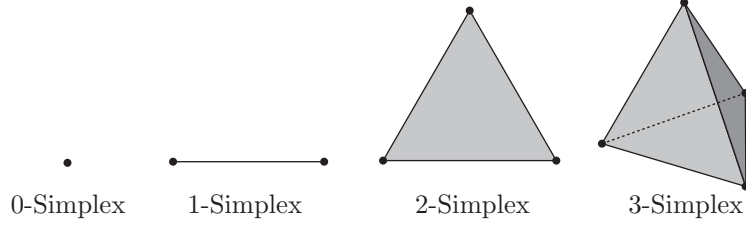


Figure 6: The four lowest-dimensional regular simplices

in/on a simplex has barycentric coordinates equal to the masses that, if placed at the corresponding vertices, would make this point the centroid of the system of masses [29, p. 216]. Correspondingly, in two-dimensions a point's barycentric coordinates are equal to the relative areas of the triangles formed by subdividing the simplex into three triangles that share this point as a common vertex. An example a 2-D barycentric coordinate system is shown in Figure 7, which also illustrates the relative areas of the triangles formed by the point with barycentric coordinates $(0.3, 0.2, 0.5)$.

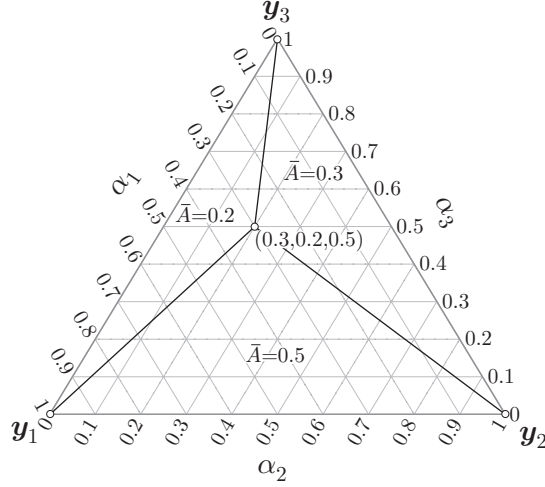


Figure 7: A two-dimensional barycentric coordinate grid

1.4 Outline

The remainder of this document is organized as follows:

- Chapter II surveys the literature related to design decision support techniques.
- Chapter III presents the results of preliminary experimental studies on the structure

of Pareto frontiers that have motivated the coordinate-based exploration approach.

- Chapter IV describes the coordinate-based exploration approach and defines the example problems on which it will be demonstrated.
- Chapter V introduces the non-domination level coordinate system.
- Chapter VI introduces the mod-NBI coordinate system.
- Chapter VII introduces the Pareto simplex self-organizing map coordinate system.
- Chapter VIII compares the results of the three coordinate systems and demonstrates their use for design space exploration.
- Chapter IX presents conclusions and discusses the broader implications of this work.

CHAPTER II

LITERATURE SURVEY OF MULTI-OBJECTIVE OPTIMIZATION AND DECISION SUPPORT TECHNIQUES

As described in the previous chapter, engineering design may be framed as a decision making process in which the goal is to decide on the best alternative design. The formal process of formulating and making a decision, called decision analysis, comprises three primary steps in its deterministic formulation. First, *alternative courses of action* are generated, with the decision being to irrevocably choose one such course of action. Second, the set of *outcomes* that are possible results of the different courses of action are determined. Third, the possible outcomes are ranked according to their *value* to the decision maker, and the alternative with the highest value is selected [63, 73].

As discussed by Hayes, actual decision analysis processes are often exploratory, requiring flexibility in information seeking, comparison, and eventual selection of design alternatives [56]. This chapter surveys the state of the art for three primary classes of methods that may be used in exploratory decision analysis processes: multi-objective optimization, for generating non-dominated alternatives, multivariate data visualization, for understanding the design analysis function's implications on the attainable outcomes, and multi-criteria decision making methods, for attributing value to the outcomes.

2.1 Multi-Objective Optimization

2.1.1 Distinction Between Single- and Multi-Objective Optimization

Optimization can be viewed as a process of generating, analyzing, and ordering designs from best to worst with the goal of finding the best design(s). Single-objective optimization seeks to minimize an objective function subject to some inequality and equality constraints. A general strategy for single-objective minimization is to seek feasible designs with successively lower values of the objective functions. Assuming the codomain of the objective function is a subset of the real numbers, then the natural ordering of the real numbers defines an

ordering relation, \leq , that is used to compare designs. For any three designs \mathbf{a} , \mathbf{b} , and \mathbf{c} , this relation is transitive ($\mathbf{a} \leq \mathbf{b} \wedge \mathbf{b} \leq \mathbf{c} \Rightarrow \mathbf{a} \leq \mathbf{c}$), antisymmetric ($\mathbf{a} \leq \mathbf{b} \wedge \mathbf{b} \leq \mathbf{a} \Rightarrow \mathbf{a} = \mathbf{b}$), and total ($\mathbf{a} \leq \mathbf{b}$ or $\mathbf{b} \leq \mathbf{a}$). The last of these properties, totality, implies that the objective function can be used to compare any two designs to determine that one is preferable to the other or that both are equally preferable.

The existence of a total order is the fundamental difference in single-objective optimization compared to multi-objective optimization. The ordering of designs in multi-objective optimization is *partial*, meaning that not all designs can be compared: given two designs \mathbf{a} and \mathbf{b} , it is not necessarily true that $\mathbf{a} \leq \mathbf{b}$ or $\mathbf{b} \leq \mathbf{a}$; \mathbf{a} and \mathbf{b} might be incomparable.

The general concept of multi-objective optimization does not prescribe the choice of a particular partial ordering method. In this work, as in most multi-objective optimization applications, the partial ordering will be defined by the strong Pareto dominance criteria, which is defined in Section 1.1.1.

The relation between single- and multi-objective optimality can also be understood by examining each case's necessary optimality conditions. For constrained single-objective minimization, the Lagrangian is the function

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \equiv -f(\mathbf{x}) + \sum_{j=1}^m \lambda_j h_j(\mathbf{x}) + \sum_{l=1}^q \lambda_{m+l} g_l(\mathbf{x})$$

where $f(\mathbf{x})$ is the objective function, $h_j(\mathbf{x}) = 0$, $j = 1, \dots, m$ are the m equality constraints, and $g_l(\mathbf{x}) \leq 0$, $l = 1, \dots, q$ are the q inequality constraints. The necessary conditions for optimality are given by the Karush-Kuhn-Tucker (KKT) conditions [81]:

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= -\nabla f(\mathbf{x}) + \sum_{j=1}^m \lambda_j \nabla h_j(\mathbf{x}) + \sum_{l=1}^q \lambda_{m+l} \nabla g_l(\mathbf{x}) = \mathbf{0} \\ \lambda_{m+l} \cdot g_l(\mathbf{x}) &= 0 \text{ for } l \in 1, \dots, q \\ \mathbf{h}(\mathbf{x}) &= \mathbf{0} \\ \mathbf{g}(\mathbf{x}) &\leq \mathbf{0} \end{aligned} \tag{1}$$

The λ_j and λ_{m+l} are the Lagrange multipliers of the equality and inequality constraints respectively. \mathbf{x}^* is called a KKT point if it satisfies these four conditions.

The first KKT condition is the most insightful—the gradient of the Lagrangian must be equal to zero at a locally optimal point. The second condition states that lagrange multiplier

of any inactive inequality constraint must be zero. The third and fourth conditions simply enforce the equality and inequality constraints.

For multi-objective minimization, the Lagrangian is slightly modified from its single-objective form to aggregate the multiple attributes into a single function [81, 58]:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = - \sum_{i=1}^k \alpha_i \mathbf{f}(\mathbf{x}) + \sum_{j=1}^m \lambda_j \mathbf{h}_j(\mathbf{x}) + \sum_{l=1}^q \lambda_{m+l} \mathbf{g}_l(\mathbf{x}) \text{ with } \alpha_i \geq 0 \text{ and } \sum_{i=1}^k \alpha_i = 1$$

Correspondingly, the first KKT condition is replaced with

$$\nabla \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = - \sum_{i=1}^k \alpha_i \nabla \mathbf{f}(\mathbf{x}) + \sum_{j=1}^m \lambda_j \nabla \mathbf{h}_j(\mathbf{x}) + \sum_{l=1}^q \lambda_{m+l} \nabla \mathbf{g}_l(\mathbf{x}) = 0$$

in the multi-objective case. The condition $\sum_{i=1}^k \alpha_i = 1$ is sometimes replaced with the condition that not all $\alpha_i = 0$.

This condition is equivalent to the corresponding single-objective condition when the single objective is a weighted sum. This may seem to conflict with the well known result that the weighted-sum aggregate objective function is unable to find non-dominated points on concave portions of the Pareto frontier [32]. The resolution of this conflict is that non-dominated designs in concave portions of the Pareto frontier are *saddle-points*, not maximizers, of the weighted sum [58].

When the objectives and constraints are convex functions, the above condition is both necessary and sufficient with the additional constraint that the α_i are strictly greater than 0 [106]. For general twice-differentiable functions, significantly more complicated second-order sufficient conditions for local optimality are needed. Details of these conditions are given by Miettinen [106].

2.1.2 Multi-Objective Optimization Algorithms

A simple method of generating alternative designs is to randomly sample vectors from the feasible design space according to a specified joint distribution for the design variables. Although this method is easy to implement, it is inefficient and ineffective when the intent is to generate a set of potentially optimal designs [18, p. 310]. This inefficiency is easily demonstrated experimentally. Figure 8 shows the convex hulls in the objective space of different sized random samplings of designs for a problem with two attributes, which are

modeled as linear functions of twelve design variables. Even for the case of 10 billion samples, which if stored at double precision would require 104 gigabytes of memory, the convex hull is determined by only a handful of points and does not closely approximate the true objective space boundary, a portion of which is the Pareto frontier.

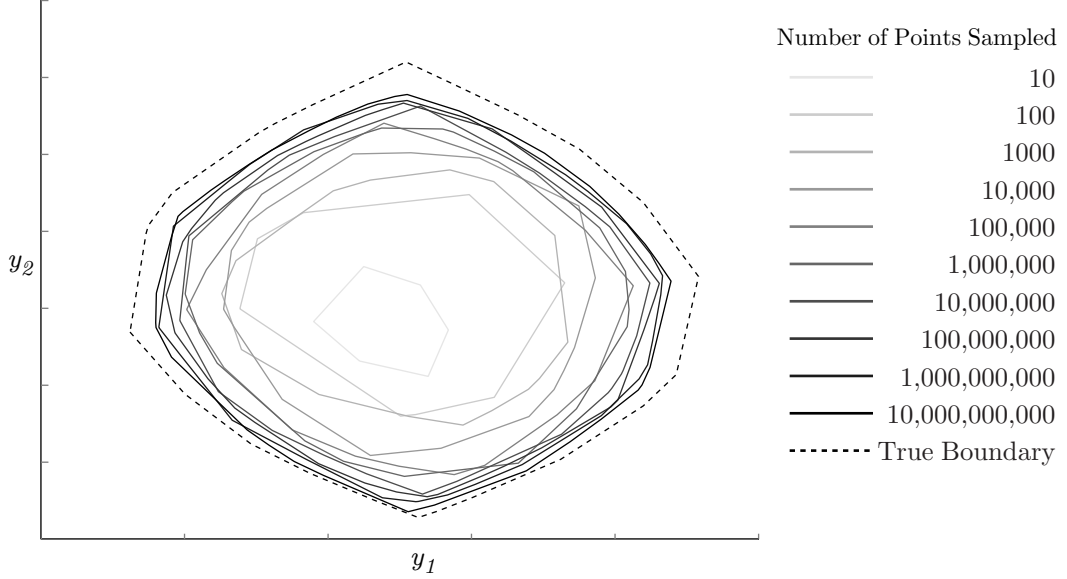


Figure 8: Extent of objective space sampled by different sizes of Monte Carlo simulation

Structured sampling, for example sampling according to a design of experiments, also fares poorly on account of the relative sparsity of non-dominated designs. For example, for the linear problem in Figure 8 it is known that the vertices of the Pareto frontier are among the 4096 points in the two-level full-factorial design of experiments. However, of these 4096 points, only five are non-dominated.

The most practical approach to densely and evenly sampling the Pareto frontier is to use multi-objective optimization. Despite having a name that seems to imply that these algorithms do more than single-objective optimization, they are actually built on a weaker premise – partial ordering instead of total ordering, as described in Section 2.1.1. Indeed, the reason that multi-objective optimizers return a set of efficient designs rather than a single design is that they lack the information needed to determine which of the efficient designs is most preferred. Whereas single-objective optimization can be viewed as an automated

implementation of decision analysis, multi-objective optimization is an automation of only the first two steps of decision analysis, generation of alternatives and evaluation of outcomes, guided by a simple dominance criterion.

Three classes of multi-objective optimization techniques are considered here: techniques that iteratively apply parameterized single-objective optimization, evolutionary algorithms that simultaneously optimize a population of alternatives, and continuation algorithms that trace a path of alternatives along the Pareto frontier.

2.1.2.1 Iterative Single-Objective Optimization

A wide class of algorithms involves repeated optimization of an aggregate objective function (AOF) of the form $g(w_1h(y_1), \dots, w_kh(y_k))$, where the w_i are weights and the y_i are the attribute values. As the weights are varied, different non-dominated designs are sampled. Surveys of such techniques have been written by Messac [103] and Marler and Arora [90].

The primary challenge of these methods is the choice of an AOF form that is appropriate for the particular design problem being addressed. Since the purpose of this the AOF in this application is to generate alternatives that will later be examined before making a final decision, it is not critical for the AOF used here to represent the design's actual value to the designer. In fact it almost certainly will not; instead, the AOF is chosen to provide an even sampling of the Pareto frontier as the weights are varied.

Perhaps the best known AOF for this application is the weighted sum function, $g = \sum_{i=1}^k w_i y_i$. This function has well known shortcomings including the inability to sample from concave portions of the Pareto frontier and the irregular spacing of sampled points for a regularly spaced grid of weights [32, 91]. Messac et al. [105, 101] have shown that the curvature of the AOF must be greater than the curvature of the concave regions of the Pareto frontier in order to sample points in those regions. Consequently, the weighted Chebyshev norm, which has infinite curvature, is able to find all non-dominated solutions regardless of convexity [106], but it too suffers from irregular spacing of sampled solutions as the weights are varied¹.

¹This is demonstrated in Figure 84 on page 193

A second strategy is to constrain the problem to a 1-D search along a specified direction in the objective space and then maximize or minimize the distance along this direction. Essentially, these approaches work by restricting the search space in such a way that the Pareto dominance ordering becomes a total ordering. This strategy is employed by the ε -constraint method [54], which uses a grid of inequality constraints on $k - 1$ of the attributes; the method of proper equality constraints, which is similar but uses equality constraints in place of the inequality constraints [84]; and by normal boundary intersection and the normalized normal constraint method, which constrain each solution to be normal to a specified point on the $(k - 1)$ -dimensional simplex.

The normal boundary intersection (NBI) [33] and the (normalized) normal constraint (NC) [102, 104] methods are explicitly based on barycentric coordinates and simplices, and thus these methods seem like good candidates on which to base a barycentric coordinate system for the Pareto frontier. Indeed, this approach is taken in Chapter 6.

The NBI method constructs a simplex whose vertices are the individual optima, i.e. the k points in the objective space that optimize each of the k attributes individually. A barycentric coordinate grid is constructed on this simplex, and the resulting coordinate tuples are used to parameterize constraints that simplify the optimization problem to a 1-D line search along a quasinormal direction to the simplex. Solving this optimization problem for a regular grid of coordinate tuples yields a set of non-dominated designs that is usually fairly evenly spaced.

Because NBI limits its searches to a barycentric grid on the convex hull of the individual optima, portions of the Pareto frontier that extend outside this simplex are not sampled, and not every solution found within the simplex is guaranteed to be globally non-dominated [33]. The NC method takes a conceptually similar approach as NBI but avoids these difficulties by enlarging the simplex to encompass the entire Pareto frontier, thus ensuring that the entire frontier can be sampled. For better efficiency, coordinate tuples on the enlarged simplex that cannot possibly yield globally non-dominated attribute vectors are discarded before the parameterized optimization subproblems are solved [104].

Except in special cases, none of the above methods exhibits a one-to-one mapping between the barycentric coordinates and the entire Pareto frontier. Often in AOF approaches, multiple combinations of weights will yield the same non-dominated design. Similarly, some non-dominated designs may not have corresponding weights. These limitations, and the typically uneven spacing of solutions for a uniform grid of weights, generally cause the particular weight vectors to be devoid of meaning beyond their use as a mechanism to populate the objective space with a variety of solutions [103, 89]. For convex Pareto frontiers, NBI creates a one-to-one mapping between the barycentric coordinates and the portion of the Pareto frontier enclosed by the convex hull of the individual optima while leaving the remainder of the Pareto frontier unexplored. The NC method ensures that each point on the Pareto frontier has a corresponding barycentric coordinate tuple; however, the mechanism used to achieve this correspondence also leads some barycentric coordinate tuples not to map to any point on the Pareto frontier.

Two recent modifications to NBI by Motta et al. [112] and Mueller-Gritschneider et al. [113] resolve the lack of a one-to-one mapping by modifying the barycentric grid to ensure that the resulting sampling covers the entire Pareto frontier. Each method uses successive optimization of subfrontiers with increasing numbers of attributes, i.e. optimizing the individual attributes, then the 2-attribute subfrontiers, then the 3-attribute subfrontiers, etc., to bound the search space in increasingly higher dimensions. The methods differ in their approaches for distributing the weight vectors within these bounds. Motta et al. [112] use a centroidal Voroni tessellation design of experiments [42], while Mueller-Gritschneider et al. [113] solve for an even distribution using a linear programming formulation. The use of these approaches to form a coordinate system on the Pareto frontier is considered in Chapter 6.

2.1.2.2 Evolutionary Algorithms

A second class of multi-objective algorithms is the evolutionary algorithms. These algorithms work with a population of designs, which they seek to “evolve” into a set of non-dominated designs. The fitness (i.e. value) of the designs is typically calculated based on

two factors: the non-domination level, which is a measure of how many other designs in the population dominate a particular design, and some measure of diversity that drives the population to spread across the entire Pareto frontier.

These algorithms have the advantage of being easy to implement, and they are generally effective at converging to regions near the true Pareto frontier in a small number of iterations. However, they have difficulties in achieving precise convergence to efficient designs because they do not use derivative information. Example algorithms in this class include the niched Pareto genetic algorithm [62], the strength Pareto evolutionary algorithm [149] and the non-dominated sorting genetic algorithm-II (NSGA-II) [39]. The NSGA-II algorithm is one of the two approaches, along with a modified NBI, will be used in this work for generating sampled Pareto frontiers.

2.1.2.3 Continuation Methods for Multi-Objective Optimization

A final class of algorithms is the continuation or homotopy methods. These methods work by following a continuous path through the design space to find efficient designs, or to trace from one efficient design to others. Most such methods work only for two-attribute problems, for which any non-dominated point has well defined notions of the *previous* and *next* points on the Pareto frontier. Algorithms that follow a path of optimal designs have been developed by Rao and Papalambros [118] and by Rakowska et al. [117]. Algorithms that trace a path from dominated designs to efficient designs have been developed by Recchioni et al. [119, 107] Hillermeier has developed a novel homotopy algorithm that works with an arbitrary number of attributes [57].

2.2 Multivariate Data Visualization

Design analyses typically result in sets of numerical data, which may contain values for dozens of variables tabulated for any number of design alternatives. The best approach to making sense of these large amounts of data is often through visualization. But using visualization effectively is often difficult. Amar and Stasko [7] have identified two “gaps” that hinder visualization’s effectiveness in supporting decisions: the *worldview gap* between the data that need to be visualized to draw a conclusion and the data that are actually

visualized, and the *rationale gap* between the user’s perception of a relationship in the data and the user’s understanding of the cause and correctness of the relationship. Bridging these gaps requires that visualization for decision making be a process of *interactive knowledge discovery*: visualizations are not only for displaying the results of data-generating analyses, but ideally should facilitate interactive exploration of the data and should help the user assess the correctness and causes of perceived relationships.

This section briefly introduces some of the many multivariate visualization techniques that may be used for knowledge discovery in multi-attribute design problems.

2.2.1 The Visualization Process

Several researchers have undertaken the task of creating visualization process models, illustrated in Figure 9. Mackinlay [87], motivated by a desire to automate the visualization process, described a visualization process model comprising *extracting* data from a database, *synthesizing* the data into a graphical design, and *rendering* the graphic. In 1989 Upson [136] published a well known visualization process model shortly after the release of the landmark NSF report *Visualization in Scientific Computing* [98] that spurred great interest in computer-based visualization. Upson’s model comprised three processes: *filtering*, *modeling*, and *rendering*. A year later, Haber and McNabb [53] expanded Upson’s model by emphasizing the corresponding evolution of the data objects. A similar process model created by Chi [21] emphasizes the state of the data at each step of the process. Dos Santos and Brodlie [41] later updated the Haber-McNabb model to address explicitly the multivariate visualization problem.

Despite their differences, many elements and themes are common among the five process models. Each begins by preparing the *data*, obtained from some source, for visualization. For design problems, the data is typically produced by a computational analysis, and is subjected to some form of dimensionality reduction before being mapped to certain *graphical objects* or *elements*, which are then rendered to produce an image. Although not listed in the processes of Figure 9, *analytical tasks* also play a key role in motivating visualization styles and in extracting information from visualizations.

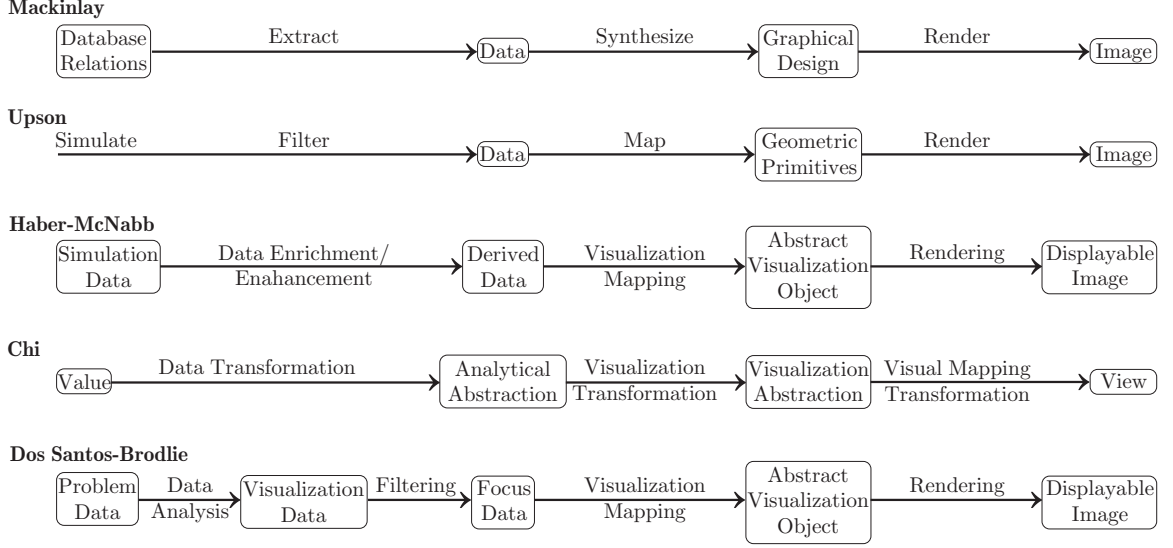


Figure 9: Comparison of visualization process taxonomies. Boxes indicate objects and arrows indicate functions. Similar concepts are vertically aligned.

The following sections consider the data and graphical elements of the process, as well as the analytical tasks invoked when viewing a visualization. The data source, in the form of design analysis computer codes, and dimensionality reduction are considered subsequently in the discussion of multivariate visualization theory.

2.2.2 Dimensionality Reduction for Visualization

As shown in the visualization processes in Figure 9, the transformation from data to graphical elements consists of two overarching functions: a preprocessing or filtering of the data produced by the design analysis, followed by a mapping to graphical elements. Any real design problem will likely involve more variables than can be trivially visualized, and dimensionality reduction will have to be used in order to visualize the data. Depending on the visualization techniques used, two or three variables can be mapped to spatial coordinate axes, and a few additional variables can be encoded by varying color, size, or other attributes of the plotted elements. Variables that are not mapped to such features are excluded from the visualization by either *projecting* or *slicing*, two dimensionality reduction techniques described below. Using this classification, one of the following actions is applied to each variable in the data set:

2.2.2.1 Plotting

Each variable that is to be meaningfully represented in the visualization must be plotted by encoding its information using a graphical feature. For graphics drawn in Cartesian axes, two variables must be plotted as the two coordinate axes, and additional variables may be represented by varying the color, size, or style of any marks drawn in the axes. Each variable that is not mapped to such a graphical feature is *unplotted* in the sense that the graphic does not explicitly show any information about how these variables vary. Although the effects of their variance may be seen indirectly through the plotted variables, the graphic does not contain enough information to establish a cause-and-effect relationship involving the unplotted variables.

2.2.2.2 Projecting

A variable can be discarded from consideration by orthogonally projecting the data set onto the lower-dimensional subspace that excludes this variable's dimension. In practice this is as simple as excluding that variable from the data set while creating a particular visualization. Crawford and Fall [30] explain that projection is a smoothing operation in that information about the projected dimensions is lost, but new information is never gained. As an example, consider the simple 2-D scatter plot on the right in Figure 10, which shows two response metrics as its coordinate variables. The scatter of the data is due to variations in the unplotted design variables, represented by the unlabeled depth axis on the left in Figure 10. This variability is unplotted in the 2-D scatter plot in the sense that it is impossible to determine which variations of the design variables cause the observed scatter in the plotted response metrics. Note that the inability to attribute the variability to a cause is a property of this particular visualization, not of the data.

2.2.2.3 Slicing

Rather than allowing an unplotted design variable to vary, it may be fixed at a particular value, reducing the dimensionality of the design problem by one. We call this operation *slicing*, using the terminology of [47] and [139], in the sense that the resulting visualization

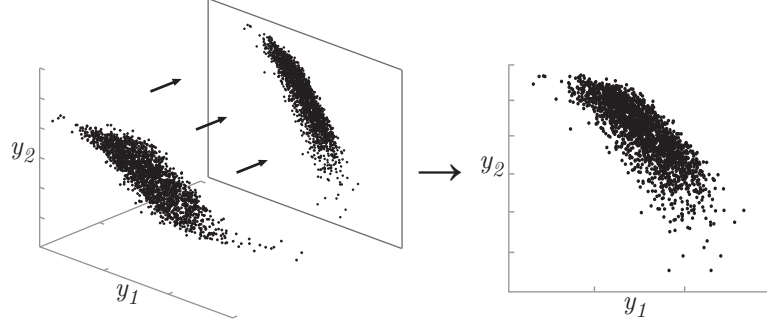


Figure 10: A 2-D graph (at right) created by projecting a multidimensional data set (at left) onto a plane. The appearance of the 2-D graph depends on the orientation of the axes plane, i.e. the choice of axis variables.

shows a single slice of the data space through the chosen value, instead of showing the entire space projected down onto a plane. Figure 11 shows a graph created by slicing, and illustrates the dependence of the resulting graph on the particular slice value chosen.

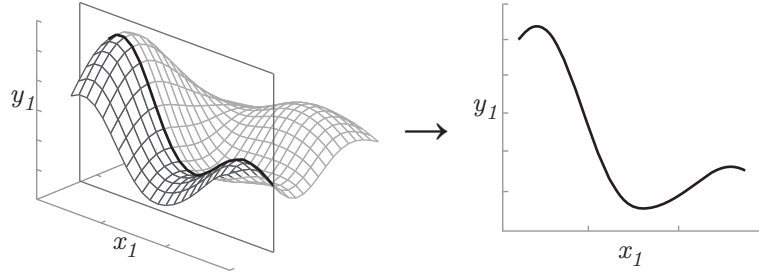


Figure 11: A 2-D graph (at right) created by slicing a continuous function (at left) at a fixed value of the unplotted variables, represented by the unlabeled axis. The appearance of the 2-D graph depends on the orientation of the plane defined by the choice of axis variables, and on the location of the plane as defined by the slice value of the unplotted variable.

A *current design* can be defined by slicing every independent variable to uniquely define a design, and ignoring the slice values when they are not needed because a particular independent variable has been plotted or projected. Because the dependent variables are functions of the independent variables, specifying slice values for all independent variables also determines corresponding slice values for the dependent variables. In practice, however, it is rare to create a graphic that simultaneously slices all design variables. Rather one or more independent variables will be mapped to a graphical feature and the dependent variables will retain some degrees of freedom. Consequently, it may also be useful to specify

slice values for the dependent variables independently of the current-design slices. For example, a contour plot shows variation in the independent variables at specified levels (slices) of a dependent variable.

Visualizations based on slicing are somewhat more difficult to construct than projections and can be computationally expensive for two reasons. First, all data that contributes to the visualization must lie in the visualization plane. Given an arbitrary data set, taking a slice at a given value will produce an empty graphic if no data happens to lie in the plane of the slice. For practicality, the data set must be tailored to the graphic by purposefully analyzing designs in the plane of the slice to populate the view. This requires not only the presence of and access to the design analysis functions or interpolating functions but also the analysis of many designs to create a single graphic.

Second, visualizations that rely on slicing are most useful when the slice values can be changed interactively by the viewer. The location of each slice is essentially a decision that locks in an independent variable at a particular value. Because there is much interest in understanding the effects of making different decisions, the most useful slicing visualizations are not static, but can be changed by controlling the location of the slices via slider bars, menus, or other controls. The need to rapidly evaluate new designs usually makes interactive slicing visualizations practical only when the design analysis has been replaced by fast surrogate models or when sufficient parallel computing capabilities are available so that the time required to run the design analysis scales weakly with the number of designs being analyzed.

2.2.3 Discrete and Continuous Visualization

The choice of either projection or slicing for dimensionality reduction correlates closely with the classification of information visualization techniques into *discrete* and *continuous* visualizations, as described by Tory and Möller [130, 131]. This classification is particularly relevant to understanding how different visualizations support different types of activities in a design process. The labels discrete and continuous refer to the visualization’s treatment of the underlying data source as comprising discrete points, or a continuous manifold of

data.

Discrete visualizations are drawn directly from an underlying data set, i.e. a discrete sampling of the objective or design space, without showing connectivity between the points in the data set. Dimensionality reduction for discrete visualizations is typically accomplished via projection. Discrete visualizations show a *global* view of the data, in the sense that they show many points simultaneously and in doing so allow the viewer to gain an understanding of the entire sampled space. This can be useful for distilling a very large data set into simple graphics that can reveal trends that would be difficult to detect otherwise. Scatter plots, histograms, and parallel coordinate plots are examples of discrete visualizations.

Continuous visualization, on the other hand, shows *local* variability around a single “current point” by showing a low-dimensional slice through the data space. Continuous visualizations typically show how one or two dependent variables vary in response to specified changes in one or two independent variables, while all other independent variables are held constant. For example, a line plot shows the change in one dependent variable caused by sweeping one independent variable through a specified range of values. Higher-dimensional continuous plots, such as contour plots, surface plots, and carpet plots, can be used to understand interactions between variables.

This classification of discrete and continuous design data visualization techniques is summarized in Table 1.

Table 1: Categorization of Multivariate Visualization Techniques

Class	Data Source	Dimensionality Reduction	Graphical Elements	Scope
Discrete	Data Set	Projection	Points, Glyphs	Global
Continuous	Design Analysis	Slice	Paths, Contours	Local







2.2.4 Basic Visualization Techniques

Once a data set has been prepared by applying any desired transformations or aggregations to the variables, and then projecting and slicing the data to reduce it to a set of variables to be plotted, the remaining variables must be plotted by mapping their values to graphical

marks. Most visualization map two or three variables to Cartesian axes, and then encode several more using points, lines, bars, colors, shapes, etc. Creating an effective visualization from these basic graphical elements is a matter of finding combinations of the elements that convey meaningful information while not interfering with each other visually.

Table 2 lists some commonly used visualizations, each of which is based on a corresponding graphical mark. It is perhaps unfortunate that these graph types are identified by the marks they use – line plot, scatter plot, pie chart, etc. – rather than by the underlying dimensionality reduction techniques each uses and the number and type of variables they plot.

Table 2: Simple Visualization Techniques

Graph Type		Elements Used	Dimension Reduction	Discrete/Continuous	Typical Axes Variables
Scatter		Point	Project	Discrete	Dep vs. Dep
Histogram/ Bar		Point	Project	Discrete	Dependent
Pie		Field	Project	Discrete	Dependent
Line		Path	Slice	Continuous	Ind vs. Dep
Contour		Contour	Slice	Continuous	Ind vs. Ind (Contours of Dep)
Carpet		Path	Slice	Continuous	Dependent (Contours of Ind vs. Ind)

The purpose of this section is to make the dimensionality reduction process more explicit and to emphasize the interdependencies among graphical elements, dimensionality reduction techniques, and the class (independent vs. dependent variable) of the coordinate axes variables.

2.2.4.1 Discrete Graphs

Among the simple discrete graphs, only the scatter plot uses marks that actually appear as points. There are many named variations of scatter plots that use particular transformations of the data, particularly in the field of statistical visualization; we will not discuss these specialized graphs here, but we emphasize that they share the common traits of emphasizing the discreteness of the data and ignoring (projecting) the unplotted variables. A review of scatter plots, including a discussion of interaction, is given by Huber [65].

The histogram is drawn using bars, but it should be apparent that each bar is simply a stylistic variation on a point placed at the top of the bar. Roth and Mattis [120] explain that bars are preferred when the showing *amounts*, whereas points are preferred when showing *coordinates*. When the x -axis variable is continuous², constructing a histogram requires the range of the variable to be divided into bins, introducing a new y -axis variable that counts the number of points in each bin. This transformation is used to avoid visual interference; plotting each point in a large data along a single continuous axis would result in significant overlapping, making it impossible to see the data distribution. Although binning reduces visual interference, it also reduces the information content of the graph as points within a bin become indistinguishable. The appearance, and consequently the interpretation, of a histogram can be strongly dependent on the choice of bin intervals. Histograms can be used to display statistical distributions such as discrete probability mass functions or continuous probability density functions. A related graph, the cumulative distribution plot, introduces a new y -axis variable that represents the number (or fraction) of data points with an x -axis value lower than the current x coordinate. This plot, drawn with points that are often connected to form a continuous curve, reduces the visual interferences while retaining all available information, but is less easily interpreted by non-specialists than the more familiar histogram.

²When the x -axis variable is discretely valued, the binning is trivial and the graph is usually called a bar chart.

2.2.4.2 *Continuous Graphs*

The line plot is undoubtedly the most commonly used continuous graph, and is perhaps the simplest of all graphs, typically consisting of only a single path element. These graphs form the backbone of the engineering visualization domain and have been widely used throughout the history of engineering design. Simple line plots are limited to a single degree of freedom, and as such the coordinate axes are almost always an independent variable on one axis (plotted on the horizontal axis by convention) and a dependent variable on the other axis. More complex line plots can be formed by plotting multiple paths in the axes, where each path shows the same coordinate variables, but differs in the value of a second (discretely valued) independent variable. In this case, each path exists in a different slice of the design space. The slices can be interpreted as being parallel to each other in the sense that the resulting graphic views each slice from the same direction, i.e. the coordinate axes are common among all slices. Another complex line plot can be drawn by overlaying a second axis system, whose vertical scale is typically drawn on the right side of the graph, and plotting two dependent variables vs. a single independent variable. In this case the two paths show the same slice, but viewed from a different direction. That is, the two paths share identical values of the unplotted independent variables, and differ only in which dependent variable is plotted on the vertical coordinate axis. This difference is illustrated in Figure 12. A recent example of two-axis line plots applied to design space exploration and decision making is given in [14].

The 3-D analogue of the line plot is the surface plot, in which the variation of one dependent variable as a function of two independent variables is plotted with a surface. The surface plot is conceptually similar to the aforementioned multi-slice line plot, but uses a single surface to show continuous changes in the second independent variable instead of using multiple paths to show discrete changes in the second independent variable.

Contour plots are conceptually similar to “bivariate line plots,” showing the change in a dependent variable as two design variables vary across a rectangular grid bounded by the coordinate axes. However, the visual strategy used to depict the relationship between the response metric and the design variables is different in a contour plot. Whereas the line

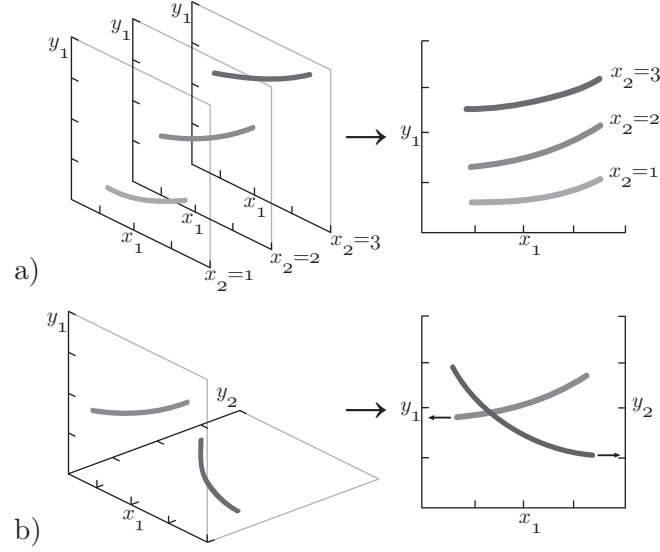


Figure 12: Two types of line plots. a) Three slices at different values of x_2 yield a one-vertical-axis plot. b) A single slice viewed from two different orientations yields a two-vertical-axis plot.

plot shows a continuous change in the response metric across the range of variability of the design variable, contour plots show only information at a finite number of values of the response metric, each represented by one contour.

Each contour may be interpreted as a slice in the response space, drawing the locus of designs that lies in a slice at a particular value of one response metric. As with the line plot, each unplotted design variable is also sliced at a particular value that remains constant among all contours in the plot. Because of the functional relationship between the design variables and response metrics, contours that show different values of a single response metric will never intersect; thus even a plot with many contours will not suffer from visual interference due to overlapping graphical elements. Larger numbers of contours allow the viewer to infer the continuous nature of the underlying design analysis more easily.

Each contour may be colored according to a continuous color map that encodes the corresponding value of the response metric. Like the dual-vertical-axis line plot, contours of multiple response metrics can be plotted in a single graph. In this case, a sparser contour density may be preferred, and, depending on the number of response metrics being plotted, it may be more desirable to draw all contours of a single response metric in the same color

and to distinguish values by including numerical labels on a subset of the contours.

A comparison of the three types of continuous graphs described above is shown in Figure 13. Each type of graph is shown for the same function of two variables. Note the differences in the roles that each variable plays in creating each type of graph.

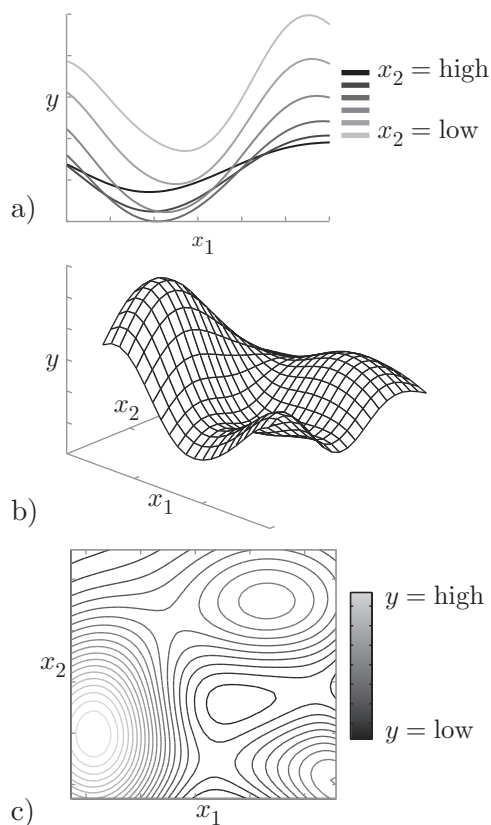


Figure 13: Three views of the same function. a) Line plot. b) Surface plot. c) Contour plot.

A contour plot can be “inverted” to create a carpet plot, which shows the value of the dependent variable on a single (typically vertical) axis. Consequently, the grid of the two independent variables is no longer orthogonal, but becomes distorted in order to draw uniform spacing of the dependent variable values. Carpet plots may be preferred to contour plots for some applications because they emphasize unit changes in the dependent variable rather than in the independent variables. However, if the functional relation between the plotted dependent variable and the plotted independent variables is non-monotonic, the grid of independent variable coordinates will fold over on itself, making the graph difficult

or impossible to interpret. Conversely, contour plots can always be drawn because the mapping from design variables to response metrics is surjective. In some cases it is possible to add an additional dependent variable axis and show how two dependent variables change as a function of two independent variables. Carpet plots have a long history in design engineering applications as exemplified by [68, 127, 140].






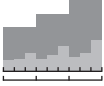

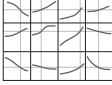

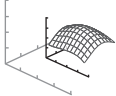
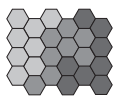
2.2.5 Multivariate Visualization Techniques

The basic graphs in the preceding section require the designer to reduce the data set to two or three variables before creating the graphs. The graphs considered in this section are inherently multivariate in the sense that they either scale to display an arbitrary number of variables or they are interactive, allowing the viewer to change the values of sliced design variables on the fly without recreating the graph from scratch. The properties of these graphs are summarized in Table 3. The subcategories used in the following sections are for organization only and are not intended to serve as a formal, mutually exclusive taxonomy.

2.2.5.1 3-D Scatter Plots and Scatter Plot Matrices

The modern era of multivariate visualization began in the 1970s when computers became sufficiently powerful to run many iterations of analyses with large numbers of independent variables and graphical interfaces became sophisticated enough to enable computer-assisted visualization. In 1974 Fisherkeller, Friedman, and Tukey [49] created the first interactive multidimensional visualization, a scatter plot named “PRIM-9” for “Picturing, Rotation, Isolation, and Masking – in up to 9 dimensions.” PRIM-9 featured a 3-D scatter plot that the user could rotate to view arbitrary projections, hide uninteresting or isolate interesting data points, and quickly change the axis variables to any of the nine variables that composed the input data set. PRIM-9 also implemented a “projection pursuit” algorithm that searched for potentially interesting projections using a numerical optimizer. Reviews of projection pursuit methods are given by Huber [64] and Crawford and Fall [30]. John and Paul Tukey’s continued interest in finding interesting projections of scatter plot data also led to the invention of “scagnostics,” a set of numerical metrics and statistical measures that describe interesting patterns in a given projection [142, 143]. A different approach to the

Table 3: Multivariate Visualization Techniques

Graph Type		Elements Used	Dimensionality Reduction	Discrete/Continuous	Typical Axes Variables
3-D Scatter Plot		Point	Project	Discrete	Dependent
Scatter Matrix		Point	Project	Discrete	Dependent and/or Independent
Parallel Coordinates		Point (Connected)	Project	Discrete	Dependent and/or Independent
Star/Radar		Point (Connected)	Project	Discrete	Dependent and/or Independent
Hyperbox		Point	Project	Discrete	Dependent and/or Independent
Hierarchical Axes		Point (Bar)	None	Discrete	All Independent and One Dependent
Glyph Plot		Glyph	None	Discrete	All Dependent, Independent optional
Prediction Profiler		Path	Slice	Continuous	Dependent vs. Independent
HyperSlice		Contour, Path	Slice	Continuous	Independent vs. Independent
Worlds Within Worlds		Surface	Slice	Continuous	All Independent and One Dependent
Self-Organizing Map		Glyph	Slice	Continuous	n/a

projection problem was taken by Asimov [10], whose “grand tour” displays an animated progression of 2-D projections of the data.

Tukey and Tukey were also among the first to use scatter plot matrices [135], which they originally called “draughtsman’s views” for their similarity to a 3-view drawing. Although they considered triangular or symmetric square matrix plots to provide an insufficient number of views of the data, scatter plot matrices using this format have become widespread

today. A scatter plot matrix is simply a tiling of scatter plots showing different projections of the data. Typically scatter plot matrices illustrate all orthogonal 2-D projections. Consequently, the techniques for interpreting and interacting with scatter plots work equally well for scatter plot matrices. Square scatter plot matrices have unused space along the diagonal, which may be used to label the scatter plots, or filled with a different type of plot. Hartigan first used this space to display histograms showing the univariate distributions of the data [55, 135]. Cui et al. [31] consider additional types of graphs to display along the diagonal of a scatter plot matrix, and explain how the graphs can be used in a brushing-enabled scatter plot matrix. Elmqvist et al. [46] also consider interactive scatter plots, focusing on interactive scatter plot matrix navigation. Inspired by Asimov’s grand tour, animated transitions are used to show the relationship between plots in the matrix.

Another issue that must be addressed when creating scatter plot matrices is point crowding. For large data sets, the elements in the data set may outnumber the number of pixels available in the computational graphical display, resulting in data loss as nearby data points redundantly color the same pixel. This issue is more problematic for on-screen data display than printed display, as printed resolution is invariably much greater than screen resolution (1200dpi for consumer-grade laser printers vs. 96dpi for LCD displays). Keim et al. [74] see this as a technical barrier to enabling new visualization techniques, noting that screen resolution has remained relatively constant despite order-of-magnitude increases in other aspects of computing technology.

Carr et al. [19] offer a solution to scatter plot crowding, suggesting that rather than plotting each point in the data set, the field of the axes is divided into an (invisible) grid of hexagonal bins, and within each bin a single marker sized according to the number of data points within that bin is drawn. In regions of sparse data, the binning scheme is abandoned and each data point is drawn at its actual coordinates. This system is still lossy in the sense that individual data points in crowded areas cannot be distinguished, however it does emphasize the density of data, whereas a standard scatter plot indicates only data coverage with no indication of density. A more modern variation of using uniformly sized (small) markers whose color corresponds to the data density may now be preferred.

2.2.5.2 *Parallel Coordinates*

Citing the problem of orthogonal axes taking up space quickly, Inselberg [66] developed the method of parallel coordinates in the 1980s, in which all plotted variables are mapped to parallel axes. In the typical construct, the parallel axes are drawn vertically, and the horizontal plot dimension has no meaning. Horizontal spacing is distributed evenly among axes to provide the separation needed for visual perception. Parallel coordinates graphs have the additional advantages of being expandable to any number of variables, giving all variables equal treatment, and having an obvious interpretation even for untrained viewers. Among the difficulties in using parallel coordinates is determining relations between non-adjacent variables, and visual crowding for large data sets. Fua et al. [51] created hierarchical parallel coordinates, in which similar designs are clustered and drawn as a thick band across the graph instead of as individual lines, reducing the graphical complexity for large data sets. Opacity varies vertically within each band according to the density of the data, and a single opaque line marks the mean of the designs that make up the band. Each band can also be decomposed into its constituent lines interactively by the user.

2.2.5.3 *Star Plots*

The *star plot* or radar plot is a polar version of the parallel coordinates graph that has gained wider recognition among non-specialists, but suffers from perceptual distortion due to the crowding of the axes at low values. However, star plots may be preferred for small data sets with few variables for their compactness. The first modern use of star plots was in the early 1970s, with each star acting as a glyph, and the individual stars being arranged in a grid [123, 48]. Color displays have since enabled multiple stars to be placed in a single axis system without interfering with each other. Recent implementations of star plots include the DataMeadow [45] multivariate visualization system, and knowCube [133], an interactive star plot in which the user can drag the vertices of the polygon to desired values and the tool selects the closest match in the database and displays it. An example application of star plots to a conceptual design problem is given in Reference [16].

2.2.5.4 *Hyperbox*

In the early 1990s Alpern [6] created *hyperbox*, which is conceptually similar to a plot matrix in that it uses orthogonal views of the data, but instead of arranging the axes in a matrix, each two-variable pair of axes becomes one face of the hyperbox. The resulting representation is more compact than a plot matrix and has better adjacency of related views, but some of the views are necessarily small or stretched in one or more dimensions. The reliance on non-orthogonal axes also limits the number of computer programs that support hyperboxes. As such, they have not been widely used.

2.2.5.5 *Hierarchical Axis Techniques*

Hierarchical axis visualizations are created by assigning multiple variables to one or more of the Cartesian axes so that any position along that axis defines values of more than one variable. In the late 1980s Mihalisin et al. created the first multivariate hierarchical axis technique, which they later patented [110] and named *multivariate visualistics* [67]. The method originally required the data set to comprise n independent variables sampled along a regular lattice and one dependent variable [108] but was later generalized for use with arbitrarily spaced data, provided the data has been binned uniformly [109]. In this case all variables in the data set are treated as independent and the dependent variable is the data count in each bin.

To construct the hierarchical axes, the independent variables are (arbitrarily) sorted from “fastest” to “slowest,” terms that do not indicate any physical property of the data but rather the order in which the variables are displayed. The axes are arranged so that variations in faster variables are grouped together at constant levels of the slower variables; thus different permutations of the ordering will yield different views. The resulting graphic takes the form of a histogram in which the horizontal axis displays the full-factorial combination of bins for all independent variables, sorted according to the assigned speeds. An example is shown in Figure 14, which shows two variables, y_1 and y_2 , sorted such that y_1 is faster than y_2 . The bins for y_1 are named a , b and c and the bins for y_2 are named d , e and f . The vertical axis shows the count of points in the data set in each combination of

bins. Dark bars show sum of spanned bin counts. Other variations of this technique are also described in the literature. For example a subset of the variables may be excluded from the axis hierarchy and treated as dependent variables. Instead of displaying histograms, boxes (similar to Tukey’s box plots [100]) showing the mean and standard deviation of the unbinned variables can be drawn on multiple vertical axes [109].

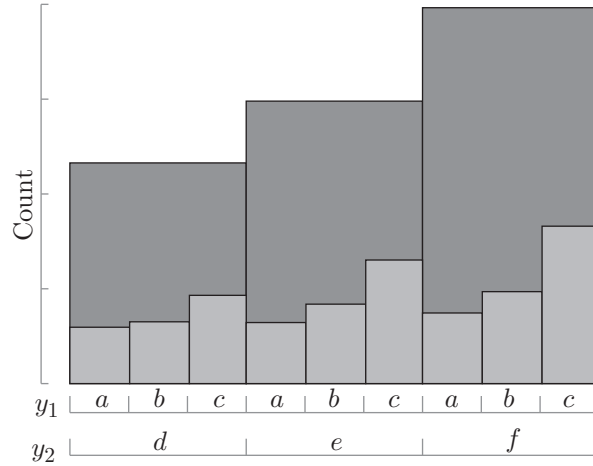


Figure 14: A notional multivariate visualistics plot

LeBlanc et al. [82] modified this technique by making the horizontal and vertical axes both hierarchical, a technique they call *dimensional stacking*. The independent variables are again ordered by “speed,” however the variables are first divided into two “orientations” that determine whether the variable will be assigned to the horizontal axis hierarchy or the vertical axis hierarchy. The resulting system of axes demarcates a 2-D grid of bins, and the interior of each bin is colored according to the value of some function evaluated at the corresponding bin values of the independent variables. An example is shown in Figure 15 for four independent variables, y_1 through y_4 . The range of each variable is divided into three bins. The brightness of each box indicates the average value of a dependent variable, y , among all points in the box. For cases where each bin contains at most one data point, as would be the case when the independent variables are sampled across a uniform lattice, the dependent function yields a single value that is mapped to a color. When a bin contains multiple points, the color might be determined by averaging the function’s value across all

points in the bin.

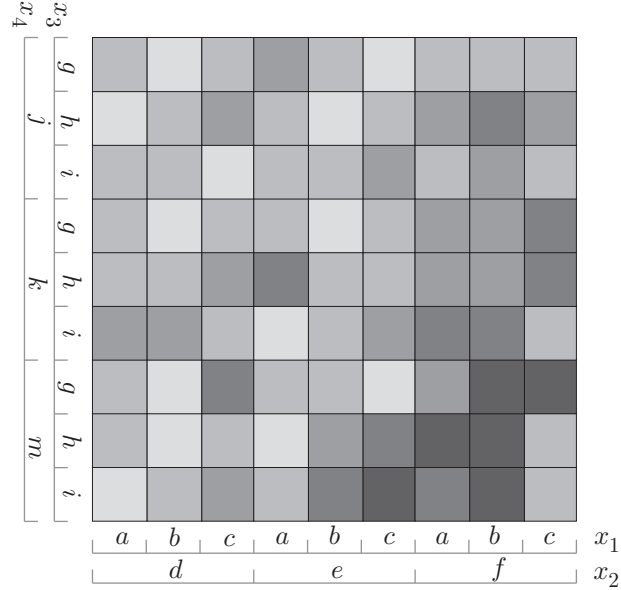


Figure 15: A notional dimensional stacking plot

In 1996, Becker et al. [13] created a different kind of hierarchical-axis visualization called *trellis display*. Rather than arranging all independent variables in a hierarchy, two variables are excluded from the hierarchy and used as the horizontal and vertical coordinates of a plot. A single such plot is associated with one bin in the hierarchy, with multiple plots arranged in a matrix representing all bins in the hierarchy. Above each plot, the hierarchy variables are listed in strips, with a portion of each strip shaded to indicate the range (bin) of the corresponding variable represented by that plot. An example single plot from a trellis display is shown in Figure 16.

The most significant difference between trellis display and the earlier hierarchical techniques is that each bin is represented by a full 2-D Cartesian axis system in which any variety of Cartesian graph can be drawn. For example, a scatter plot could be drawn that shows the variability of the data with respect to the two coordinate-axis variables, only showing those points that fall in the bin defined by the current ranges of the hierarchy variables. The resulting display is a sort of variation on a scatter plot matrix in which all plots share the same coordinate axis variables and each data point only appears in a

single plot. To allow easy comparison between plots in the matrix, all would share the same axis scales as defined by the minimum and maximum values across all plots in the matrix. Another notable differences between trellis display and the earlier hierarchical techniques is that trellis display explicitly allows for non-uniform bin widths, for example defining bins so that the common property is the number of contained data points rather than the bin width. This is not feasible for the earlier techniques, for which the bin widths are linked to the size of the graphical elements.

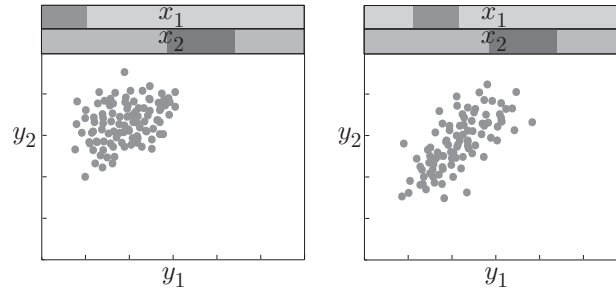


Figure 16: Two notional plots from a trellis display. The shaded bands at the top indicate the relative range of x_1 and x_2 plotted in this graph.

Tufte’s definition of *small multiples*[134] is closely related to trellis display, but less strictly defines the relation between the individual plots. Instead, small multiples require only that the arrangement and similarity of the plots makes the intended comparison obvious to the viewer. As an example, small multiples are used by Mullur and Messac[114] to compare the performance of different types of metamodels against different sample functions.

Another hierarchical axis technique, *worlds within worlds*, is described below in the section on continuous multivariate visualization.

2.2.5.6 Glyph-Based Techniques

Other researchers have taken a different approach to displaying multivariate data that does not rely on increasing the number of axes or producing axis groupings. These methods use more complex point-like marks called *glyphs* while maintaining simplicity in the coordinate axes, or even doing away with coordinates entirely. Glyph plots were first used in 1957

by Anderson [8], whose glyphs consisted of a central circle labeled with an identifier, and variable-length bars emanating from different locations on the circle to encode variables. Anderson demonstrated use of these glyphs as scatter plot markers in a coordinate system and arranged in a grid with no coordinate system. Different glyph-based techniques use different graphical elements to encode the variables, but the general strategy of encoding many variables using some parametrically defined object is common to all techniques. For example, Chernoff [20] created a well known type of glyph based on line drawings of faces, reasoning that people would be more easily able to interpret differences in faces than in abstract glyphs. A sample of different glyph types is shown in Figure 17.

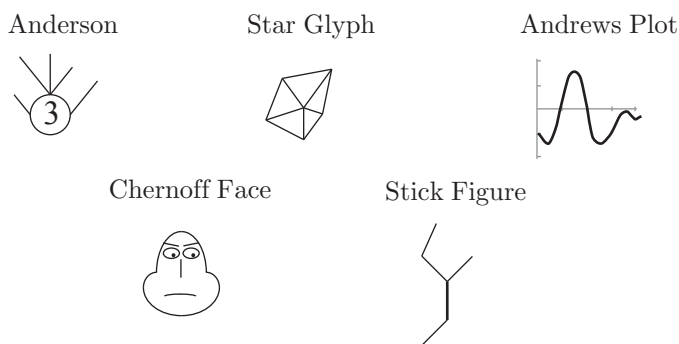


Figure 17: Example glyph styles.

Because of their complexity and the large number of relationships shown, glyph plots are best used for identifying outliers and observing trends across large subsets of the data rather than for detailed study of particular designs. Some glyph plots are specifically tailored to this purpose. For example, *Andrews plots* [9] are created by using the data values as coefficients in a single-parameter periodic function, a transformation that visually isolates outliers in the data set while almost completely obfuscating the actual numerical values of the plotted variables.

In practice glyphs are difficult to implement because of their limited availability in commercial tools and the considerable training required to learn to interpret them. More commonly, commercial tools allow creation of glyph-like plots that are based on scatter plots, using simple geometric shapes as markers and using variable shape, color, size, and

border appearance to encode additional variables. The difficulty in utilizing such plots is the need to compare disparate visual elements simultaneously in order to understand relationships between the variables. Just as the different elementary perceptual tasks favor different types of data, the different parametric elements of glyphs are perceived differently. Care must be taken to map the variables to the glyph parameters in a way that maximizes the readability of the resulting graphic [116].

2.2.5.7 *Prediction Profiler*

The techniques described thus far are all discrete visualizations; continuous multivariate visualizations have also been created by pairing line and contour plots with controls that let the viewer interactively control the slice value for each unplotted variable. Line plots are often arranged in rectangular matrix, commonly called a *prediction profiler*, that shows all combinations of design variables and response metrics. A notional prediction profiler is shown in Figure 18. The prediction profiler is notable for being interactive while not requiring external controls. Each independent variable appears as a coordinate axis in the matrix, and markers are drawn on each axis to show the current slice values and also to act as slider-type controls. Although the current-value marker of a particular variable is shown on all graphs for which that variable is mapped to a coordinate axis, the slice value affects only *other* graphs in the matrix, for it is those graphs that require a fixed slice value of this variable. Recall that the inclusion of a particular variable as a coordinate axis variable negates the need to slice or project that variable, thus the slice values are simply ignored in the graphs for which they are not needed.

The use of prediction profilers in design studies has been extensively studied by Mavris et al. [95, 76, 43] and linked prediction profilers forming a “unified tradeoff environment” have been explored [11].

2.2.5.8 *Hyper Slice*

The first implementation of contours in an interactive plot matrix to show slices of the design space was published by van Wijk and van Liere [139, 138] in 1993 with the name *HyperSlice*. A notional HyperSlice visualization is shown in Figure 19. Each off-diagonal plot shows a

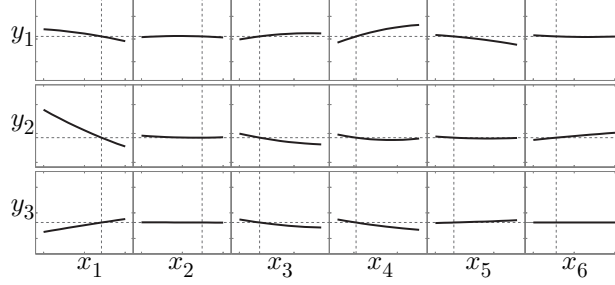


Figure 18: Prediction profiler showing six independent variables and three dependent variables. Horizontal and vertical lines show current slice values.

contour plot of the same dependent variable as a function of the two independent variables listed in the corresponding row/column of the matrix. The current design point is indicated by the circle at the center of each plot, and each plot can be clicked and a new point dragged to the center, becoming the new current design. In addition to interactive dragging, the viewer may define a path through the n -dimensional design space and watch an animation of the contour plots as the current design is moved along that path. Such a path is overlaid on the contour plots in Figure 19 beginning at the current design and passing through two more points marked with crosses. The plots along the diagonal in the HyperSlice use the same horizontal axis as the other plots in their column, but the vertical axis is mapped to the *dependent* variable. Each plot on the diagonal contains a single path that shows the univariate sensitivity of the dependent variable on the corresponding independent variable at the current design point, similar to the plots in the prediction profiler.

Despite the large amount of information it displays, HyperSlice has not been widely adopted, perhaps because of the computational expense associated with updating each plot in the matrix when the current design is changed. Non-matrix variations of this technique that show a single interactive contour plot with unfilled sets of contours representing multiple dependent variables are more commonly implemented. HyperSlice has also been adapted to multiresolution data analysis by Wong et al. [147], who used wavelet transforms to vary the data resolution and projection with different norms to vary the display resolution. One triangular half of the square matrix was also modified to display the error inherent in low-resolution views.

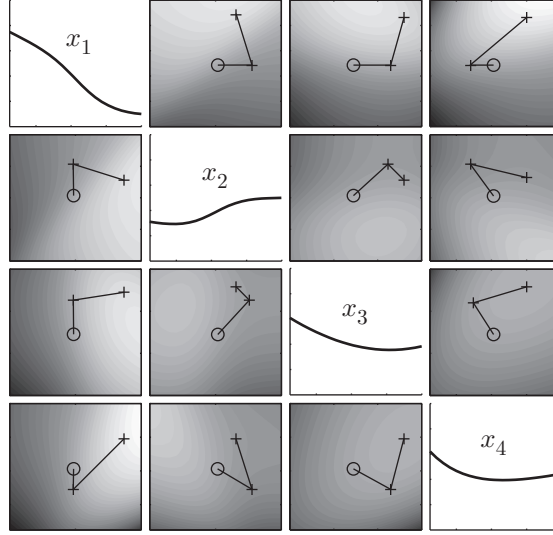


Figure 19: Notional HyperSlice showing four independent variables.

2.2.5.9 World within Worlds

Around the same time as Mihalisin and LeBlanc created their discrete hierarchical axis techniques, Feiner and Beshner [47] created a continuous hierarchical-axis visualization technique called *worlds within worlds*. This technique does not require binning of variables; rather each independent variable is allowed to vary continuously. The independent variables are each assigned to one axis of a 3-D coordinate system, with enough coordinate systems to accommodate all independent variables plus one dependent variable. The coordinate system that contains the dependent variable is the lowest-level coordinate system, and its origin is a point (in the graphical element sense) located in a higher-level coordinate system. The origin of that coordinate system in turn represents coordinates in a still higher-level space, and so on until all independent variables have been accounted for. The origin in this context refers to the location where the three coordinate axes intersect as drawn on screen, not necessarily the zero-point of the coordinate system. The lowest-level coordinate system is the only one to contain any graphical elements, namely a surface plot showing the variability of the dependent variable as a function of the two independent coordinate variables. The higher-level coordinate systems serve only to provide a visual, interactive means of slicing

the remaining independent variables. The user can drag each coordinate system’s origin within the next-highest coordinate system to change the slice values of up three variables simultaneously.

Feiner and Beshner’s main focus is on virtual reality interaction, and as such they only describe 3-D coordinate systems; however, this technique works equally well with 2-D coordinate systems by replacing the lowest-level surface plot with a line plot. A notional worlds within worlds system is shown in Fig. 20.

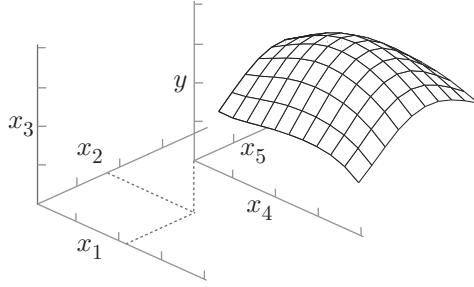


Figure 20: Notional worlds-within-worlds visualization.

A variant of this technique that is particularly relevant to some adaptive sampling and optimization methods involves drawing a local probability density function at one or more points in a graph. The probability-density axis forms a local coordinate system within the larger graph. Examples can be found in Keane [72] and Forrester et al. [50].

2.2.5.10 Self-Organizing Maps

Self-organizing maps (SOMs), a type of artificial neural network invented by Kohonen in the 1980s [77], offer a novel technique for creating 2-D visualizations from multidimensional data that bear little resemblance to any of the visualization techniques considered thus far. Whereas projection pursuit and principal component analysis are limited to finding interesting *planes* in the data space, SOMs can find arbitrarily shaped 2-D manifolds. A 2-D grid of nodes, usually rectangular or hexagonal, is overlaid on the design space and each point in the data set “trains” these nodes by attracting the nearest node, which in turn exerts a weaker pull on the nodes to which it is connected. In this way, the node grid is distorted as different parts of it are attracted to different regions of the n -D data space.

Importantly, however, the grid itself remains two-dimensional and maintains adjacency of neighboring nodes. Thus SOMs always yield a 2-D map of the n -D data space.

Corresponding visualizations are created by drawing the node grid as if it were undistorted, but coloring each node according to its trained coordinates with respect to data. The nodes in the grid therefore do not correspond to points in the data set, but to locations in the data space [78, 94]. The resulting visualization lacks meaningful horizontal or vertical coordinates; rather meaning is derived by observing the color patterns of nearby nodes. Example studies using self-organizing maps for design are presented in References [69, 22, 115].

In Chapter 7, a coordinate system based on self-organizing maps is introduced. This coordinate system is based on the SOM's mapping capabilities and does not directly make use of the corresponding visualization capabilities described in the previous paragraph.

A similar technique to SOMs is *generative topographic mapping*, which uses a probabilistic approach to introduce noise to the mapping to account for the loss of information in going from the high-dimensional data space to the low-dimensional map [17]. An application of generative topographic mapping, SOMs and hierarchical axes to an aircraft design problem was published by Holden and Keane in 2004 [59].

2.2.6 Visualization Techniques Developed and Implemented by the Design Community

While the visualizations described in the preceding section were developed for general multivariate data visualization, the engineering design community has also been active in creating new visualization techniques and tools tailored to visualizing design data. In this section recent design space visualization techniques created and implemented by the design community are surveyed.

2.2.6.1 Visualization-Enabled Design Processes

In 2002, Winer and Bloebaum published a series of papers [145, 146, 144] describing *visual design steering*, defined as using visualization to steer the solution of some computational analysis to achieve certain desired results. Visualization can be used while an analysis is

running, or between successive runs as would be the case for steering an iterative optimizer. The first implementation of visual design steering was called *graph morphing*, a type of interactive visualization in which two or three design variables are plotted on Cartesian axes showing contours or isosurfaces of response metrics while the unplotted independent variables are controlled via slider bars.

A second visual design steering technique called *cloud visualization*, a type of interactive scatter plot for conducting user-in-the-loop optimization, was developed by Eddy and Lewis [44]. Cloud visualization uses linked 2-D or 3-D scatter plot views of the design space and response space, with points colored according to preference, Pareto optimality, or feasibility. Cutting planes can be used to exclude uninteresting designs, similarly to the masking feature in PRIM-9 [49]. When more than three response variables must be considered, a worlds-within-worlds [47] technique can be used, whereby the values of the currently displayed axis variables may be locked (sliced) at the values corresponding to a user-selected design in the scatter plot. A new “child” scatter plot is then created allowing the remaining, previously unplotted, variables to vary while enforcing all specified slices [44]. Cloud visualization derives its name from the idea that a single point in the initial scatter plot is actually representative of the entire cloud of points in the child scatter plot.

Visual design steering was also applied in *Brick Viz*, a technique developed by Kanukolanu et al. [71] intended to deal with design problems involving coupled subsystems with uncertainty. In this technique, a fixed design point in one subsystem is examined while independent variables of the other subsystems are modeled as random variables and sampled using a Monte Carlo simulation. The results of the Monte Carlo simulation are plotted in a 3-D scatter plot whose axis variables are chosen from the inputs to the other subsystems. Rather than plot every Monte Carlo point, all points belonging to the same simulation are grouped and represented with a “brick” (i.e. a 3-D field) whose color indicates the fraction of the composing points that are feasible.

In 2006, Hongman et al. published the Interactive Design Selection Process with the (iterative) steps: 1) Select design parameters, 2) Generate initial exploratory designs, 3) Screen design parameters, 4) Generate Pareto set, 5) Select preferred designs, 6) Refine

preferred designs [61]. This process is interesting because it explicitly calls for the designer to choose from the Pareto set. The authors recommend the use of visualization to aid the definition of preferences in step 5, but do not present a specific generalizable process for how visualization may be used.

2.2.6.2 Visualizing the Pareto Frontier

In 2004 Agrawal et al. [2] introduced *hyperspace diagonal counting* (HSDC), a dimensionality reduction technique similar to LeBlanc’s [82] dimensional stacking. Dimensional stacking groups the *independent* variables into two subsets, which are then binned *hierarchically*, while HSDC groups the *attributes* into two subsets, which are then binned by *diagonal counting*, as illustrated in Figure 21.

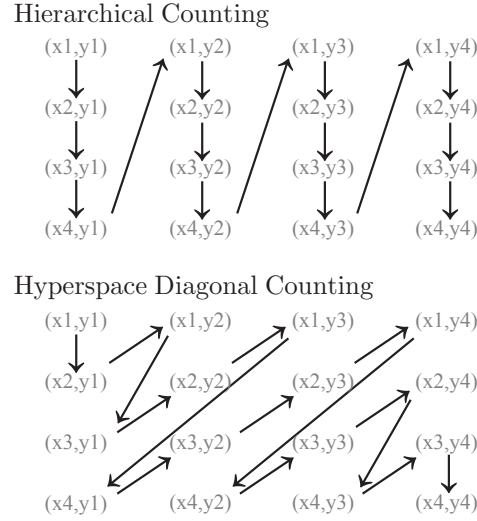


Figure 21: Comparison of bin ordering for two variables, x and y , using hierarchical counting and hyperspace diagonal counting. Each variable’s range has been divided into four numbered bins. Based on [5].

The effect of diagonal counting is that bins closer to the origin in the resulting graph contain designs with more favorable attribute values than bins farther away from the origin [1]. This desirable trait is not shared by the hierarchical axis techniques; however the tradeoff is that the actual values of the individual attributes can no longer be compactly shown directly on the axes of the plot. Instead only the bin indices are shown and the user must cross reference the indices with actual values by using, for example, a table.

After the binning scheme has been established, only the non-dominated data points are plotted in a 3-D histogram in which the x and y axes are as described above and the z axis shows the count of data points in each bin. By assigning preference weightings to each attribute, each bin can be assigned an overall desirability level, and the corresponding bar in the histogram can be colored according to its desirability [1]. Alternatively, desirability can be defined by comparing the non-dominated points to a hypothetical candidate design point [3]. A technique for coloring the bars according to robustness is also described [5]. Lastly, a variation of HSDC for finding and approximate Pareto frontier is described [4]. This technique has more in common with the hierarchical axis methods, as it uniformly samples the design space and bins the independent variables instead of the attributes, with a single attribute plotted on the z axis and points colored according to feasibility or dominance.

In 2008 Chiu and Bloebaum [23] began a series of papers that describes another visualization technique for the non-dominated points in a data set. Like HSDC, the new technique, named *hyper-radial visualization* (HRV), reduces the high-dimensional objective space to two dimensions for visualization. However, rather than using a binning scheme, the response metrics are normalized and divided into two groups, and the normalized attribute values within each group are simply summed to obtain two aggregate response metrics. These two aggregate metrics become the axis variables of a scatter/contour plot where the scatter points show the known non-dominated designs and the contours show levels of constant radius from the origin in the aggregate objective space. The points closest to the origin are identified as being the “best” designs. The technique is also demonstrated with different preferences (weightings) on each attribute [24, 25] and in a “design-by-shopping” process to generate additional designs in favorable regions of the design space [26].

HRV is notable for its combination of discrete and continuous visualization within a single graph – the combined scatter/contour plot. However, the significance of the particular choice of axis-variables is unclear. The decision-making elements of HRV that rely on spatial coordinates are based on the radial contours rather than the axis coordinates. Unless a particular grouping of variables leads to two inherently meaningful aggregate variables to serve as coordinate axes, the process could be just as effective using a single-axis plot of the

hyper-radius or a two-axis plot of hyper-radius and an additional decision metric.

2.2.7 Visualization-Enabled Decision Support Software

Several commercial and open source programs to aid design decision making are now available, including modeFRONTIER (Esteco) [27], Model Center (Phoenix Integration) [75], VisualDOC (Vanderplaats Research and Development) [129], DAKOTA from Sandia National Labs, and OpenMDAO from NASA [111]. In addition to providing user-friendly interfaces for linking multiple design analysis codes, these frameworks also provide visualization tools for viewing the data that is generated by these analyses. Some decision support tools do not provide the capability to link and automate design analysis codes but are instead used to interact with and visualize tabular data sets that were obtained from a separate design analysis code. Two examples are the ARL Trade Space Visualizer (Penn State), which incorporates many visualization-enabled design techniques [124] and JMP (SAS), a statistical analysis tool that also incorporates many visualization and surrogate modeling techniques.

The author has also developed an open-source decision support tool named Rave [34, 36]. Rave implements over 20 types of visualizations as well as surrogate modeling and optimization capabilities. Notably, Rave includes both discrete and continuous visualizations and allows them to be used side by side. Rave was created to provide researchers with an easily extensible framework for testing and distributing new decision support techniques and to provide users with point-and-click access to many decision support capabilities. Three key characteristics of Rave enable these capabilities: 1) A novel “workspace” system that lets users create persistent arrangements of visualizations. 2) Data structures that are tailored to design problems by incorporating analysis functions, constraints, and relevant metadata such as the target values and preferences for each variable. The data structures also incorporate a mechanism, called Analyses, for sharing information between different decision support techniques. 3) A “plug-in” system for adding new capabilities that largely avoids the need for researchers to modify Rave’s existing source code.

A survey of the key capabilities incorporated in these each of these tools is presented

in Table 4. The capabilities listed in the rows of Table 4 are grouped into five high-level categories of decision support techniques that represent the general scope of the capabilities provided by this type of software: design analysis linking and automation, visualization, optimization, surrogate modeling, and probabilistic methods.

Table 4: Capabilities and features of some design support tools

	ATSV	DAKOTA	JMP	modeFRONTIER	Model Center	OpenMDAO	VisualDOC	Rave
Analysis Function Linking		●		●	●	●	●	○
Discrete Visualization	●		●	●	●		●	●
Continuous Visualization	○		●	○	○		○	●
CAD Model Visualization					●			
Linked Visualizations	○		●	●	●			●
Interactive Visualization	●		●	●	○		○	●
Persistent Arrangement of Plots			●	○				●
Single-Objective Optimization		●	○	●	●	●	●	●
Multi-Objective Optimization	○	●		●			●	●
Constraints/Feasibility Assessment	●			●	●		●	●
MCDM/Interactive Preferences	●			●	○			●
Surrogate Modeling		○	●	●	○	○	○	●
Design of Experiments		●	●	●		●	●	○
Statistical Analysis			●					
Probabilistic Design		●	●	●	●		○	
● = Primary Capabilities ○ = Secondary Capabilities								

Despite the availability of these decision support tools, the scientific and engineering research communities have expressed a need for tools that better accommodate human decision making processes, especially by providing improved interaction between the different categories of decision support capabilities. A 2006 NIH/NSF report cites a need for tighter integration of information visualization and other types of data analysis tools [70]. This type of integration is the goal of *visual analytics*, an emerging multidisciplinary field focused on augmenting analytical reasoning with interactive visualization in order to allow decision makers to better understand information and communicate findings [28]. Ward et al. contend that no software system has yet provided this sort of seamless integration of interactive

visualization techniques with computational analysis techniques, and that this integration is an important direction for future research [141]. In 2008, Hayes and Akhavi examined the use of design decision support tools and found that design tasks have a greater need for flexibility than other types of decision problems due to their poor definition and exploratory nature [56]. They found that existing tools do not adequately support designers' need to alternate between information seeking, comparison, and selection tasks. Similarly, participants in a 2010 NSF workshop on multidisciplinary design optimization expressed a need for tools and techniques that support human decision making and design space exploration, as well as for tools that can be used by experts and non-experts alike [122].

These reports attest to the continuing evolution of the needs of the design optimization community. In the past, the community's greatest needs were related to automating the exchange of data between design analyses. This would allow a greater number of analyses (i.e. disciplines) to contribute to the design problem, which in turn would allow more realistic design problems to be studied earlier in the design process. In response to this need, several analysis integration frameworks were developed that aid the user in linking analysis codes, as indicated in Table 4, and this capability is now becoming a commodity. The needs expressed in the previous paragraph reflect a desire for new types of linkages — not between analysis codes, but between the decision support methods that are used to explore the analysis codes. This seems to be motivated by an increasing realization that in the absence of an unequivocal value function, design decisions are often made by examining the results of trade studies, optimizations, and other computer experiments.

2.3 Multi-Criteria Decision Making

Belton and Stewart define multi-criteria decision making (MCDM) as “a collection of formal approaches which seek to take explicit account of multiple criteria in helping individuals or groups explore decisions that matter” and state that “the aim of (MCDM) should be... to facilitate decision makers' learning about and understanding of the problem faced, about their own, other parties', and organizational priorities, values, and objectives and through exploring these in the context of the problem to guide them in identifying a preferred course

of action” [15].

To more clearly identify categories of methods, some authors divide MCDM into two sub-fields: *multi-attribute* decision making (MADM), which applies to discrete decision spaces (i.e. decisions with a finite number of alternatives), and *multi-objective* decision making (MODM), which applies to continuous decision spaces [132]. However, this distinction is blurred by techniques that are applicable in either case, and is further blurred by its apparent inclusion of multi-objective optimization as a technique in multi-objective decision making. To make a clear distinction between multi-objective optimization and MCDM, we here consider the methods of generating Pareto-efficient alternatives to fall under the category of multi-objective optimization, and the methods of ranking alternatives to fall under MCDM. We further divide MCDM into two categories: those methods applicable only to small numbers of alternatives, and those applicable to arbitrary numbers of alternatives. These categories are similarly, but not exactly, aligned with MADM and MODM respectively.

The first category, MCDM methods that are applicable only to a small number of alternatives, includes methods that require the decision maker to spend a significant amount of time considering each alternative; therefore these methods scale poorly to large numbers of alternatives. These methods typically compare the alternatives without using a closed-form value function and generally emphasize qualitative over quantitative information. Consequently, these methods are more useful early in the design process for conceptualizing and synthesizing. Examples of such methods are QFD, ELECTRE, and Pugh matrices.

The second category, MCDM methods that are applicable to arbitrarily large numbers of alternatives, includes methods that work equally well for discrete or continuous decision spaces, with few or many alternatives. These methods generally aid the decision maker in developing a closed-form value function of the attributes, which can then be used to rank arbitrarily many alternatives. These methods typically depend on quantitative information and are more useful for parameterized design problems. Examples of such methods are TOPSIS and the weighted sum method.

This second category of methods is the more relevant to this dissertation. The sections that follow describe the main points, benefits, and detriments of this class of MCDM methods whose goal is to help the decision maker choose a value function.

2.3.1 Value-Function Based MCDM Methods

Value theory can be informally defined as the specialization of utility theory to cases of certainty. Many authors (e.g. Luce and Raiffa [86], and Keeney and Raiffa [73]) do not explicitly consider this a separate theory from utility theory, but simply a special case. Utility theory, and consequently value theory, is based on several axioms, which differ slightly across its various formulations [86, 128]. Two axioms, however, are common to all implementations and are relevant to all MCDM methods. These are *completeness* and *transitivity*, which are defined below. In these axioms, $y_1 \dots y_n$ are different alternatives in a decision, the symbol \succ is the binary relation “is preferred to,” and \asymp is the relation “is indifferent to.”

Definition (Completeness Axiom). *Either $y_1 \succ y_2$ or $y_2 \succ y_1$ or $y_1 \asymp y_2$.*

Definition (Transitivity Axiom). *If $y_1 \succ y_2$ and $y_2 \succ y_3$ then $y_1 \succ y_3$.*

These axioms serve to establish an ordering on the set of possible outcomes, and we will call this ordering the *preference structure* (or simply the *preference*) for the outcomes. A function $v : \mathbb{R}^k \rightarrow \mathbb{R}$ such that $y_1 \succ y_2 \succ \dots \succ y_m \iff v(y_1) > v(y_2) > \dots > v(y_m)$ is called a *value function*. It can be seen that a value function uniquely determines a preference structure, but a preference structure does not uniquely determine a value function. Two value functions that determine the same preference structure are said to be strategically equivalent [73].

An important point must be emphasized about value functions: The numerical values, $v(x_1), \dots, v(x_m)$ are meaningful only to the extent that they define an ordering of the outcomes. No other relations are meaningfully defined for the value—in particular, differences and ratios of values are meaningless. In other words, value functions can be used to determine that one outcome is preferable to another, but they cannot be used to quantify

the *extent* of this preference. Although it may be possible to construct a particular value function that defines other relations, MCDM methods generally do not seek to do so. Consequently, if the decision maker already knows his preference for the outcomes, then defining a value function serves no additional purpose. The motivation for using a value function is to help the decision maker understand how the (subjective) relative importance he ascribes to the various conflicting objectives can be aggregated into a ranking of the outcomes.

It is additionally worth noting that nothing in the axioms of value theory requires or even suggests a particular form of a value function, nor is there anything that suggests that there should be a universal form applicable to many types of problems. Nevertheless, many MCDM methods seek to form such a function, often with a surprisingly simple/basic functional form. These methods work by first defining a set of conditions that could underlie a decision makers preference structure. If the decision maker agrees to the particular method's conditions, then it provides a value function form that also agrees with the stated conditions. The most commonly invoked condition regarding the attributes is *preferential independence*:

Definition (Preferential Independence). *Given k attributes, $y_1 \dots y_k$, let $\hat{\mathbf{y}}$ represent any subset of these attributes and $\tilde{\mathbf{y}}$ the complementary subset. Let the accents ' and '' denote particular values of the attributes. Then the set of attributes $\hat{\mathbf{y}}$ is preferentially independent of the set of attributes $\tilde{\mathbf{y}}$ if, for some $\tilde{\mathbf{y}}''$, $(\hat{\mathbf{y}}', \tilde{\mathbf{y}}'') \succ (\hat{\mathbf{y}}'', \tilde{\mathbf{y}}'')$ implies $(\hat{\mathbf{y}}', \tilde{\mathbf{y}}') \succ (\hat{\mathbf{y}}'', \tilde{\mathbf{y}}')$ for all $\hat{\mathbf{y}}', \hat{\mathbf{y}}'', \tilde{\mathbf{y}}'$ [73].*

Keeney and Raiffa show two important results related to preferential independence. First, if all pairs of attributes are preferentially independent of their complementary subsets of attributes (e.g if y_1, y_2 is preferentially independent of $y_3 \dots, y_k$), then all the attributes are mutually preferentially independent. Second, if all attributes are mutually preferentially independent, then there exists a value function of the form [73]

$$v(y_1, y_2, \dots, y_k) = \sum_{i=1}^k v_i(y_i) \quad (2)$$

Because taking the logarithm of a value function does not change the resulting preference

structure, the above result also establishes the existence of a value function of the form

$$v(y_1, y_2, \dots, y_k) = \prod_{i=1}^k v_i(y_i) \quad (3)$$

2.3.2 Proposed Value Function Forms

The results at the end of the previous section establish the existence of an additive or multiplicative value function for the case of preferential independence of the attributes. However, it does not prescribe a particular form for the value function. Table 5 lists several forms of commonly used value functions that were considered in a 2000 review by Messac et al. [105]. Each of these functions uses either the additive or multiplicative form, and so each requires the preferential independence condition. In this table, G is a the ideal outcome and B is the worst, also called the negative ideal, outcome.

Table 5: Examples of Value Functions used in MCDM Methods

Name	Value Function
Absolute Value Method	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \sum_{i=1}^m \mathbf{w}_i y_i(\mathbf{x}) - G_i $
Weighted Square Sum	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \sum_{i=1}^m \mathbf{w}_i (y_i(\mathbf{x}) - G_i)^2$
Weighted Maximum	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \max_i \frac{y_i(\mathbf{x}) - G_i}{B_i - G_i}$
Substitute Objective Function	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \prod_i \frac{y_{i,max}(\mathbf{x}) - y_i(\mathbf{x})}{y_{i,max}(\mathbf{x}) - y_{i,min}(\mathbf{x})}$
Kresselmeir-Steinhauser Function	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \frac{1}{\rho} \ln \sum_{i=1}^m \exp(\rho y_i(\mathbf{x}) - y_{i,max})$
Distance from Utopia Point	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \sum_{i=1}^m \mathbf{w}_i (y_i(\mathbf{x}) - y_{i,min})^2$
Distance from Utopia Point	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \sum_{i=1}^m \mathbf{w}_i (y_i(\mathbf{x}) - y_{i,min})^2$
Exponential Weighted Method	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \sum_{i=1}^m \mathbf{w}_i \exp(c_i y_i(\mathbf{x}))$
Weighted Compromise Programming	$\min_{\mathbf{x} \in D} V(\mathbf{x}) = \sum_{i=1}^m \mathbf{w}_i (y_i(\mathbf{x}))^{c_i}$

As indicated by the functions in Table 5, the most common form for a value function is based on a *metric*, which is a function d of two points³ \mathbf{y}_1 and \mathbf{y}_2 with the following properties:

1. $d(\mathbf{y}_1, \mathbf{y}_2) \geq 0$
2. $d(\mathbf{y}_1, \mathbf{y}_2) = 0 \iff \mathbf{y}_1 = \mathbf{y}_2$
3. $d(\mathbf{y}_1, \mathbf{y}_2) = d(\mathbf{y}_2, \mathbf{y}_1)$
4. $d(\mathbf{y}_1, \mathbf{y}_3) \leq d(\mathbf{y}_1, \mathbf{y}_2) + d(\mathbf{y}_2, \mathbf{y}_3)$

A metric is thus generalization of the distance between the two points $d(\mathbf{y}_1, \mathbf{y}_2)$ in Euclidean space to other spaces and other distance-like measures. Any metric in the objective space fulfills the requirements for a value function when one of the points, i.e. vectors of outcomes, being compared is fixed so that the metric can be treated as a function of a single vector of outcomes. Metrics, by means of the triangle inequality (property 4 above), have the additional property that ensures that nearby designs have similar values.

These properties, along with metrics' simple forms and ease of implementation, likely account for their popularity in MCDM methods. It is unclear, however, whether metrics are actually well suited for expressing preferences, or if they are used simply for their convenience. An implication of metric-based MCDM methods is that value is equated with distance from some "ideal" solution. Distance, as defined by the metric, thus acts as a surrogate for similarity in the sense that designs that are near the ideal solution are deemed to be most similar to it and therefore preferred to designs that are farther away. While this is a convenient, intuitive analogy, it is perhaps taken for granted that it will yield a meaningful ranking of alternatives.

³In the context of MCDM these are understood to be vectors in the objective space

CHAPTER III

EXPERIMENTAL OBSERVATIONS OF PARETO FRONTIERS AND THEIR IMPLICATIONS

The Pareto Simplex Exploration method described in the next chapter is based on the realization that the Pareto frontier is generally an invertible subset of the objective space with a simplex-like geometry. This chapter describes the experimental observations that led to this realization. The experiments presented in this chapter are additionally described in Reference [35].

These experiments were motivated by the scarcity of information about real multi-dimensional Pareto frontiers in the engineering and optimization literature. Published examples of Pareto frontiers are often idealized and low dimensional, as evidenced by the examples in Figure 22. Correspondingly, there is little discussion in the literature about real Pareto frontiers' geometry and structure beyond whether or not a Pareto frontier is convex, and consequently there are few methods that utilize geometry as a motivation or enabler. The intent of the experiments presented in this section was to investigate properties of real Pareto frontiers that may suggest particular strategies for developing a method of exploring Pareto frontiers.

3.1 Two-Attribute Problem

The problem considered here is the design of a turbofan engine for a 150-passenger class airliner. There are six design variables (Fan Pressure Ratio, Low Pressure Compressor Pressure Ratio, Overall Pressure Ratio, T4max, Velocity Ratio, and Turbine Loading) and five attributes (Fan Diameter, Takeoff Field Length, Max Range, TSFC, and Engine Weight). All attributes are to be minimized except Max Range, which is maximized. The design analysis mapping from the design variables to the attributes is a set of artificial neural networks and quadratic response surface equations built from a basis data set that was created using NPSS and FLOPS. The creation of these surrogate models is described

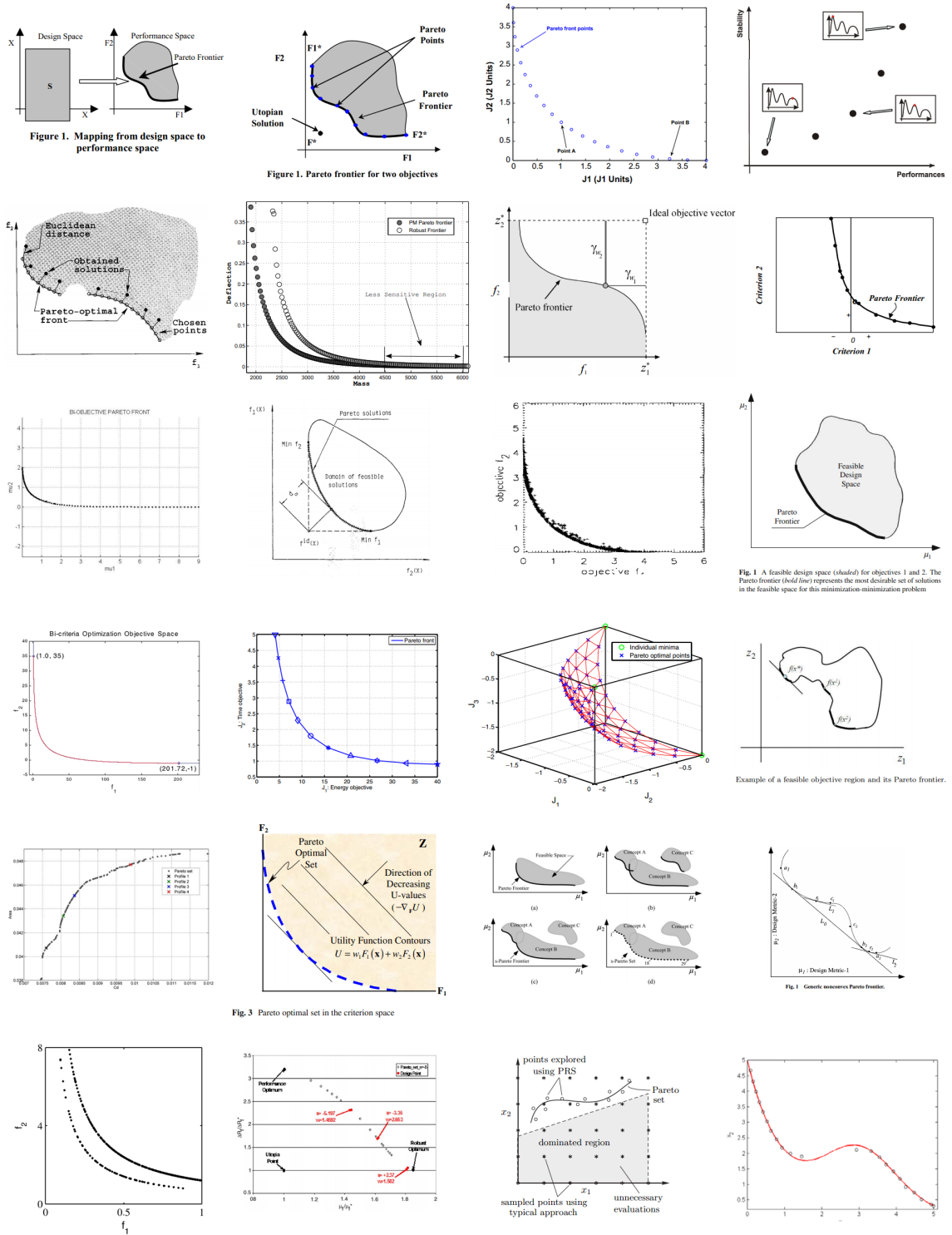


Figure 22: Examples of Pareto frontier illustrations that have appeared in journals and conference papers. (Sources are listed in Appendix A)

in Reference [92]. The sections below consider the Pareto frontiers of subsets of the five attributes. The Pareto frontiers were calculated with a Gray-coded NSGA-II algorithm [39], using large population sizes in order to clearly illustrate the various interesting properties.

The two-attribute frontier of Engine Weight and TSFC is shown in Figure 23a for a population of 300 and in Figure 23b for a population of 30,000, with 100 generations in both cases. The left side of the figures show the image of the Pareto frontier in the objective space. The small and large-population graphs look similar, although the larger population better explores the ends of the frontier. The right side of the figures show the preimage of these frontiers in the (normalized) design space. Although the frontiers are discretely sampled, these paths have been drawn as continuous curves, sorted in order of increasing Engine Weight, to better show the sequential ordering of the points along the frontier.

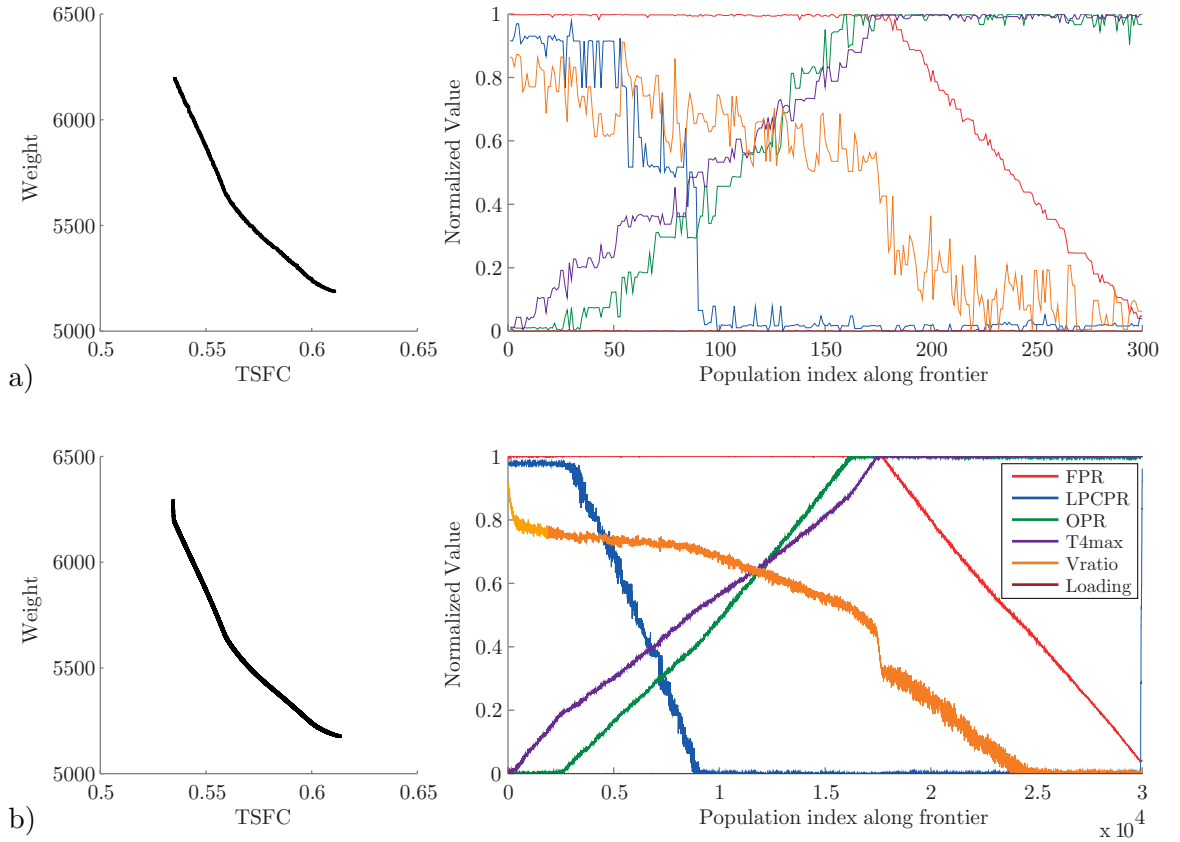


Figure 23: Two-attribute NSGA-II results. a) 300 population, 100 generations. b) 30,000 population, 100 generations

For this two-dimensional objective space, the Pareto frontier is seen to be one-dimensional:

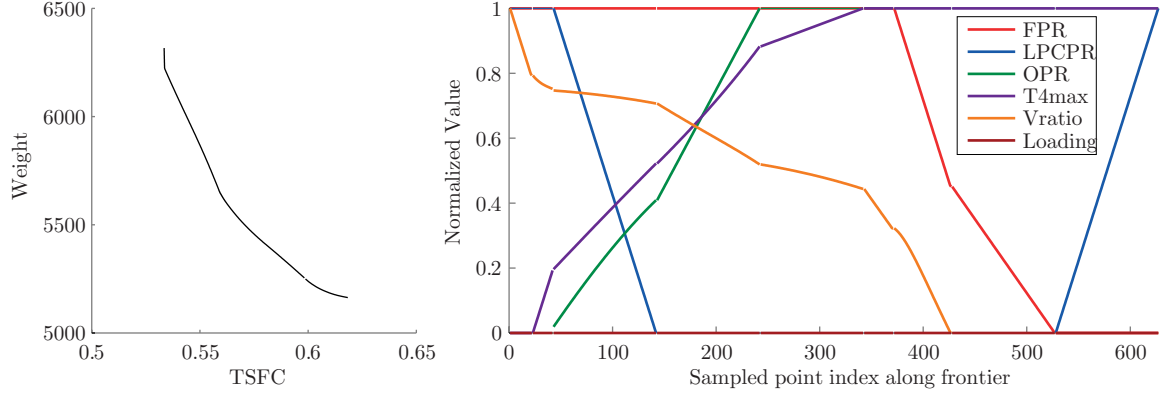


Figure 24: Two-attribute frontier found using a path-tracing algorithm

a curve connecting the minimum TSFC outcome to the minimum Engine Weight outcome. The fact that the design analysis is continuously differentiable (comprising polynomial and neural network regressions) suggests that the preimage of the Pareto frontier in the design space should also be one-dimensional, and indeed the right hand side of the figures confirm this by the continuity of the curves (more clearly visible in the 30,000 population case).

Two additional observations are made from these plots. First, although the Pareto frontier appears well converged in the objective space, the plots of the design space show significant noise in the design space for the 300 population, and a fair amount of noise even with a population of 30,000. In the 30,000 population case, the local deviations are small, typically around 2% of the variable’s range, and it is easily inferred that the “true” Pareto frontier is continuous and unique in the design space. In the 300 population case, the local deviations in the design variables around any particular point on the frontier are significantly larger and may be between 5% (for FPR, OPR, and T4max) to as much as 20% (for LPCPR and Vratio) of the variables’ ranges.

A designer who selects a set of designs within a small region of interest from this sampling of the Pareto frontier might be surprised to find such large fluctuations in the design variables; indeed it may be impossible to ascertain any meaningful relations between the designs within a small neighborhood. In fact, noisiness in the design space is to be expected on account of the mechanism by which NSGA-II optimizes. Fitness is calculated according to domination and crowding (nearness to other points) in the objective space, with no

regard to the shape of the frontier in the design space.

The 30,000 population case demonstrates better convergence because both dominance and crowding are functions of a point’s *relation to other points* in the population. Regardless of its source, the noise may lead a designer to believe that the fluctuations are indications that the non-dominated attribute vectors are non-unique, and that a wide variety of designs yield these vectors.¹ It will be shown later in this section that this conclusion would be wrong—the non-dominated designs in this problem are unique. It is worth noting, however, that the fact that such wide fluctuations in the design variables lead to very similar attribute vectors suggests that while the designs on the “true” Pareto frontier are unique, there is a large amount of design freedom that can be leveraged by pursuing attribute values that are near, but not precisely on, the Pareto frontier. Design freedom on and off the Pareto frontier has recently been examined by Simov and Ferguson [121].

The second observation from the above examples is the significant diversity of the efficient designs. For this relatively simple two-attribute frontier, five of the six design variables span their full range of feasible values along the frontier. Thus although the frontier itself is 1-dimensional, the tradeoffs between design variables along this 1-D path are complicated. Indeed, two or more design variables are changing values simultaneously almost everywhere on the frontier.

As a separate exercise, this two-attribute frontier was sampled using an algorithm that traces the path of efficient designs by directly enforcing the multi-objective optimality conditions. This is a continuation-type multi-objective algorithm that works only for two-attribute problems. The results of using it on this problem are shown in Figure 24, confirming that the true Pareto frontier is indeed unique and continuous in the design space. As in the previous figures, the frontier is drawn as connected to emphasize its continuity in the design space.

Whereas NSGA-II seeks an even distribution of points in the objective space, this algorithm seeks an even distribution in the design space. Consequently, the plot of the design

¹The issue of uniqueness here refers to the question of whether a given non-dominated outcome is the result of a unique design, or if there are multiple designs that have the same outcome.

variables in Figure 24 is locally stretched/compressed horizontally compared to the plots in Figure 23, but some “landmark” features are clearly seen in both figures.

The well-defined preimage of the Pareto frontier in the design space seen in Figure 24 is indicative of the type of improved understanding of Pareto frontiers that this dissertation seeks to enable. By using visualizations similar to the one in Figure 24, the decision maker can understand not only the available outcomes and required tradeoffs in this problem, but the values and changes in the design variables that lead to different non-dominated outcomes. To be sure, this visualization does not contain all information that the decision maker may wish to understand. For example, it is difficult to infer the sensitivities of the outcomes to the unconstrained design variables, and no information is displayed about the sensitivities of outcome to the constrained design variables. However, this visualization demonstrates the general idea of interpreting the Pareto frontier as a continuous manifold, which can be explored using a coordinate system. In this example the coordinate is simply the count of points along the Pareto frontier, mapped to the horizontal axis of the right hand graph.

This experiment led to the following research question:

Research Question: How can the idea of parameterizing a Pareto frontier using coordinates be extended to higher-dimensional (>2 dimensions) Pareto frontiers, for which the continuation approach’s idea of previous and next points is not well defined?

A related experiment on the two-attribute frontier was conducted to make observations related to the multi-objective optimality conditions and the uniqueness of non-dominated designs. Figure 25 shows a graphical representation of the Jacobian at one design selected from the 30,000 population-member NSGA-II results. The point at which the derivatives are calculated is located at the intersection of the arrows. Each arrow shows the path that would be traced out by making a small change in the value of one design variable, as indicated in the legend. (The effect of LPCPR is weak compared to the other variables, so a larger deviation is shown.) Figure 25 shows that the partial derivative paths with respect to

LPCPR, OPR, T4max, and Vratio (the unconstrained variables at this point) correspond to motion along the frontier. The two constrained variables, loading and FPR, do not share this property. It is easily seen that if these variables were not constrained, decreasing Loading would lead to a design that dominates this baseline. Similarly, a combination of increasing FPR while also increasing LPCPR, T4max, or OPR, or while decreasing Vratio, would lead to a design that dominates the baseline.

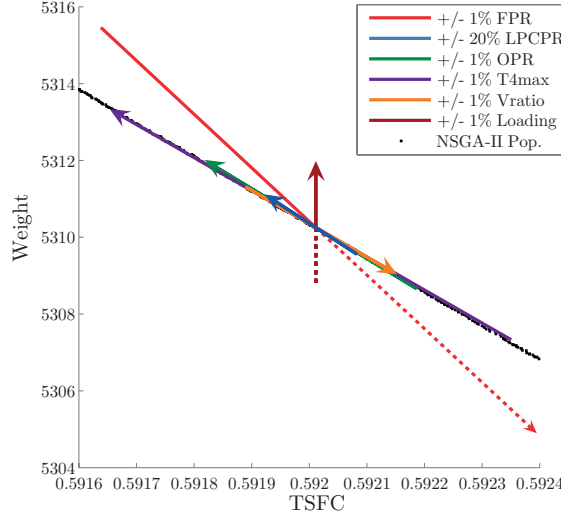


Figure 25: Graphical representation of the Jacobian (calculated numerically) at one NSGA-II population member. Dashed lines indicate moves that would violate the side constraints; arrow head shows direction of increasing independent variable.

3.2 Four-Attribute Problem

Figure 26 shows four views of a 4-attribute (Max Range, Weight, TOFL, and Fan Diameter) Pareto frontier plotted in three of its attributes. This frontier was calculated using a population of 30,000 with 300 generations. The frontier, which is plotted in Figure 26a in its entirety, has two distinct regions: a region where it is locally 2-dimensional and a region where it is locally 1-dimensional. (This is not an illusion caused by plotting in only three of the four attributes; the same distinct topology can be seen by plotting this frontier with any other subset of its attributes as the three axes.) Figures 26b-d show different subsets of the sampled Pareto frontier that are also Pareto optimal with respect to a subset of the attributes.

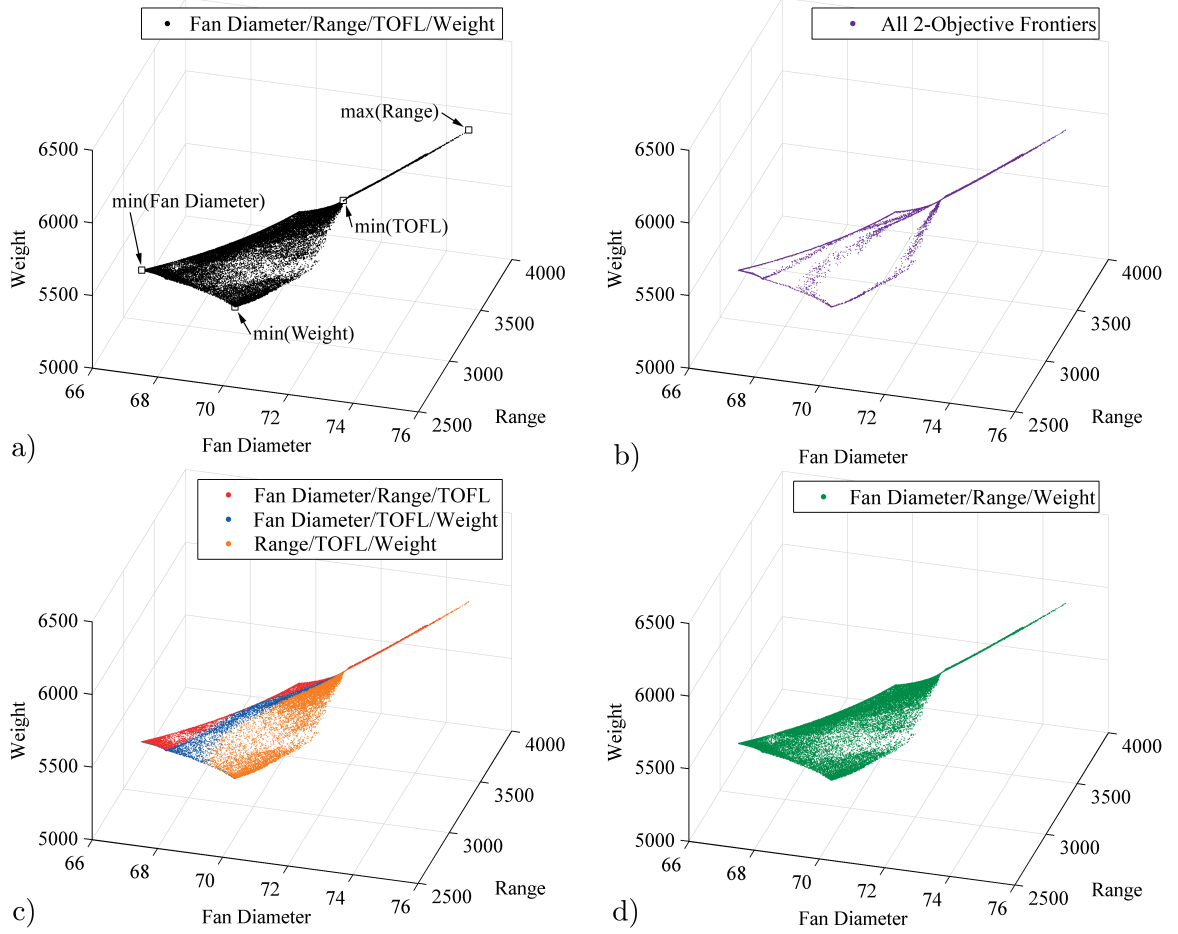


Figure 26: The 4-objective Pareto frontier of Fan Diameter, Range, TOFL, and Weight projected into 3 dimensions. a) Full 4-attribute frontier. b) 2-attribute frontiers. c,d) 3-attribute frontiers.

This experiment led to the key observation of how subfrontiers relate to the full Pareto frontier that suggests the strategy of forming a bijection between a Pareto frontier and a simplex.² The effect can be seen most clearly in Figure 26b and c, which show that the three-attribute subfrontiers join along edges, and that these edges are the two-attribute subfrontiers.

A second set of observations concerns the dimensionality of the subfrontiers. Although this problem has four attribute, the Pareto frontier is not three-dimensional anywhere.

²To simplify this discussion, the subsets of points that are non-dominated with respect to fewer attributes will be called “subfrontiers,” although it has not yet been shown that these subsets are equivalent to the subfrontiers as defined in Section 1.1.2

This seems to suggest that nowhere on the Pareto frontier are all attribute simultaneously conflicting. Figure 26d shows that the subfrontier with respect to Fan Diameter, Range, and Engine Weight is identical to the Pareto frontier in all four attributes, suggesting that the objective minimize Takeoff Field Length is redundant everywhere.

This experiment led to the following research questions:

Research Questions:

- Are Pareto-efficient-in-fewer-attribute subsets of the sampled Pareto frontier equivalent to subfrontiers? (I.e. can subfrontiers contain points not on the full Pareto frontier?)
- Are $(p - 1)$ -attribute subfrontiers always bounding subsets of p -attribute subfrontiers?
- What factors determine the local dimensionality of the Pareto frontier?

The next section begins to answer some of the research questions that resulted from these experiments by considering the mathematics of multi-objective optimality.

3.3 Formalization of Pareto Frontier Geometry

This section formalizes some of the observations in the previous section by describing the properties that lead to topological similarities of Pareto frontiers and simplices³. We begin by describing the geometry of simplices of various dimensionality. Next we show how the dominance definitions lead to a simplex-like structure for regular Pareto frontiers. Finally, we consider the issue of Pareto frontier dimensionality and show that the “full-dimensional” Pareto frontier of a k -dimensional objective space corresponds to a $(k - 1)$ -dimensional simplex.

³Simplices are defined in Section 1.3.

3.3.1 Relation Between Simplex Geometry and Pareto Frontiers

Three properties of simplices suggest that they may be relevant to the study of Pareto frontiers. First, the geometry of a p -simplex is such that, when it is embedded in an $(p+1)$ -dimensional Cartesian coordinate system with its vertices as the Cartesian unit vectors, a move from any point on the simplex to any other point also on the simplex necessarily increases some coordinates while decreasing others. This property is closely related to the Pareto optimality conditions, which require improvement in one attribute only at the expense of others. Second, the number of $(p-1)$ -dimension faces of a simplex and the number of p -subfrontiers of a Pareto frontier are equal – both corresponding to the entries in the p -th row of Pascal’s triangle. Third, the Pareto frontier’s subfrontiers are arranged and intersect in a way that mimics the arrangement of the faces of the corresponding simplex.

The remainder of this section is devoted to exploring these similarities between simplices and Pareto frontiers. We proceed by next examining the structure and arrangement of the subfrontiers and their relation to the faces of a simplex.

3.3.2 Structural Similarities Between Pareto Frontiers and Simplices

The goal of this section is to demonstrate that when $\mathcal{P}(Y)$ is a regular Pareto frontier, defined here to mean that all weakly non-dominated points are strongly non-dominated, it is simplex-like in the sense that its p -subfrontiers are bounded by their $(p-1)$ -subfrontiers in the same way that the p -faces of a simplex are bounded their $(p-1)$ -faces.⁴ Because each p -subfrontier has p $(p-1)$ -subfrontiers of its own, this further implies that each p -subfrontier has p “edges,” which gives rise to the characteristic simplex-like shape.

We begin by stating some basic results about how subfrontiers $\mathcal{P}(A)$ and $\mathcal{P}(B)$ relate to the (regular) Pareto frontier $\mathcal{P}(Y)$. If $\mathcal{P}(Y)$ is irregular, then these results hold if consideration is limited to a regular subset of $\mathcal{P}(Y)$.

1. If $\mathbf{y}^* \in \mathcal{P}(A)$ then $\mathbf{y}^* \in \mathcal{P}(Y)$

Let A^- denote the convex polyhedral cone $\{\hat{\mathbf{y}} : \hat{\mathbf{y}}_i \leq \mathbf{y}_i^* \ \forall \ i \in A\}$ and let Y^- similarly

⁴The symbols used in this section are defined in Section 1.1.

denote the cone $\{\hat{\mathbf{y}} : \hat{\mathbf{y}}_i \leq \mathbf{y}_i^* \ \forall i \in Y\}$. When the objectives are to minimize, the weak dominance criterion requires that $\mathbf{y}^* \in \mathcal{P}(A)$ if and only if $\{\mathbf{y}^* \in f(D)\} \cap A^- = \emptyset$ and $\mathbf{y}^* \in \mathcal{P}(Y)$ if and only if $\{\mathbf{y}^* \in f(D)\} \cap Y^- = \emptyset$. Noting that A^- and Y^- are the sets of points that satisfy some inequality constraints, when $A \subset Y$, then $Y^- \subseteq A^-$, since Y^- requires points to satisfy all the inequality constraints that define A^- and additional constraints for the attributes in $Y \setminus A$. Therefore $\mathbf{y}^* \in \mathcal{P}(A)$ implies $\mathbf{y}^* \in \mathcal{P}(Y)$. ■

2. If $A \subset Y$, then $\mathcal{P}(A) \subseteq \mathcal{P}(Y)$

This statement simply applies (1) to all $\mathbf{y}^* \in \mathcal{P}(A)$. $\mathcal{P}(A)$ is not required to be a strict subset of $\mathcal{P}(Y)$, but generally if $A \neq Y$, there are points in $\mathcal{P}(Y)$ that are not in $\mathcal{P}(A)$. ■

3. When A and B are both subsets of Y , $\mathcal{P}(A) \cap \mathcal{P}(B) = \mathcal{P}(A \cap B)$

By (2), $\mathcal{P}(A \cap B) \subseteq \mathcal{P}(A)$ and $\mathcal{P}(A \cap B) \subseteq \mathcal{P}(B)$ so that $\mathcal{P}(A \cap B) \subseteq \mathcal{P}(A) \cap \mathcal{P}(B)$. It remains to be shown that there are no points that are in both $\mathcal{P}(A)$ and $\mathcal{P}(B)$ but not in $\mathcal{P}(A \cap B)$. Let A^- denote the cone $\{\hat{\mathbf{y}} : \hat{\mathbf{y}}_i \leq \mathbf{y}_i^* \ \forall i \in A\}$ and let B^- denote the cone $\{\hat{\mathbf{y}} : \hat{\mathbf{y}}_i \leq \mathbf{y}_i^* \ \forall i \in B\}$. As in (1), $\mathbf{y}^* \in \mathcal{P}(A)$ if and only if $\{\mathbf{y}^* \in f(D)\} \cap A^- = \emptyset$ and $\mathbf{y}^* \in \mathcal{P}(B)$ if and only if $\{\mathbf{y}^* \in f(D)\} \cap B^- = \emptyset$. These conditions similarly require that the intersection of any subset of A or B respectively with the set $\{\mathbf{y}^* \in f(D)\}$ must be empty. In particular this requires that $A^- \cap B^- \cap \{\mathbf{y}^* \in f(D)\} = \emptyset$, and $A^- \cap B^-$ is precisely the set $\{\hat{\mathbf{y}}_i \leq \mathbf{y}_i^* \ \forall i \in A \cap B\}$. ■

The above results imply several properties of Pareto frontiers that are analogous to properties of simplices. From result (3) above, the intersection of any two subfrontiers is itself a subfrontier in fewer attributes – in particular it is the subfrontier of the set of attributes that are common to the intersecting subfrontiers. Applying this property recursively leads to a hierarchy of subfrontiers that has the same structure as the hierarchy of faces in a simplex. This is most easily demonstrated with an example, as in Figure 27, which compares the faces, edges and vertices of a 3-simplex to the subfrontiers of $\min(\{a, b, c, d\})$.

For an arbitrary number of attributes, k , the number of p -subfrontiers is always equal to the number of $(p-1)$ -dimensional faces of a $(k-1)$ -simplex – both quantities being equal to

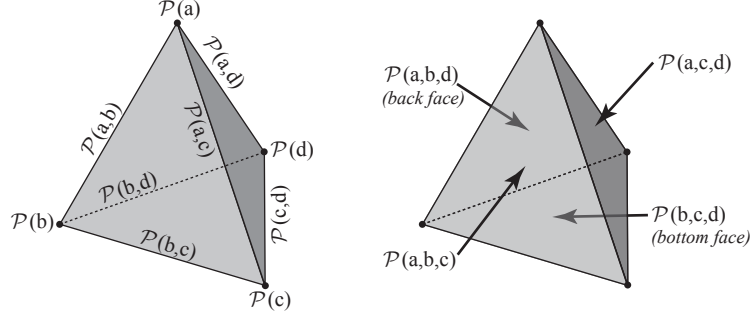


Figure 27: Relations between subfrontiers of a four-attribute optimization problem, showing similarity to 3-D simplex

$\binom{p}{k}$. Having established that the number and arrangement of the subfrontiers matches the number and arrangement of faces of a corresponding simplex, all that remains to be shown in comparing Pareto frontiers to simplices is that the dimensionality of the subfrontiers matches the dimensionality of the corresponding simplex faces. This is considered in the next section.

3.3.3 Local Dimensionality of the Pareto Frontier

For the general case in which the number of attributes, k , is less than the number of design variables, n , and the mapping from the design space to the objective space is smooth, the objective space is a k -dimensional manifold. It is well established that in this case the Pareto frontier is a subset of the boundary of the objective-space manifold, and so its dimensionality is generally equal to $k - 1$, the dimensionality of the boundary of the objective space.⁵ However, many real problems deviate from this ideal. If $n < k$, then the dimensionality of the objective space and the Pareto frontier is limited to at most n . Further reductions in Pareto frontier dimensionality occur because of (locally) nonconflicting attributes.

Figure 28 shows an example of an objective space whose Pareto frontier has varying

⁵The term “boundary” is used here to mean the subset that is not in the relative interior of the objective space.

local dimensionality. The surface drawn in Figure 28 is the objective space of the problem

$$\begin{aligned} \min_x \quad \mathbf{y} &= \begin{bmatrix} \cos(\frac{\pi}{4})x_1 - \sin(\frac{\pi}{4})x_2 \\ \sin(\frac{\pi}{4})x_1 + \cos(\frac{\pi}{4})x_2 \\ -3x_1^2 - x_2^3 \end{bmatrix} \\ \text{s. t.} \quad &-5 < x_1 < 5 \\ &-5 < x_2 < 5. \end{aligned} \tag{4}$$

The Pareto frontier of this problem, shown as the darkened portion of Figure 28, is one-dimensional in many places because there is locally no tradeoff between y_1 and y_3 or between y_2 and y_3 .

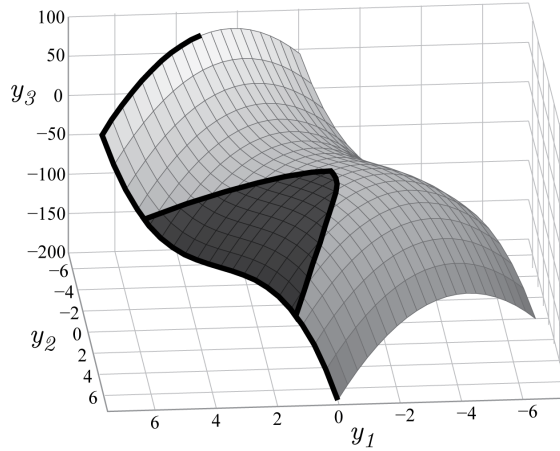


Figure 28: Pareto frontier (darkened) of the 3-attribute problem (4) demonstrating varying Pareto frontier dimensionality

The conditions that lead to reduced dimensionality Pareto frontiers are considered in the two subsections below, beginning with a graphical interpretation of Pareto optimality in two dimensions, and then expanding this idea to arbitrary dimensions by means of a condition on the Jacobian matrix.

3.3.4 Graphical Interpretation of Multi-Objective Optimality in 2-D

Figure 29 shows the outcome of a baseline design, $\tilde{\mathbf{x}}$ indicated by the white dot, viewed in a notional 2-D objective space. A vector space is defined locally around the baseline design, with quadrants defined by the roman numerals. Using the strong Pareto dominance partial

ordering with the attributes of minimizing y_1 and y_2 , any outcome in quadrant III will dominate $\tilde{\mathbf{x}}$, any outcome in quadrant I is dominated by $\tilde{\mathbf{x}}$, and outcomes in quadrants II and IV are incomparable to $\tilde{\mathbf{x}}$. We are concerned with the question of whether $\tilde{\mathbf{x}}$ is locally efficient.

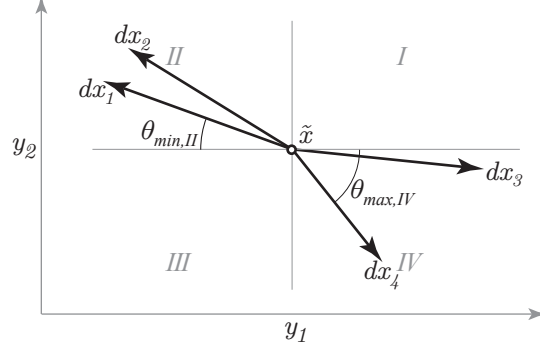


Figure 29: A baseline design $\tilde{\mathbf{x}}$ and four possible changes to $\tilde{\mathbf{x}}$ that correspond to small increases in four design variables.

The arrows in Figure 29 show the effect on y_1 and y_2 of increasing four different independent variables, $x_1 \dots x_4$ from their $\tilde{\mathbf{x}}$ values. We presume that none of these moves violates any constraint. If any such arrow were to lie in quadrant III, then clearly $\tilde{\mathbf{x}}$ would be dominated by the outcomes that result from making that move. Furthermore, it can be seen that a move of the form $\gamma_1 dx_1 + (1 - \gamma_1) dx_4$, $0 < \gamma < 1$ can result in an outcome in quadrant III, which will dominate the baseline outcome. We observe that $\tilde{\mathbf{x}}$ is locally efficient when no arrow lies in quadrant III and $\theta_{min,II} \geq \theta_{max,IV}$, where these angles are defined as in Figure 29 and are calculated over all feasible moves in the indicated quadrants.

Now suppose that the variables x_1 and x_2 are unconstrained, such that the moves dx_1 , dx_2 , $-dx_1$, $-dx_2$ are all feasible. Figure 30 shows this case (ignoring the possibility of moves dx_3 and dx_4 from Figure 29). We note that the relation between dx_1 and $-dx_2$ in Figure 30 is analogous to the relation between dx_1 and dx_3 in Figure 29, and so $\tilde{\mathbf{x}}$ is locally dominated. The interesting result is that $\tilde{\mathbf{x}}$ can only be non-dominated if these vectors are co-linear – precisely the behavior that was observed in Figure 25.

To generalize these observations, we first note that the vectors in Figs. 29 and 30 are $(\frac{\partial y_1}{\partial x_i}, \frac{\partial y_2}{\partial x_i})$, $i \in 1 \dots 4$ evaluated at $\tilde{\mathbf{x}}$ and that in this two-attribute problem, for convenience,

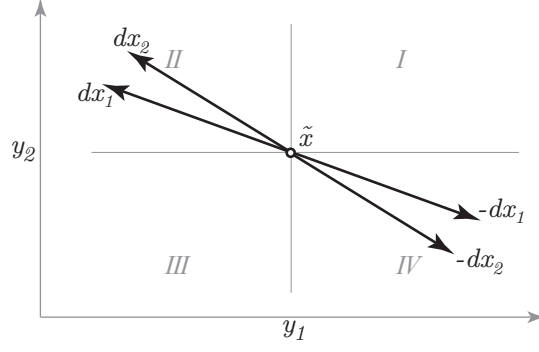


Figure 30: Two possible changes to a baseline design $\tilde{\mathbf{x}}$ due to increasing/decreasing two unconstrained design variables

we can consider the slope of these vectors in the y_1, y_2 plane. These slopes are equal to $\frac{\partial y_2}{\partial x_j} / \frac{\partial y_1}{\partial x_j}$ evaluated at $\tilde{\mathbf{x}}$. Then in order for $\tilde{\mathbf{x}}$ to be locally non-dominated with respect to two attributes, y_1 and y_2 , the following conditions must be met for all *feasible moves*:

1. No vector $(\frac{\partial y_1}{\partial x_i}, \frac{\partial y_2}{\partial x_i})$ has all non-positive components (with at least one non-zero component).
2. The slopes $\frac{\partial y_2}{\partial x_j} / \frac{\partial y_1}{\partial x_j}$ for all variables x_i that can both increase or decrease without violating a constraint must be equal.
3. The vectors $(\frac{\partial y_1}{\partial x_j}, \frac{\partial y_2}{\partial x_j})$ for the remaining variables x_j that can increase or decrease (but not both) without violating a constraint must lie above (i.e. in the dominated region) the line through $f(\tilde{\mathbf{x}})$ and $f(\tilde{\mathbf{x}} + dx_i)$ where x_i is defined as in condition 2.

The first condition expresses the trivial observation that if a feasible change of any design variable reduces both attributes, such a move will dominate the baseline design. Conditions 2 and 3 concern feasible moves that involve simultaneously changing multiple design variables. Of these, condition 2 is more interesting in the sense that it concerns variable whose values are changing along the frontier, whereas condition 3 concerns the variables that are fixed. Condition 3 explains the location of transition points where fixed design variables begin changing – this occurs in the limiting condition when the slope $\frac{\partial y_2}{\partial x_j} / \frac{\partial y_1}{\partial x_j}$ for a fixed variable matches the slopes described in condition 2.

3.3.5 Considerations in more than Two Attributes

With three or more attributes, the concept of equating slopes is no longer meaningful. For this case, we revert back to a vector representation. Given a vector $\mathbf{y}^* = f(\mathbf{x}^*)$ that is known to be non-dominated, we consider the pre-image X of a small neighborhood $Y = f(X)$ of non-dominated vectors around \mathbf{y}^* . Presuming that X is continuous in the design space, we define $active(\mathbf{x}^*)$ as the set of design variables whose values are not constant throughout X , and we call these design variables “active.” Note that $active(\mathbf{x}^*)$ is analogous to the variables x_i considered in condition 2 of the previous section.

Condition 2 from the previous section can then be expressed as

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_i} \\ \frac{\partial y_2}{\partial x_i} \end{bmatrix} dx_i = \begin{bmatrix} \frac{\partial y_1}{\partial x_j} \\ \frac{\partial y_2}{\partial x_j} \end{bmatrix} dx_j \text{ for all } i, j \in active(\tilde{\mathbf{x}})$$

Recognizing that the vectors in this equation are columns of the Jacobian matrix, we form the submatrix of the Jacobian by taking only the columns corresponding to the active variables and call this matrix $\tilde{J}(\mathbf{x}^*)$:

$$\tilde{J}(\mathbf{x}) = \begin{bmatrix} \frac{dy_1}{dx_1} & \dots & \frac{dy_1}{dx_m} \\ \vdots & \ddots & \vdots \\ \frac{dy_p}{dx_1} & \dots & \frac{dy_p}{dx_m} \end{bmatrix} \quad x_1 \dots x_m \in active(\tilde{\mathbf{x}}) \quad (5)$$

Since the non-active design variables are known to be constant in the region around \mathbf{y}^* we are not concerned with the corresponding columns of the Jacobian. The rank of \tilde{J} is thus at most equal to the lesser of the number of attributes and the number of active variables.

In this case, the dimensionality of the Pareto frontier at \mathbf{y}^* is then equal to the rank of $J(\mathbf{x}^*)$. A thorough discussion of the role of such Jacobian matrices in multi-objective optimization is given by Hillermeier [58]; here we simply note that the relation between the rank of \tilde{J} and the dimensionality of the Pareto frontier is related to the columns of $\tilde{J}(\mathbf{x}^*)$ forming a basis for the tangent space of the Pareto frontier at \mathbf{y}^* .

The local dimensionality of the Pareto frontier is thus limited to be at most the lesser of k and the number of unconstrained design variables. However, a necessary condition for

Pareto optimality is that the rank of \tilde{J} be strictly less than k , for otherwise there would exist a $d\mathbf{x}$ (with $dx_i = 0$ for the constrained x_i) such that $f(\mathbf{x} + d\mathbf{x})$ dominates $f(\mathbf{x})$.

Because the constraints have been temporarily removed from consideration, the only possible source of further reductions in the rank of \tilde{J} , and thus the Pareto frontier dimensionality, is a lack of conflict between attributes. A full-dimensional $(k - 1)$ Pareto frontier indicates that each attribute conflicts with each other attribute such that any one may be improved only at the expense of others. When some of the attributes do not conflict with each other, the local dimensionality is reduced just as if the redundant attributes, i.e. all but one of the nonconflicting attributes, had been removed from the problem with k being reduced correspondingly.

In this dissertation we are primarily concerned with the full-dimensional case in which the Pareto frontier is locally $(k - 1)$ -dimensional at each point that is not a boundary point. Because the subfrontiers are simply Pareto frontiers calculated over fewer attributes, the rank condition described in this section applies to subfrontiers by limiting \tilde{J} to include only the rows corresponding to the attributes in the subfrontier of interest. The $(k - 1)$ -attribute subfrontiers are thus limited to at most a dimensionality of $k - 2$, which is the dimensionality of the *boundary* of the k -attribute Pareto frontier. Applying this property recursively reveals the simplex-like structure of the subfrontiers illustrated in Figure 27 in which, for example, the 2-attribute subfrontiers are the 1-dimensional boundaries (edges) of the 3-attribute subfrontier faces.

3.3.6 Deviations from Simplex-Like Geometry

The above sections have described the structural similarities between simplices and Pareto frontiers, in particular that there is generally a correspondence between the number and dimensionality of the faces of a simplex and the subfrontiers of a Pareto frontier. However, Figures 2 and 28 have shown that not all Pareto frontiers exhibit a strictly simplex-like structure. This section briefly considers the possible sources of non-simplex-like structures in a Pareto frontier in more detail.

For any sampling of a Pareto frontier, at least one vector optimizes each individual

attribute. Consequently, by result (2) of Section 3.3.2, each subfrontier is non-empty, and the number of subfrontiers depends only on the number of attributes and thus always matches the number of faces of the corresponding simplex. Deviations from a simplex-like structure are therefore limited to the dimensionality of the subfrontiers differing from the dimensionality of the corresponding simplex faces.

As discussed in the previous section, the dimensionality of a particular simplex face is equal to the maximum possible dimensionality of the corresponding subfrontier. When any pair of attributes does not conflict, meaning that both can be improved without tradeoff over some portion of the objective space, the local dimensionality of the Pareto frontier is reduced by one. Correspondingly, the local dimensionality of all subfrontiers involving this pair of variables is also reduced. The result is that in addition to the local individual optima (local in this sense meaning the optima considered only over the portion of the Pareto frontier with reduced dimensionality) being 0-D (points), the two-attribute subfrontiers of non-conflicting attributes are 0-D, indicating that the local individual optima of these points coincide. This can be seen in Figure 28 which has three 1-D segments on the Pareto frontier. The uppermost segment connects (local) $\max(y_3)$ to $\max(y_1, y_2)$, the middle segment connects (local) $\max(y_1, y_3)$ to $\max(y_2)$, and the lower segment connects (local) $\max(y_1, y_3)$ to $\max(y_2)$.

Thus far we have assumed that the Pareto frontier is continuous in the objective space and is regular, as defined in Section 1.1.3. The Pareto frontier being discontinuous in the objective space is not generally problematic, but the mapping from the coordinates described in the next section to the objective space will be similarly discontinuous. When the Pareto frontier is irregular, different results are obtained depending on whether the weak or strong Pareto optimality conditions are used. Using the weak Pareto optimality conditions, subfrontiers may have higher dimensionality than they would in the regular case. This is shown in Figure 2b, in which the subfrontier of $\mathcal{P}(\{y_3\})$ is 1-D instead of being 0-D. Using the strong Pareto optimality conditions, result (1) in Section 3.3.2 is no longer true for all points (consequently result (2) is also false). We note however, that irregularities are usually limited to certain regions of the Pareto frontier, and the remainder of the frontier

may behave as if it were “locally regular.”

CHAPTER IV

METHODOLOGY

The literature review in Chapter 2 provides a survey of some of the methods and tools that can be used to explore a multi-attribute design problem with the goal of either developing a value function or otherwise gaining enough knowledge about the design alternatives to allow a decision to be made.

For some problems, these techniques can be used effectively to reach a decision that is at least close to the true (unknown) “best” decision. But in many cases, multi-criteria decision making remains difficult, even in the simplified deterministic case considered here. At least two high-level sources of difficulty are apparent. First, the methods described in Chapter 2 are simply not applicable in all cases. Computational limitations may preclude the use of multi-objective optimization when the design analysis is slow. The decision maker may be unable to identify an MCDM method that offers an agreeable value function form. The sampled data may be too complex or high-dimensional for to be visualized effectively. Second, even when the methods are applicable, they may be difficult to use effectively. Multi-objective optimizers must be carefully matched to the characteristics of the design analysis if they are to provide an even sampling of the entire Pareto frontier with a minimum number of function calls. MCDM methods may require the decision maker to specify values for relative weights on the attributes without offering guidance for how these weights can be chosen to reflect true preferences. Interactive, multivariate visualization may require specialized software and expertise that decision makers lack.

Among the specific difficulties and shortcomings observed in Chapter 2 are:

- MCDM techniques are not applicable if the decision maker does not agree with the specific premise on which the technique bases its value function. For example, the TOPSIS method is “based on the concept that the chosen alternative should have the shortest distance from the positive-ideal solution and the longest distance from the

negative-ideal solution” [148]; decision makers may be unsure if they agree with this rather abstract premise.

- Most MCDM techniques provide only a functional form for the value function and leave it to the decision maker to select values for relative weights of the attributes or other parameters. Yet these weights are meaningful only in the context of the value function, making it exceedingly difficult to select their values.
- Most MCDM techniques base value on a metric defined in the objective space, e.g. distances from each alternative to an ideal design. It is not clear that the interpretation of the objective space as a Euclidean metric space is meaningful considering that attainable subset of the objective space has an unknown, potentially complicated, topology.
- The vast majority of multi-variate visualization techniques are unrelated to using data to make a decision. Consequently, they do not attempt to order or represent the data in terms of optimality.
- There has been little investigation of how the specific geometry of Pareto frontiers, as observed in Chapter 3, can be used to inform the selection of or creation of new visualization techniques.
- The few visualization techniques that have been developed specifically for visualizing Pareto frontiers are discrete techniques that attempt to visualize the entire Pareto frontier. No continuous techniques exist for visualizing local tradeoffs around a point on the frontier.

The present work is predicated on the belief that much of the difficulty of design decision making, including several of the shortcomings listed above, stems from a lack of understanding of the geometry of the objective space – more specifically, the potentially optimal portion of the objective space as defined by the Pareto frontier. The particular approach taken here is to enable greater understanding of the Pareto frontier by using an alternate form of the design analysis.

It may seem that the aforementioned shortcomings of MCDM and multivariate visualization are largely unrelated to the form of the design analysis, $\mathbf{y} = f(\mathbf{x})$. Consider, however, a design analysis of the form $\mathbf{x} = g(\mathbf{y})$. This “inverse design” formulation would allow the designer to simply propose desirable outcomes and obtain a corresponding design. There can be little doubt that this would greatly simplify design decision making. Unfortunately, such inverse functions are generally not merely difficult, but impossible, to obtain.

The primary objective of this dissertation is the creation of a method for developing an alternate design analysis of the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ ¹, where $\boldsymbol{\psi}$ is a tuple of barycentric coordinates. This formulation is not as powerful as the hypothetical inversion described in the previous paragraph, but it has the very desirable property that each attribute y_i has a corresponding coordinate ψ_i with which it increases monotonically. The $\boldsymbol{\psi}$ coordinates thus act as a set of independent (up to the barycentric constraint $\sum \psi_i = 1$) proxies for the attributes that describe their relative balance. In many ways, the $\boldsymbol{\psi}$ coordinates are thus similar to the weights that appear in some multi-objective optimization algorithms’ aggregate objective functions and some MCDM approaches’ value functions. An important difference, however, is that the $\boldsymbol{\psi}$ coordinates are constructed *from the sampled Pareto frontier* and thus are able to adapt to the geometry of the particular Pareto frontier being considered rather than forcing the representation of the Pareto frontier to adapt to the geometry implied by a particular weighted aggregate objective function or value function.

The intended application of the inverse-like function $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ is not as a replacement for the original design analysis. Rather, the two functions are intended to be used side by side: the original function when it is more useful to explore the problem by using with the physical parameters that define the design (\mathbf{x}) as independent variables, and the inverse-like function when it is more useful to explore the problem by using the relative balance of optimality in the attributes ($\boldsymbol{\psi}$) as independent variables.

The particular strategy that will be taken here is to construct an inverse-like function of the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ by limiting consideration to *only the Pareto frontier* of the original

¹For brevity, g is denoted as being a single function with two vector-valued outputs, but in the actual implementation separate scalar-valued functions are created for each attribute and design variable.

design analysis. This strategy is motivated by the realization of the similarities between Pareto frontiers and simplices described in Chapter 3. This following hypothesis summarizes this approach:

Hypothesis: Because of the topological equivalency between *regular Pareto frontiers of full dimensionality* and *simplices of matching dimensionality*, full-dimensional Pareto frontiers may be parameterized by a barycentric coordinate system so that the resulting mapping from coordinates to Pareto-efficient outcomes is one-to-one and onto, i.e. bijective. The resulting coordinates, ψ , can then serve as independent variables in an interpolating function of the form $(\mathbf{y}, \mathbf{x}) = g(\psi)$

Chapters 5–7 address this hypothesis by defining three methods for creating a coordinate system for the Pareto frontier. Each of these methods has different merits which may make it preferable for different types of design problems. Chapter 8 then demonstrates the use of these coordinate systems and the associated inverse-like functions for exploring several example problems.

This chapter proceeds by first presenting a list of assumptions about the nature of the design problem that limit the scope of this work. Next, the problem of developing an inverse or inverse-like design analysis is considered. The major steps of a process, named “Pareto Simplex Exploration” for creating and using an inverse-like analysis function is then described. Finally, several example problems are defined that will be used to demonstrate the coordinate systems introduced in Chapters 5–8 and their use for forming the inverse-like functions described above.

4.1 Assumptions about the Design Problem

In order to manage the scope of this dissertation, some assumptions are made regarding the nature of the design problem and the Pareto frontier. These are listed below along with brief descriptions of the motivation for making each assumption.

- The *design problem* is provided in the form described in Section 1.1.

The details of this assumption can be found in Section 1.1. In short, this assumption limits consideration to design problems whose design analysis can be expressed in the form $\mathbf{y} = f(\mathbf{x})$. The process of abstracting a particular design problem into this functional form is considered to be outside the scope of this dissertation. Additionally, this form precludes the existence of variables that affect the outcomes but are not under the decision maker’s control, i.e. “noise variables.”

- All variables (*design variables* and *attributes*) are scalar-valued, and there are a reasonably small number of each type of variable.

The coordinate systems introduced here theoretically do not limit the number of attributes that may be considered, but for practical purposes the number should be small. Problems with 2-6 attributes are probably appropriate, whereas problems with more than 10 attributes are probably not. This same restriction could be placed on almost any MCDM or multivariate visualization method. The number of design variables is less constraining, as the coordinate systems are constructed based solely on the values of the attributes. If the number of design variables is large, the designer may wish to select only an interesting subset of them to model as outputs of the function $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$.

Limitation to scalar values ensures that the notions of minimizing and maximizing attributes, as well as constraining design variables between minimum and maximum bounds are well defined.

- The *design analysis* is deterministic and all variable values are treated as being known with certainty.

In order to simplify the development of the coordinate systems, uncertainty is not considered in either the design analysis or the variable values. This is a significant limitation, as uncertainty is always present in real design problems and is perhaps the primary source of difficulty in design. Additionally, an underlying premise of this work is that the set of alternatives may be restricted to the Pareto frontier without risk of excluding the best alternative. This is true only if the attributes on which the

Pareto frontier is based are the only factors affecting the decision. In the presence of uncertainty, the best design may not be among the deterministically non-dominated set.

Considering, however, that the development of coordinate systems for Pareto frontiers is an entirely new approach, it seems appropriate to first develop useful techniques for the deterministic case before attempting to solve the parameterization of uncertain Pareto frontiers.

- The *design variables* are continuous, and the *design analysis* is a continuous function.

These assumptions reflect the type of example problems that are considered in this work. It is possible that the coordinate systems may prove useful in discretely valued or discontinuous problems, but these cases are not considered here.

- The only constraints are upper and lower limits on the allowable values of each *design variable*.

This assumption is used to ensure that the design space is bounded and the constraint qualifications needed for the multi-objective KKT conditions are met [81]. Although not demonstrated in this dissertation, other types of constraints are also allowable as long as they do not lead to irregular Pareto frontiers. For example, (active) constraints that directly limit the optimality of the attributes always lead to irregular Pareto frontiers and are therefore largely inappropriate for the present work. One of the example problems to be considered in the subsequent chapters, the truncated sphere described below, is irregular due to such a constraint. As will be shown the coordinate systems generally still work well in regions away from the irregularity.

- The Pareto frontier is regular and full dimensional.

This assumption is needed to ensure that the mapping from coordinates to Pareto-efficient outcomes is one-to-one. As noted above, one of the example problems has an irregular Pareto frontier to demonstrate the effects of using these coordinate systems in such cases. Because the Pareto frontier is known only as a discrete set of sampled

points, the requirement that the Pareto frontier be regular could be relaxed to “the sampled Pareto frontier is regular.” Thus, a regular subset of a larger irregular Pareto frontier could be considered within the scope of this work.

Similarly, a constant-dimensional subset of a varying-dimensional Pareto frontier could be used. Varying-dimensional (subsets of) Pareto frontiers are not considered. Parameterizing such frontiers would require a coordinate system that also has varying dimensionality, and this complication is considered to be outside the scope of the present work.

- The *design analysis* is sufficiently fast that it is not a limiting factor in the selection of a multi-objective optimization algorithm.

Creating any of the coordinate systems described in Chapters 5–7 requires a computation time on the order of a few minutes, but each requires as an input a set of non-dominated points that make take significantly longer to obtain using multi-objective optimization. Consequently the limiting factor in defining a coordinate system for the Pareto frontier is the ability to obtain an appropriately sampled frontier using multi-objective optimization. A reasonable interpretation of this assumption is that the design analysis is assumed to comprise fast (surrogate) models such as polynomial response surface equations or artificial neural networks. Design analyses comprising physics-based analyses or iterative convergence loops are likely unsuitable for the proposed method; however, the method may become applicable if faster surrogate models of the design analysis can be created.

Of these assumptions, the two that most limit the practical applicability of this work are that the problem is being treated deterministically and that the Pareto frontier must be full-dimensional. These assumptions are violated by many real design problems. However, as indicated above, these complications seem best left for future work in this area.

4.2 Reformulating the Design Analysis on the Pareto frontier

The general strategy behind the approach outlined at the beginning of this chapter is to reformulate the design analysis from the form $\mathbf{y} = f(\mathbf{x})$ to the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ where $\boldsymbol{\psi}$, represents a tuple of barycentric coordinates. This section describes the reasoning behind this functional form as being a compromise “inverse-like” formulation that can be used in place of the generally unobtainable true inverse.

The design analysis is invertible if and only if it is both *surjective* and *injective*. A function $f : X \rightarrow Z$ is called *surjective* or *onto* if every $z \in Z$ equals $f(x)$ for some $x \in X$. Consequently, the fulfillment of this property depends on the choice of the sets X and Z . A function is called *injective* or *one-to-one* if for every $z \in Z$ there is at most one $x \in X$ such that $z = f(x)$ where $f : X \rightarrow Z$ as before. Note that if f is also surjective, this condition strengthens to require that there is *exactly* one $x \in X$ such that $z = f(x)$. A function that is both surjective and injective is also called *bijective*, or a *bijection*. By definition, given a bijection $f : X \rightarrow Z$, there exists a function $f^{-1} : Z \rightarrow X$ such that $f^{-1} \circ f(x) = x$ for all $x \in X$ and $f \circ f^{-1}(z) = z$ for all $z \in Z$. The function f^{-1} is called the inverse of f .

Using the terminology introduced in Section 1.1, it can be seen that the design analysis when written as $f : D \rightarrow O$ is surjective by definition because O is defined as the image of D under f . In practice, however, the set O is unknown to the designer, and in general the strongest knowable form of the design analysis is $f : D \rightarrow \mathbb{R}^k$, i.e. that the design analysis maps designs to some k -vectors of attribute values. The implication here is that the design analysis is generally not surjective in its given form, but could become surjective if appropriate knowledge about the image of the feasible design space, O , were available.

Likewise, the design analysis is generally not one-to-one. To demonstrate this, consider the case when a vector of attribute values \mathbf{y} is given, and the question is asked: how many designs map to \mathbf{y} according to the design analysis? In this case, we can treat the design analysis as a collection of k functions (one for each attribute) in n unknowns, the design variables. Unlike systems of linear equations, it is not possible to make general statements about whether such a (non-linear) system has no solution, one solution, or an infinite number of solutions. Rather, we simply note that in general there is no reason to expect that this

system will have exactly one solution as would be required for a one-to-one mapping. For example, consider the problem

$$X = \mathbb{R}, Y = \mathbb{R}$$

$$f : \begin{cases} X \rightarrow Y \\ x \Rightarrow y = x^2 + 1 \end{cases}$$

When attempting to solve for x given a value of y , this system has one unknown, x and one equation, $y = x^2 + 1$. Choosing $y = 2$ gives two solutions, $x \in -1, 1$. Choosing $y = 0$ gives no solution. Only choosing $y = 1$ gives exactly one solution, $x = 0$. The implication is that given a general non-linear, multi-dimensional design analysis, it would be exceedingly unlikely that for all vectors $\mathbf{y} \in O$ there is exactly one vector $x \in D$ such that $f(\mathbf{x}) = \mathbf{y}$.

Based on the fact that a general design analysis would not be expected to fulfill either of the two conditions required for invertibility, it seems that there is little hope for developing an inverse design methodology. However, this is not the case. By restricting attention to carefully chosen subsets of the design space and the objective space, the design analysis can be made to be invertible over these subsets.

The present goal then is to find a suitable subset of the feasible design space, denoted by D^* , and a corresponding subset of the objective space, denoted by O^* , such that these subsets have the following properties:

1. The mapping $f : D^* \rightarrow O^*$ is bijective.
2. D^* contains all designs that are potentially optimal.

The second property is, of course, not a requirement for inversion, but simply states that the inversion will serve no useful purpose if D^* does not contain the potentially optimal designs.

Based on the experiments of Chapter 3, it is clear that the Pareto frontier in the objective space and its preimage in the design space typically fulfill both of these requirements. For the case of decision making under certainty being considered in this work, a rational decision maker would choose a non-dominated design over any dominated design. Consequently,

$\mathcal{P}(Y)$ fulfills the second property listed above. ($\mathcal{P}(Y)$ likely also contains many designs that would not be considered good; the important point here is that it contains *all* designs that *are* good.)

It has also been observed in Chapter 3 that the mapping from designs to outcomes on the Pareto frontier is generally one-to-one. The relevant question here is “does there exist more than one \mathbf{x} that yields a particular non-dominated outcome, \mathbf{y}^* ?” Globally, for a non-linear design analysis, no definitive conclusion can be drawn here. Indeed, an easily observed counterexample is the case in which the global Pareto frontier transitions from one local frontier to another. At the intersection point, clearly there are at least two designs that yield the same outcome. For real design problems, the number of such points is likely small enough that they do not actually preclude the frontier from being inverted.²

The more pressing question is whether the one-to-one mapping can fail *locally*, i.e. given a non-dominated outcome, \mathbf{y}^* , and its preimage, \mathbf{x} , does there exist a $\delta\mathbf{x} < \epsilon$ such that $\mathbf{x} + \delta\mathbf{x} = \mathbf{y}^*$? In these cases the inverse mapping would fail to be a function in some neighborhood around \mathbf{x} . Perhaps surprisingly, such cases, which are common on the interior of the objective space, almost certainly do not exist on the Pareto frontier. Recall from Section 3.3.5 that if \mathbf{y}^* is non-dominated, the columns of the Jacobian of f must form a basis for the tangent space to the Pareto frontier. Essentially this requires that not only would $\delta\mathbf{x}$ have to satisfy the condition $\mathbf{x} + \delta\mathbf{x} = \mathbf{y}^*$, but also a condition on matching the Jacobian. The likelihood of this happening in a real design problem by chance is exceedingly small.

Based on this reasoning, a primary thesis of this work is that the set of non-dominated designs fulfills the requirements for the subset D^* defined above. Yet, a problem remains to be solved. Although an invertible subset of the design space and a corresponding subset of the objective space have been found, a useful inverse design analysis cannot be created from just these subsets. The problem is that we have no way to *describe* the outcomes in O^* in

²In practice, the limiting factor is not whether multiple designs yield the same outcome, but whether the changes in the designs implied by motion along the frontier in the objective space are predictable enough to allow an inverse function to be posed. As long as only a few outcomes have multiple corresponding designs, these few cases can be accommodated by a piecewise continuous inverse function with jump discontinuities at the transition points between subfrontiers.

such a way as to construct useful arguments to serve as inputs to this inverse function.

Consider, for example, that the Pareto frontier is sampled as a discrete set of point designs, and the values corresponding to each of these designs and their outcomes have been tabulated. In this case, the “inverse function” would consist simply of looking up values for the outcomes in this table and checking the corresponding values of the design variables. This approach would be almost useless as a knowledge generating tool. This trouble arises because the outcomes that compose O^* are known only as discrete points, with no order or other relationships defined between them. Consequently, given one point in this set, e.g. one row of the above table, its corresponding design may be looked up in the table, but there is no straightforward way to learn about other outcomes or designs in the set that are in some way related.

The resolution to this problem that is proposed here is to parameterize O^* with a coordinate system. By representing the discrete outcomes that compose O^* as points in a metric space, parameterized by a coordinate system, we may then describe relations between the outcomes using intuitive notions of nearness and relative location along each coordinate axis.

The reasoning behind this approach may seem circular. The vectors \mathbf{y} that compose O^* are already represented as points in a metric space parameterized by a coordinate system: the k -dimensional objective space parameterized by the Cartesian coordinate system in which each attribute is mapped to one axis. The difference in the present approach is that the goal is now to construct a coordinate system that parameterizes *only* O^* while excluding the remainder of the objective space, which is either suboptimal, infeasible, or unobtainable.

To achieve this mapping, the new coordinate system must adhere to the simplex-like geometry of the Pareto frontier described in Chapter 3. Consequently, the simplest approach is to use a $(k-1)$ -dimensional barycentric coordinate system, which requires that each tuple of coordinates, $\boldsymbol{\psi}$, obey the constrain $\sum \psi_i = 1$. By design, the mapping from coordinate values to non-dominated outcomes is a bijection. Thus, a function of the form $\mathbf{y} = g(\boldsymbol{\psi})$ can be constructed. As described above, a bijective mapping $\mathbf{x} = h(\mathbf{y})$, $\mathbf{y} \in O^*$ can also be

constructed. In practice this may be as simple as a table lookup that pairs each sampled outcome with its corresponding design. The composition of these two functions yields the values of the design variables associated with each coordinate tuple, $\mathbf{x} = (h \circ g)(\boldsymbol{\psi})$. For simplicity, the composition will henceforth be omitted from the notation and g will be represented as a two-output function of the coordinates: $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$.

4.3 Pareto Simplex Exploration

The overarching approach suggested here – constructing a coordinate system on the Pareto frontier that can be used to form an alternative design analysis of the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ for exploring design problems – is named “Pareto Simplex Exploration.” The coordinate space and its relation to the design and objective spaces by the various mappings is illustrated conceptually in Figure 31.

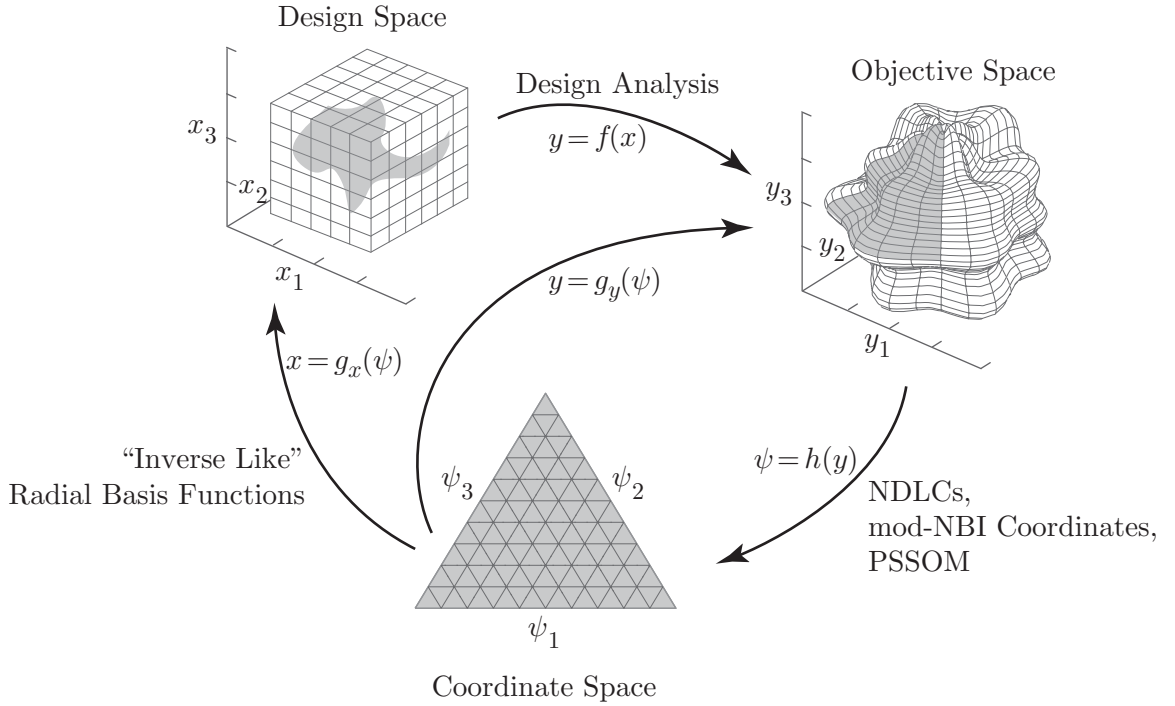


Figure 31: The three spaces relevant to Pareto Simplex Exploration, design space, objective space, and coordinate space, and mappings between them.

In a traditional multi-attribute design problem, only the design analysis mapping, $y = f(x)$, is available, and the analysis proceeds in one direction from the design space to the

objective space. The primary novelty of Pareto Simplex Exploration is the creation of a coordinate system that parameterizes the Pareto frontier. As shown in Figure 31, this coordinate system is then used to construct functions, $g(\psi)$, that calculate the values of the design variables and attributes for all points on the Pareto frontier. Optionally, the coordinates can be used to calculate the design variables and the original design analysis function can then be used to calculate the attributes corresponding to these design variable values (i.e. $y = f(g_x(\psi))$).

Pareto Simplex Exploration entails a sequence of functions that must be fulfilled and leaves it to the user to select the best method(s) to fulfill each function as required by his or her particular design problem. As described above, the most novel of these functions is to form a coordinate system that parameterizes the Pareto frontier, and the primary objective of this dissertation is thus to provide three possible approaches to fulfill this need. Representative methods for fulfilling the remaining functions, which are described below, are implemented throughout Chapters 5–8. However, the particular multi-objective optimization algorithms, interpolating functions, and visualization techniques that are implemented here should not be interpreted as definitive components of the Pareto Simplex Exploration approach, but simply as illustrative examples of the methods that may be used to fulfill each function.

The proposed Pareto Simplex Exploration approach for exploring Pareto frontiers that are parameterized with a coordinate system is shown in Figure 32. The functions corresponding to each step of this approach are described in the subsections that follow.

4.3.1 Step 1: Sample the Pareto Frontier

Each coordinate system is constructed from a discrete sampling of the Pareto frontier, and the first step of Pareto Simplex Exploration is to obtain such a sampling. As described in Section 2.1.2, densely and accurately sampling the Pareto frontier is essentially impossible without using an algorithm that explicitly seeks out non-dominated designs. It is thus presumed that this step will involve the application of a multi-objective optimization algorithm, and the main question to be answered is which algorithm should be used?

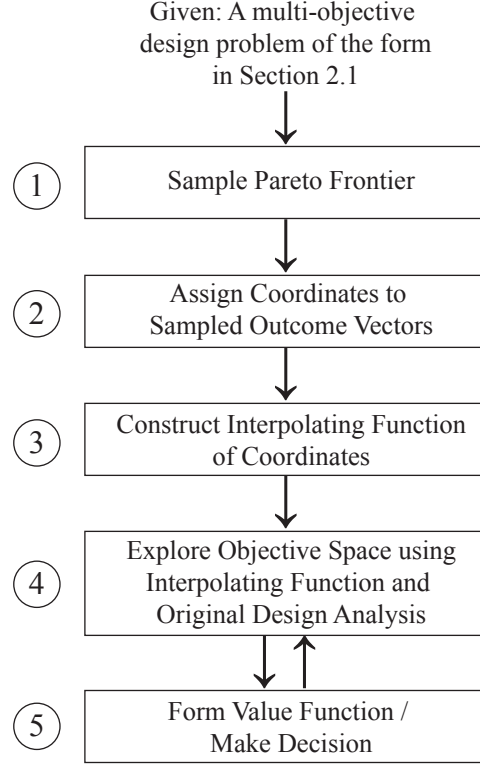


Figure 32: Steps in proposed Pareto Simplex Exploration approach

Two criteria are desirable for sampling the frontier: the sampling should cover the entire frontier and the sampled points should be fairly evenly spread throughout the frontier. Consequently, the multi-objective optimizer must be chosen such that it is able to sample from the entire frontier considering the particular geometry of the design problem being considered. Evolutionary algorithms, such as NSGA-II are able to find globally non-dominated points regardless of the frontier’s geometry; however, these algorithms require many function calls and generally fail to precisely converge to the frontier, as demonstrated in Figure 23. Nevertheless, they are attractive for their flexibility and ease of implementation. Aggregate objective function (AOF) based algorithms may be used if the particular AOF is able to sample from all portions of the Pareto frontier. As shown by Messac et al. [105, 101], this requires that the AOF’s local curvature exceed the (concave) Pareto frontier’s local curvature.

A related question is how many non-dominated points should the sampling comprise? In

Step 3, the sampled points and their associated coordinates will be used as the basis for an interpolating function. The error of interpolation is thus related to the spacing between the sampled points. For fairly linear frontiers, a small sampling may be used, while complicated non-linear frontiers will require a very large sampling. The complexity of the frontier must be considered in the design space as well as the objective space. Consider again the example problem in Figure 23. The appearance of the frontier in the objective space is deceptively simple; this curve could be approximated well with as few as 10 sampled points. However, the preimage in the design space reveals complicated changes in the independent variables and frequent changes in the active constraint set. It is likely that 100 or more sampled points would be needed to accurately model these trends.

In Chapters 5–7, two multi-objective optimization algorithms are considered. The first is NSGA-II, which as noted above generally yields evenly spaced, samplings of the entire Pareto frontier in the objective space but fails to converge well in the design space. The NSGA-II’s crowding distance has been modified in order to more evenly space the sampled outcomes; the details of this modification are described in Section 5.6. The second algorithm is a variant of normal boundary intersection that yields an almost perfectly even sampling and that uses derivative information to converge well in both the objective and design spaces.

4.3.2 Step 2: Assign Coordinates to Sampled Outcome Vectors

The most novel step of Pareto Simplex Exploration is the definition of a coordinate system on the Pareto frontier. The proposed strategy is to define a one-to-one mapping between points on the Pareto frontier and points on a simplex of matching dimensionality. The coordinates of the simplex can then serve as independent variables for exploring the Pareto frontier in the objective space. For this application it is convenient to use a barycentric coordinate system on the simplex, which identifies any point as a weighted sum of its vertices with the weights being the barycentric coordinates. Based on the dimensionality results of Chapter 3, the simplex will have as many vertices, and therefore as many coordinates, as the Pareto frontier has attributes.

The central challenge of constructing a barycentric coordinate system on a Pareto frontier is the choice of a suitable mapping from coordinate tuples to points on the frontier. One possible mapping from a 2-D barycentric coordinate grid to a spherical Pareto frontier is illustrated in Figure 33. The left plot shows the grid of barycentric coordinates, the center plot shows the Pareto frontier as a continuous surface, and the right plot shows the coordinate grid after being mapped to the Pareto frontier surface.

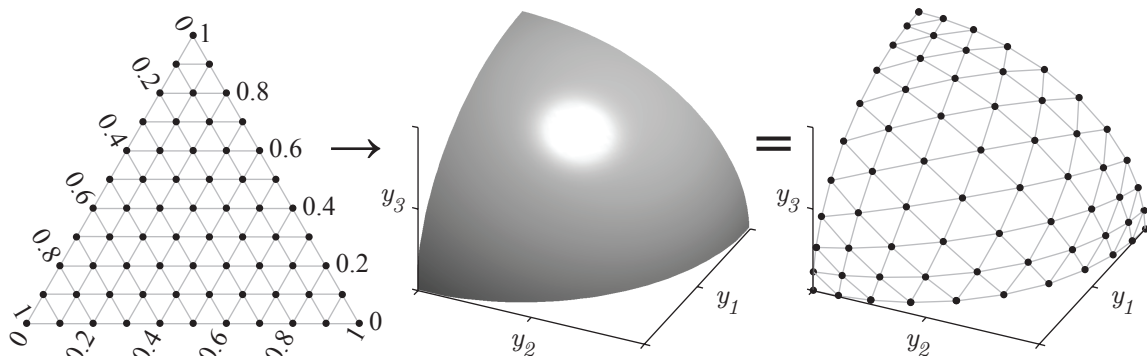


Figure 33: Notional mapping of a barycentric coordinate grid to a spherical Pareto frontier

The extent to which any particular mapping, such as the one in Figure 33, is intuitively meaningful to a decision maker will determine the usefulness of the resulting coordinate system. This assessment of usefulness is necessarily subjective, but some “best practices,” all of which are demonstrated by the mapping in Figure 33, seem indisputable:

1. A k -attribute problem leads to a system with k coordinates. It therefore seems desirable to identify each coordinate with a particular attribute.
2. Each coordinate’s values should be monotonic with its corresponding attribute’s values, such that increasing the coordinate improves the attribute.
3. Consequently, the coordinate tuple whose k -th coordinate is equal to 1 and whose remaining coordinates are then necessarily equal to 0 should correspond to the solution of the single-objective problem $\min y_k$.
4. Extending this idea to higher dimensions, coordinate tuples that have a 0 value for the i -th coordinate should correspond to an outcome on a subfrontier that excludes

the i -th attribute. This idea scales to tuples with any number of 0-valued coordinates.

In the remainder of this dissertation, three possible methods of creating coordinate systems that adhere to these guidelines are presented, and their construction and use for exploring example problems is demonstrated. The three methods are:

- Non-domination level coordinates (NDLCs)– an easily constructed, but locally noisy, coordinate system that can be created from discrete sampling of a Pareto frontier.
- mod-NBI coordinates – a coordinate system based on the weights that parameterize subproblems in a modified normal boundary intersection (mod-NBI) algorithm. Unlike the other two coordinate systems, this approach relies on a particular method of sampling the frontier.
- Pareto simplex self-organizing map (PSSOM) coordinates – a type of self-organizing map that is tailored to conforming a coordinate system to the Pareto frontier. This method generally yields the best coordinate system and is insensitive to unevenness in the sampled frontier, but it is the most difficult of the three approaches to implement.

4.3.3 Step 3: Construct Interpolant for Coordinates

The previous step identifies a coordinate tuple for each point in the sampled Pareto frontier. One goal of Pareto Simplex Exploration is to enable exploration that is independent of the particular outcomes sampled in Step 1. This is enabled by constructing an interpolating function of the coordinates so that the existence of intermediate points on the Pareto frontier (i.e. outcomes of designs not sampled in Step 1) can be inferred.

The approach used here is to model each design variable and attribute as a radial basis function of the coordinates. Together, these individual functions constitute the inverse-like design analysis, $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$. Other types of interpolating functions could also be used, and may be preferred in particular cases. Regression-based models, such as response surface equations, are unsuitable in most cases because the design variables do not vary smoothly in regions of the frontier where the active constraint set changes.

4.3.4 Step 4-5: Explore Objective Space using Sampled Data and Interpolated Coordinates and Form Value Function / Make Decision

Steps 4 and 5 of Pareto Simplex Exploration are intended to be applied iteratively. Step 4 entails applying the new exploration methods that are enabled by the interpolating functions created in Step 3, along with existing methods based on the original design analysis, to gather knowledge about the decision. Step 5 entails ranking the alternatives to make a decision, either by defining a value function or by directly ordering the alternatives according to preferences.

Several example exploration techniques that may be used in Step 4 are presented in Chapter 8. Step 5 is not directly addressed in this work, as the suitability of a particular value function and the ranking of alternatives is necessarily subjective.

4.4 Multi-Objective Optimization Test Problems

The test problems that will be used to demonstrate the three methods for defining a coordinate system on the Pareto frontier described in Chapters 5–7 are listed in Table 6, and their Pareto frontiers are plotted in Figure 34. Each of the example problems considered here has a Pareto frontier that exhibits some behavior found in real optimization problems. The optimization problem statements for each example are listed in Table 6. To aid in visualization, each of these example problems has three attributes; however, the methods described in Chapters 5–7 are applicable to design problems with arbitrary numbers of attributes.

Table 6: Problem Statements for Exemplar Pareto Frontiers

Name	Variables	Objectives/Attributes	Constraints
Sphere	$0 \leq x_i \leq 1$	$\max \begin{pmatrix} y_1=x_1 \\ y_2=x_2 \\ y_3=x_3 \end{pmatrix}$	$\sqrt{y_1^2 + y_2^2 + y_3^2} < 1$
Truncated Sphere	$0 \leq x_i \leq 1$	$\max \begin{pmatrix} y_1=x_1 \\ y_2=x_2 \\ y_3=x_3 \end{pmatrix}$	$\sqrt{y_1^2 + y_2^2 + y_3^2} < 1$ $y_3 < 0.8$
Concave	$0 \leq x_i \leq 1$	$\max \begin{pmatrix} y_1=x_1 \\ y_2=x_2 \\ y_3=x_3 \end{pmatrix}$	$\sqrt{y_1^{0.5} + y_2^{0.5} + y_3^{0.5}} < 1$
Teardrop	$0 \leq x_i \leq 1$	$\max \begin{pmatrix} y_1=x_1 \\ y_2=x_2 \\ y_3=x_3 \end{pmatrix}$	$\sqrt{y_1^{0.5} + y_2^{0.5}} - y_3^4 + 2y_3^2 < 1$
Bumpy Sphere	$0 \leq \phi \leq \pi/2$ $0 \leq \theta \leq \pi/2$	$\max \begin{pmatrix} y_1=r \cos(\phi) \sin(\theta) \\ y_2=r \sin(\phi) \sin(\theta) \\ y_3=r \cos(\theta) \end{pmatrix}$ $r=1 + \frac{1}{20} \cos(10\theta) \sin(10\phi)$	no constraints

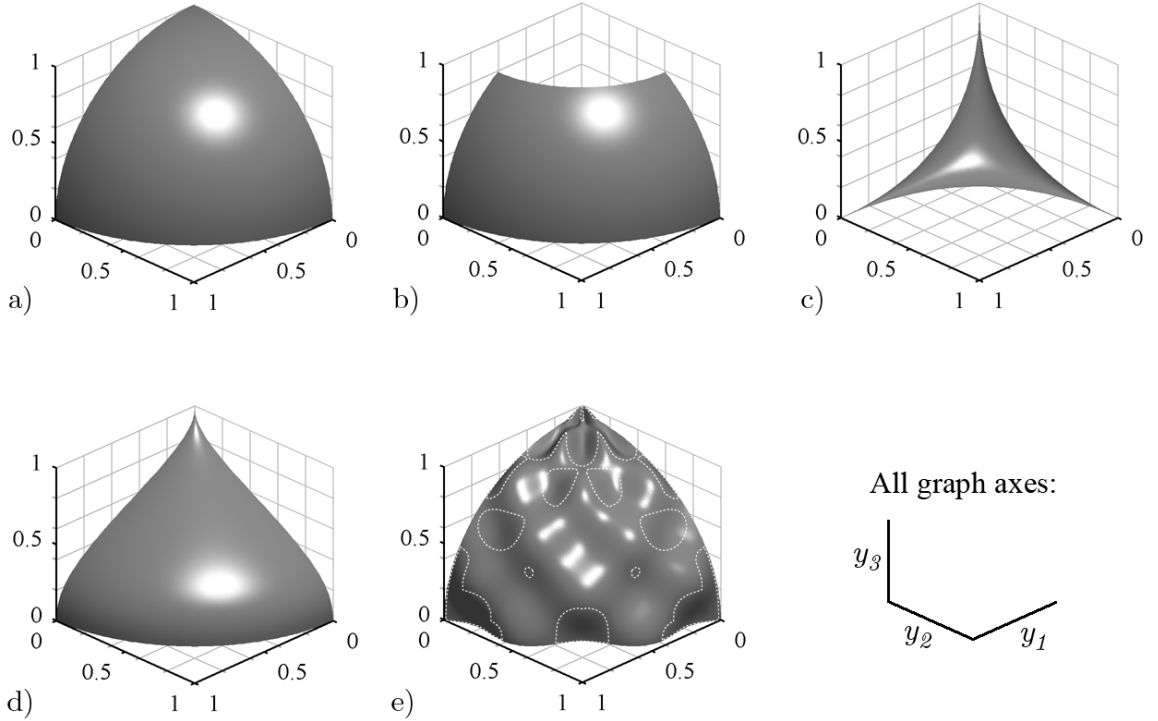


Figure 34: Example Pareto frontier shapes: a) Sphere, b) Truncated Sphere, c) Concave, d) Teardrop, e) Bumpy Sphere (dashed lines enclose dominated regions)

CHAPTER V

A NON-DOMINATION LEVEL COORDINATE SYSTEM

This chapter describes a coordinate system for the Pareto frontier based on the concept of *non-domination levels*, which were introduced by Deb [38, 39] for partially ordering a set of points in the objective space according to their relative dominance.¹ The general approach here is to calculate a “robust” version of non-domination levels on a sampled frontier that is projected into a subset of $k - 1$ of the attributes. The resulting levels form the basis for a coordinate that describes relative optimality in the attribute that was left out. The full coordinate system is constructed by repeating this process k times, leaving each attribute out in turn.

These “non-domination level coordinates” (NDLCs) [37] can be created from any sufficiently dense and evenly spaced sampling of a Pareto frontier. Consequently NDLCs can be constructed as a post processing step to any multi-objective optimization algorithm.

As will be seen in the results section, the non-domination level coordinate system is not perfect. It captures the large-scale geometry of the frontier, but is prone to local “noise.” Subsequent chapters describe two alternative methods of constructing coordinate systems, one based on a modified normal boundary intersection algorithm and one based on self-organizing maps, that generally yield smoother coordinate systems than the non-domination level coordinates. However, the NDLCs remain interesting because of its conceptual simplicity, ease of creation, and because of the conceptual connection it establishes between the non-domination levels of a projected Pareto frontier and relative optimality in the projected dimension. The particular shortcomings of NDLCs also serve to motivate the modified NBI and self-organizing map based approaches to constructing a coordinate system.

Source code for the methods introduced in this chapter is included in Appendix B. The reader may find it helpful to refer to the source code to clarify the steps of the process for

¹Pareto dominance is defined in Section 1.1.1

creating the non-domination level coordinates.

5.1 *Non-Domination Levels*

The *non-domination level* of a point in the objective space is a calculated value that extends the binary concept of dominated vs. non-dominated to a more informative set of integer levels that describe *how dominated* the point is relative to a set of such points. Roughly speaking, a point with a lower non-domination level is less dominated, i.e. closer to being Pareto optimal, than a point with a higher non-domination level.

The procedure for calculating non-domination level coordinates of a set of points, P , in the objective space is described in Algorithm 1.² First, all points in P that are non-dominated are assigned to level 0. These points are then temporarily removed from P , and the non-dominated points in the remaining subset are assigned to level 1. This process is repeated until all points in P have been assigned to a level [39].

In order to use non-domination levels as the basis for a coordinate system that describes relative optimality, these integer levels are normalized to range from 0–1 by simply dividing the calculated integer levels by the maximum level in P . This is necessary because the integer levels describe absolute, rather than relative, dominance. For example, a point with level 5 could be extremely dominated relative to the other points in P , if the maximum level is 5 or slightly higher, or relatively optimal, if the maximum level is much higher than 5. In the remainder of this chapter, “non-domination levels” always refers to the normalized decimal levels.

5.2 *Non-Domination Levels as the Basis for a Coordinate System*

The inspiration for using non-domination levels as the basis for a coordinate system comes from an observation about the relation between the iso-coordinate lines of the notional coordinate system in Figure 33, which is reproduced as Figure 35a, and the non-domination levels of an underlying sampled Pareto frontier. Suppose that the notional Pareto frontier were sampled according to the black points plotted in Figure 35a. Consider the subset of

²A MATLAB implementation of Algorithm 1 can be found in Appendix B.

Algorithm 1 Non-Domination Level Assignment (adapted from Deb [39])

```
1: function NONDOMINATIONLEVELASSIGNMENT( $P$ )
2:    $i = 0$ 
3:   while  $P \neq \emptyset$  do
4:      $q \leftarrow \text{NonDominatedSubset}(P)$ 
5:      $\text{integerLevels}(q) \leftarrow i$ 
6:      $P \leftarrow P \setminus q$   $\triangleright$  Remove subset  $q$  from  $P$ 
7:      $i \leftarrow i + 1$ 
8:   end while
9:    $\text{decimalLevels} = \text{integerLevels} / \max(\text{integerLevels})$ 
10:  return  $\text{decimalLevels}$ 
11: end function
```

these points in Figure 35b that are connected by the iso-coordinate curve $\alpha_1 = 0$. These points all have the property that they are non-dominated with respect to the subset of the attributes $\{y_2, y_3\}$, i.e. they are on the subfrontier $\mathcal{P}(\{y_2, y_3\})$. Calculating the non-domination levels of the black points based only on attributes $\{y_2, y_3\}$ extends this result to the remaining iso-coordinate curves: the curves of constant non-domination level are identical to the iso-coordinate curves plotted in Figure 35b. Repeating this process for non-domination with respect to $\{y_1, y_3\}$ and $\{y_1, y_2\}$ yields the curves plotted in Figure 35c and d respectively.

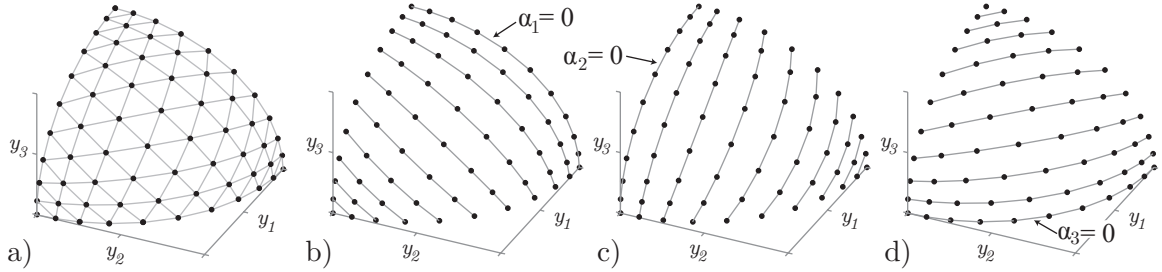


Figure 35: Coordinate curves of a Pareto frontier as defined by the mapping in Figure 33. a) All coordinates b) α_1 coordinates c) α_2 coordinates d) α_3 coordinates

This correspondence between the isocoordinate curves and the non-domination levels is not coincidental. The points on non-domination level 0 with respect to $\{y_2, y_3\}$ in Figure 35b are, by definition, the points that would be desirable to a decision maker who is indifferent to y_1 . Indifference in this context means that the decision maker would be unwilling to sacrifice optimality in y_2 and/or y_3 to improve y_1 . Points with higher non-domination levels

are less optimal in y_2 and y_3 , and more optimal in y_1 , such that increasing non-domination level with respect to $\{y_2, y_3\}$ corresponds to increasing preference for optimality in y_1 at the expense of y_2 and y_3 .

This notional example illustrates the general idea behind the non-domination level coordinate system. Constructing NDLCs is conceptually simple, easily implemented, and fast. The remainder of this chapter describes the generalization of this process to arbitrary, possibly irregularly sampled, Pareto frontiers.

5.3 Robust Non-Domination Levels

The starting point for constructing the non-domination level coordinate system is the set of sampled points in the objective space that compose the sampled Pareto frontier. This set is presumed to have been obtained as the output of a multi-objective optimization algorithm that is capable of finding all non-dominated points, such as NSGA-II, and to be large enough and sufficiently well converged to be representative of the true global Pareto frontier. By definition, each of these points belongs to non-domination level 0 with respect to all k attributes.

The conceptual premise behind NDLCs is to calculate the non-domination levels of each sampled point with respect to the *subsets of attributes* taken $k - 1$ at a time. There are k such subsets, so each point has an associated k -tuple of non-domination levels. These form the conceptual basis for a coordinate system; however, as will be described below, the final coordinate system is based on a normalized “robust” variation on the non-domination levels.

As an example, the three attribute problem $\max(y_1, y_2, y_3)$ would entail ranking the sampled frontier (i.e. calculating each point’s non-domination levels) with respect to the subproblems $\max(y_2, y_3)$, $\max(y_1, y_3)$, and $\max(y_1, y_2)$. The computational cost of this ranking is generally negligible compared to the cost of generating the sampled frontier, thus the use of NDLCs is not limited by computational feasibility.

As the basis for a coordinate system, the non-domination levels have two shortcomings that must be addressed. First, the non-domination levels are calculated with respect to the

sampled frontier and therefore are strongly dependent on the distribution of these points. Local clustering or an absence of sampled points in some region of the frontier will reduce the meaningfulness of the non-domination levels. Second, a requirement of a barycentric coordinate system is that the coordinates of each point sum to one, and it is not guaranteed that each point's k non-domination levels – one for each $(k-1)$ -attribute subproblem – will do so.

The first of these shortcomings arises because the non-domination levels define a partial ordering on the points that is based on the *number* of other points that dominate a given point without regard to the *distances* between those points.

The example in Figure 36 shows the non-domination levels of two point distributions. The population in Figure 36b differs from that in Figure 36a only by the addition of two points between levels zero and one in Figure 36a. The addition of these two points causes the iso-level lines to bulge outward and to condense in the densely sampled region. Similarly, if a particular region were sparsely sampled compared to its surroundings the iso-level lines would bulge inward.

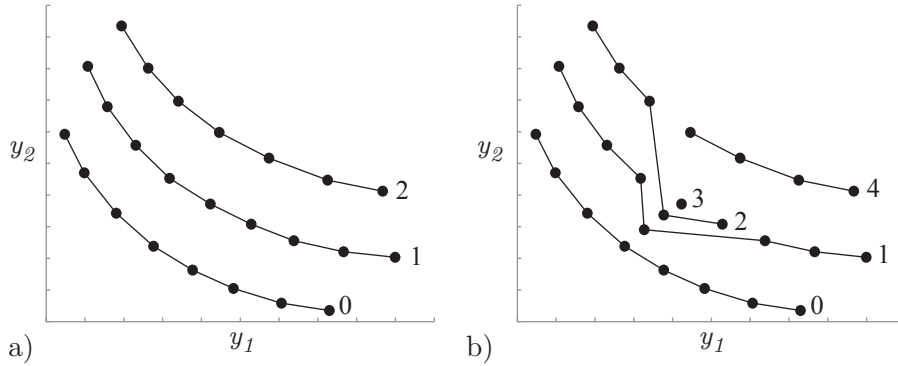


Figure 36: A sampled population, with points of the same non-domination level connected with lines. a) Baseline population b) Similar population with very different non-domination level curves (c.f. Figure 38a)

Any multi-objective optimizer in which the spacing between points in the sampling involves an element of randomness may exhibit clustering, and as Figure 36 suggests, even a single small cluster can severely disrupt the smoothness of the non-domination levels. To alleviate this problem, we have developed a modified “robust” non-domination level

calculation based on a resampling technique, which is outlined in Algorithm2. The approach in Algorithm 2 is to randomly select a small subset of the population and calculate its non-domination levels using Algorithm 1, ignoring the points that are not included in this subset. This process is repeated many times, and each time the levels calculated for each point in the subset are added to the levels calculated in previous repetitions. The final “robust” non-domination levels are calculated by dividing each point’s summed levels by the number of times that point was included in the subset. The resulting robust levels can thus be thought of as an average non-domination level when the Pareto frontier is resampled with a relatively small sample size.

Algorithm 2 Robust Non-domination Level Assignment

```

1: function ROBUSTNONDOMINATIONLEVELASSIGNMENT( $P, samplingFraction, repetitions$ )
2:   for all  $p \in P$  do
3:      $count(p) \leftarrow 0$ 
4:      $sumOfLevels(p) \leftarrow 0$ 
5:   end for
6:   for  $r = 1$  to  $repetitions$  do
7:      $S \leftarrow \emptyset$ 
8:     for all  $p \in P$  do
9:       if  $UniformRandom < samplingFraction$  then
10:         $count(p) \leftarrow count(p) + 1$ 
11:         $S \leftarrow S \cup \{p\}$ 
12:       end if
13:     end for
14:      $subsetLevel \leftarrow NondominationLevelAssignment(S)$ 
15:     for all  $p \in S$  do
16:        $sumOfLevels(p) \leftarrow sumOfLevels(p) + subsetLevel(p)$ 
17:     end for
18:   end for
19:   for all  $p \in P$  do
20:      $robustLevel(p) \leftarrow sumOfLevels(p) / count(p)$ 
21:   end for
22:   return  $robustLevel$ 
23: end function

```

Algorithm 2 requires two user-specified parameters: the sampling fraction and the number of repetitions. The sampling fraction is the fraction of the points from the original Pareto set that is sampled to define each subset. As implemented in Algorithm 2, the sampling fraction is simply the probability that any given point is included in the subset;

hence the precise subset size may vary between repetitions. The sampling fraction should be small enough that local clusters or voids in the original set are smoothed out by the sparsely sampled subsets but large enough that the subsets retain the global character of the Pareto frontier. The sampling fraction may be thought of as controlling the variability of the levels assigned to each point as it is resampled. As the sampling fraction is increased, the robust non-domination levels approach the (deterministic) non-domination levels, and in the limit of a sampling fraction of 1, Algorithms 1 and 2 yield the same result. At the opposite extreme, very low sampling fractions yield subsets that contain few non-domination levels – a subset of size k can have at most k different non-domination levels and likely will have far fewer than k levels. In these cases, the variability in the levels calculated for each point may be so high that a meaningful average value cannot be obtained.

A balance between these two extremes must be found such that the sampling fraction is small enough that the algorithm does not simply reproduce the deterministic levels, which would be calculated by Algorithm 1, but large enough that the variance of levels assigned to each vector is small. Through experimenting, it has found that sampling fractions near 0.05 are generally effective when the sampled Pareto frontier contains significant clustering or irregular spacing, and fractions as high as 0.75 may be used when the sampled frontier is evenly spaced or when the population is so small that a large fraction is needed to ensure the sampled subsets contain enough points to allow meaningful non-domination levels to be calculated.

The number of repetitions is more easily chosen, as the only tradeoff is between time and convergence. A large number of repetitions ensures that the final robust levels are well converged. Because the computational complexity of Algorithm 2 is generally much less than that of the multi-objective optimizer needed to sample the Pareto frontier, very large numbers of repetitions can be used with little incentive to use few repetitions. Algorithm 2 works well when the sampling fraction and the repetitions are chosen so that each point is sampled approximately 100 to 1000 times – values that are practicable and sufficient in most cases.

To calculate the final non-domination level coordinates, first the robust non-domination

levels are calculated over each $(k - 1)$ -subset of the attributes, so that k robust levels are calculated for each point. Each point's non-domination level coordinates are equal to its robust non-domination levels divided by their sum. This normalization forces each point's coordinates to sum to 1 as required for barycentric coordinates.

5.4 *Optional Preprocessing*

As an optional pre-processing step to improve the mapping of the non-domination level coordinates to the Pareto frontier, the Pareto frontier points may be projected onto the hyperplane that contains the individual optima before the robust non-domination levels are calculated. The projected points are used only for calculating the robust non-domination levels associated with each point; subsequent explorations are conducted using the original data. Projecting the points onto the indicated hyperplane before calculating the robust non-domination levels generally results in more uniformly distributed coordinates across the Pareto frontier so that the mapping from coordinates to non-dominated points is more linear. However, the desirability of this result is subjective and so the projection is considered to be an optional step.

Figure 37 shows an example of the effect of projecting points onto the hyperplane before calculating the levels. The black points represent the original Pareto frontier sampling for $\max(y_1, y_2)$. The calculation of the non-domination levels for the subproblem $\max(y_1)$ are based only on the y_1 values, which are distributed as indicated by the white points. When the points are not projected onto the individual-optima hyperplane (in this case the line connecting the points (0,1) and (1,0)), the y_1 values are distributed irregularly as shown in Figure 37a. The clustering of the points near $y_1=1$ is particularly problematic for correctly assigning non-domination levels, especially in cases in which the real data set is noisy because of incomplete convergence of the original multi-objective optimizer. When the data are first projected onto the line connecting $\max(y_1)$ and $\max(y_2)$, the distribution of y_1 becomes more uniform as indicated by the white points in Figure 37b. As indicated by this example, we have found that projecting the points is especially helpful in regions of the Pareto frontier where the tradeoffs are such that one attribute must be severely sacrificed

for just a modest improvement in the other attributes.

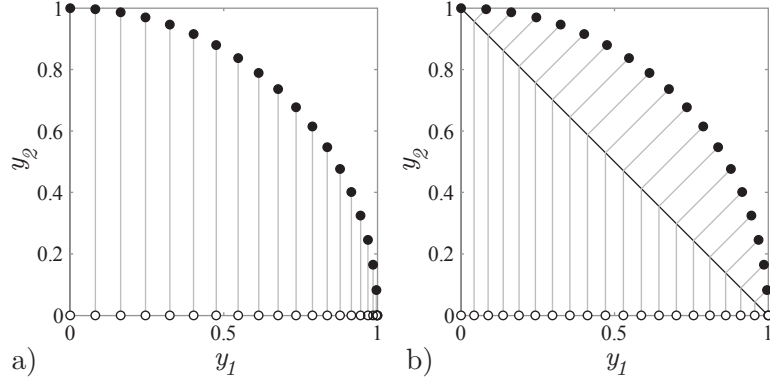


Figure 37: Comparison of point distribution a) without hyperplane projection and b) with projection

5.5 Example Coordinate Calculations

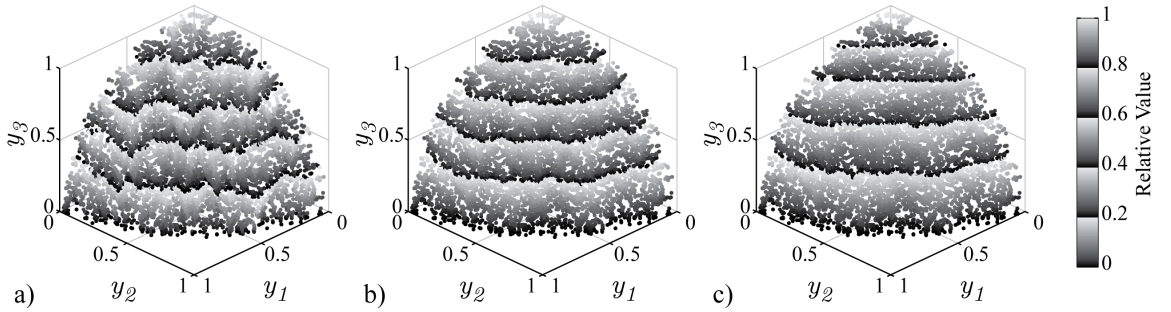
An example of the coordinate calculations for one point on a 3-attribute Pareto frontier is shown in Table 7. The upper part of the table shows the non-domination levels calculated for this point for each of the three subfrontiers, expressed as ranging from 0 to 1 as described in Section 5.1. These values represent the *subsetLevel* calculated in line 14 of Algorithm 2. Missing entries indicate that this point was not included in the sampled subset during the specified repetition. The average of the normalized levels is shown in the second-to-the-last line; these are the robust non-domination levels as calculated in line 20 of Algorithm 2. The bottom line of the table lists the final non-domination level coordinates, obtained by dividing the three robust non-domination levels by their sum.

A comparison of non-domination levels, robust non-domination levels, and non-domination level coordinates is shown in Figure 38 for the sphere-constrained optimization problem defined in Table 6. Figure 38 shows a 10,000 point NSGA-II sampling shaded according to (a) non-domination levels, (b) robust non-domination levels, and (c) non-domination level coordinates for the levels of non-domination of $\max(y_1, y_2)$, which corresponds to the y_3 coordinate. A banded color map is used in Figure 38 to more clearly show small changes in the values. The lowest levels are located toward the bottom of the figure, progressing to level/coordinate 1 at the top of the Pareto frontier. Comparing Figure 38a and

Table 7: Example coordinate calculations for a point on a 3-attribute Pareto frontier

Repetition	Levels calculated with respect to:		
	$\min(y_2, y_3)$	$\min(y_1, y_3)$	$\min(y_1, y_2)$
1	0.85	-	0.09
2	-	0.18	0.11
3	0.78	-	0.09
4	-	0.17	-
5	0.82	0.21	0.13
\vdots	\vdots	\vdots	\vdots
n	0.77	-	0.12
Average	0.825	0.194	0.117
Normalized	0.726	0.171	0.103

b), the benefit of the robust non-domination ranking is immediately clear. The original non-domination levels demonstrate the jaggedness seen in Figure 36b over nearly the entire Pareto frontier. By comparison, the robust levels in Figure 38b are relatively smooth. The final non-domination level coordinates in Figure 38c are similarly smooth and ensure that each point’s coordinates sum to one.

**Figure 38:** Sampled Pareto frontier colored according to: a) Non-domination levels b) Robust non-domination levels c) Non-domination level coordinates

5.6 Application to Example Problems

Each example problem was sampled using the NSGA-II algorithm [39] with 5,000 population members, 500 generations, and mutation and crossover rates of 0.8. The independent variables were discretized using 16-bit (per variable) Gray code. NSGA-II uses the population’s non-domination levels (calculated in the full k -dimensional objective space) as its primary objective function, with ties being settled by a metric called “crowding distance,”

which is a measure of how isolated each point in the population is. The original crowding distance metric proposed by Deb et al. [39], however, performs poorly for problems with more than two attributes. For the present application, each point’s crowding distance is instead equated with its Euclidean distance to the third nearest point on the same non-domination level.³ The use of the third nearest point has been found to give a more even spread than the distance to the nearest point, likely because the mutation step often leads to pairs of very close points.

Additionally, as recommended in Section 5.4, the sampled points were projected onto the plane spanning the individual optima for the calculation of the robust non-domination levels. The robust non-domination levels were calculated using a 0.05 sampling fraction with 10,000 repetitions.

The final NSGA-II populations, colored according to the NDLCs, are plotted in Figure 39, which also shows the corresponding bounding surfaces of the objective space for comparison. For all examples except the bumpy sphere, the Pareto frontier is the entire surface; the Pareto frontier of the bumpy sphere excludes some portions of the surface that are sufficiently concave as to be dominated. The second, third, and fourth columns of Figure 39 are colored according to the non-domination level coordinates of y_1 , y_2 , and y_3 respectively. In almost all cases, the coordinates behave as expected, varying from 1 at the individual optima to 0 at the opposing subfrontier.

One exception is noteworthy. In the case of the truncated sphere, which is not a regular frontier as defined in Section 1.1.3, the points for which the constraint $y_3 < 0.8$ is active have irregular coordinates. Each unique coordinate value along this constraint corresponds to two points in the objective space, mirrored on either side of the frontier. This behavior is the result of projecting the frontier onto the plane of the individual minima before calculating the robust non-domination levels; if projection is not used then each point along the constraint will have identical coordinates. Even for this irregular frontier, however, points with y_3 values less than about 0.6 have well behaved coordinates that are comparable to the

³Generally the entire NSGA-II population converges to being on non-domination level 0 within fifty generations, so the restriction of the crowding distance comparison to only those points on the same non-domination level is relevant only early on.

coordinates of the sphere in Figure 39a. The implication is that even when a Pareto frontier is irregular, non-domination level coordinates generally remain meaningful in regions that are far from the irregularities.

The main drawback that can be observed in Figure 39 is that the NDLCs, despite capturing the large scale trends implied by the various Pareto frontier geometries, are locally noisy. This is most easily observed along the iso-coordinate curves for $\psi \in \{0.2, 0.4, 0.6, 0.8\}$ where the banded colormap resets. The jaggedness in these curves is due to the irregular sampling of the NSGA-II population. As such, this is less a shortcoming of the NDLCs than of the NSGA-II algorithm that generated the sampled frontier.

In the next chapter, a modified normal boundary intersection algorithm for sampling the frontier more evenly is introduced. When NDLCs are calculated on frontiers sampled using this algorithm, the results⁴ are found to be significantly better than the NDLCs based on NSGA-II sampling. However, the modified NBI method has its own, simpler coordinate system in the form of weights used to parameterize its subproblems.

⁴These results are presented in Figure 51 on page 136.

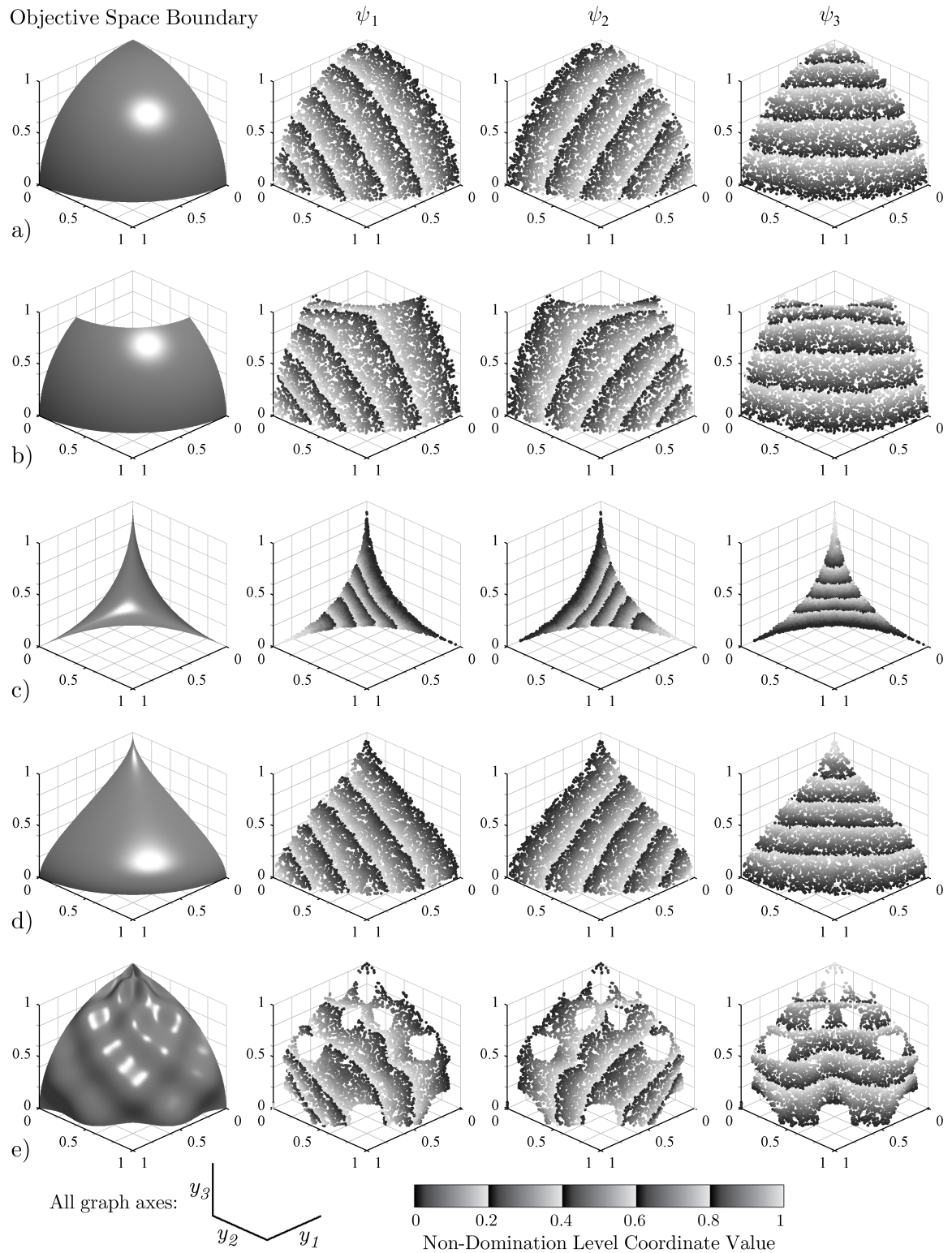


Figure 39: Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding mod-NBI weights. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

CHAPTER VI

A MODIFIED NORMAL-BOUNDARY INTERSECTION COORDINATE SYSTEM

Many multi-objective optimization methods use weights, whose values obey the barycentric-coordinate constraint $\sum w_i = 1$, to formulate an aggregate objective function. A natural approach for forming a coordinate system to parameterize the frontier would be simply to assign each point sampled with such a method its corresponding weights as coordinates. However, as discussed in Section 2.1.2, nearly every weighted multi-objective optimization approach either fails to provide a one-to-one mapping between its weights and points on the frontier or fails to sample the frontier evenly enough for the weights to serve as meaningful coordinates.

One method, however, seems to have great potential: a modified version of normal boundary intersection (NBI) by Mueller-Gritschneider et al. [113]. The original NBI formulation by Das and Dennis [33] provides an even sampling of the frontier, but has the shortcoming of being unable to sample from regions near the boundary of the frontier for problems with more than two attributes. The work by Mueller-Gritschneider et al. adaptively extends NBI's search space to ensure that the entire frontier is evenly sampled while still ordering the points relative to each other in a way that allows the weights to serve as meaningful coordinates.

This chapter describes the original NBI algorithm, the improvements made by Mueller-Gritschneider et al., and two additional modifications to their algorithm that result in a more even sampling of the frontier. The end result is a method for evenly sampling from Pareto frontiers that has a built-in coordinate system in the form of the weights.

MATLAB source code for the methods considered in this chapter is provided in Appendix C. The reader may find it useful to consult the appendix to clarify the concepts introduced in this chapter. The source code is particularly instrumental in illustrating the

simple relations between the methods introduced here – each differing by only a few lines of code – which yield very different samplings of the Pareto frontier.

6.1 Introduction to Normal Boundary Intersection

Normal boundary intersection is a multi-objective optimization method developed by Das and Dennis [33] in the 1990s. The method produces a fairly even sampling of Pareto frontiers in any number of attributes, but it generally cannot sample from some portions of the frontier. Like most weight-based approaches to multi-objective optimization, NBI is not an optimization algorithm *per se*; rather, it is a method for defining a series of single-objective optimization subproblems whose solutions constitute a sampling of the Pareto frontier.

The description of NBI that follows refers to three types of points, which are defined here and illustrated conceptually in Figure 40. Throughout this chapter, the subscript j is used to reference the j -th NBI subproblem, and the subscript i is used to denote either the i -th element of a vector or to distinguish a different subproblem than j depending on the context. Each subproblem involves one point from each of the following sets:

- The weights, \mathbf{w}_j , are k -tuples that obey the constraint $\sum_{i=1}^k w_{j,i} = 1$.
- The basepoints, \mathbf{y}_j^0 , are corresponding points in the objective space. The j -th subproblem is constrained to search along a line through \mathbf{y}_j^0 .
- The sampled points, \mathbf{y}_j , are the solutions to the optimization subproblems. The final set of sampled points constitutes the solution to the NBI method.

The first step of NBI is to construct a grid of weights $\{\mathbf{w}_1, \dots, \mathbf{w}_m\}$. This grid is constructed by selecting an integer number of levels, n , and then constructing all possible k -tuples whose values, $w_{j,i}$, are taken from the set $\{0, 1/(n-1), 2/(n-1), \dots, 1\}$ and are constrained by $\sum_{i=1}^k w_i = 1$. In practice, an algorithm to construct the weight grid generally requires $k-1$ nested loops; however, high-level programming languages enable simpler implementations. An example MATLAB implementation is given in Appendix C.

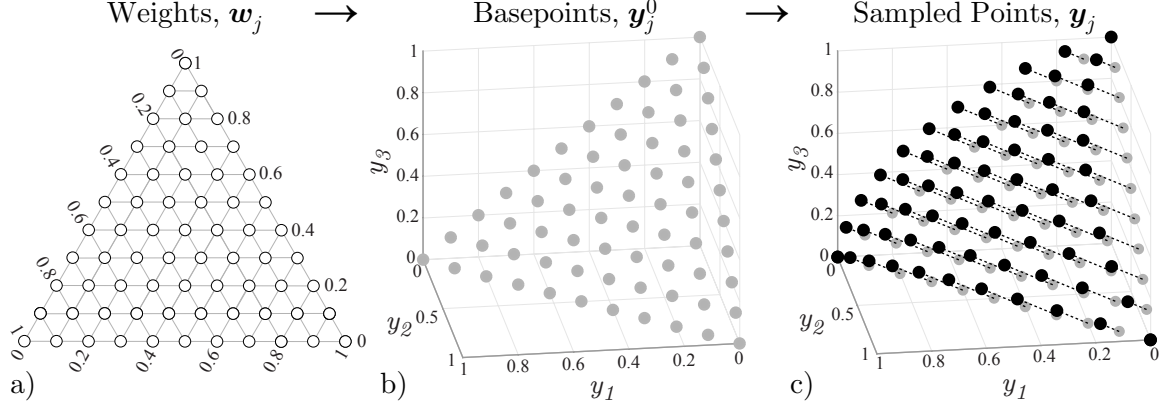


Figure 40: Major steps of normal boundary intersection a) construct grid of weights, b) construct basepoints, c) sample objective space boundary along quasi-normal directions from basepoints

The resulting weight tuples may be represented, as in Figure 40a for $n = 11$, as a uniform grid on a simplex with n tuples per edge. The total number of tuples is $m = \binom{k+n-2}{n}$, each of which leads to a corresponding optimization subproblem to sample a single point on the boundary of the objective space. The number of levels must therefore be carefully chosen to yield the desired frontier sampling resolution while obeying feasibility constraints on the number of subproblems that can be solved.

Next, the weights are used to calculate the corresponding grid of basepoints in the objective space. Each basepoint is a weighted sum of the *individual minima* – the k points in the objective space that are the solutions to the k single-objective problems

$$\begin{aligned} \min_{\mathbf{x}} \quad & y_k \\ \text{s. t.} \quad & \mathbf{x} \in D \end{aligned} \tag{6}$$

The individual minima, \mathbf{y}_k^* that are the solutions to (6) are used to define the utopia point and the matrix Φ , which are used to calculate the basepoints and form the NBI subproblems. The utopia point, \mathbf{u}^+ , is the k -vector whose i -th entry is the best value of the i -th attribute observed in the individual minima $\{\mathbf{y}_1^*, \dots, \mathbf{y}_k^*\}$. Φ is the matrix whose columns are the individual minima shifted by the utopia point: $\Phi = [\mathbf{y}_1^* - \mathbf{u}^+, \dots, \mathbf{y}_k^* - \mathbf{u}^+]$. By definition, the diagonal entries of Φ are 0 and the remaining entries are nonnegative.

Each basepoint is calculated as a linear combination of the columns of Φ using the

corresponding weights as coefficients: $\mathbf{y}_j^0 = \Phi \mathbf{w}_j$. Thus each basepoint lies on or in the simplex in the objective space whose vertices are the individual minima, which is called the “convex hull of individual minima” (CHIM) by Das and Dennis [33]. The quasi-normal search direction is defined as an equally weighted linear combination of the columns of Φ , multiplied by -1 so that it points toward the origin $\hat{\mathbf{n}} = -\Phi[1, \dots, 1]^\top$ [33].

The NBI optimization subproblems are then given by

$$\begin{aligned} & \max_{\mathbf{x}, t} t \\ & \text{s. t. } \mathbf{y}_j^0 + t\hat{\mathbf{n}} = f(\mathbf{x}) - \mathbf{u}^+ \\ & \mathbf{x} \in D \end{aligned} \tag{7}$$

The first constraint in the subproblem defines the central mechanism of NBI: a search along the quasi-normal direction emanating from each basepoint. The first term in this constraint is the location of the j -th basepoint in the objective space; note that for any subproblem, this term is a constant. The second term is the distance, t , along the quasi-normal search direction, $\hat{\mathbf{n}}$, from this basepoint. The right hand side of the constraint requires that the resulting point be the solution to $f(\mathbf{x})$ for some \mathbf{x} , i.e. that this point has a corresponding design, which the second constraint requires be feasible. The final $-\mathbf{u}^+$ term on the right hand side is effectively a simple shift of the origin.

A notional NBI problem of minimizing two attributes is illustrated in Figure 41. Ten basepoints are distributed along the CHIM, indicated by the dashed line, and the search proceeds from these basepoints along quasinormal vectors, indicated by the dotted lines, that are parallel to the line connecting the utopia and anti-utopia points. Note that, despite the name “normal boundary intersection,” the search direction is generally not normal to the CHIM. The black curve indicates the underlying continuous Pareto frontier, and the black points are the sampled frontier found by solving the NBI subproblems.

Several important observations about the general behavior of NBI can be drawn from the example in Figure 41:

1. The region of the objective space that was searched for the Pareto frontier is essentially a translation of the CHIM (dashed line) along the search direction (dotted lines).

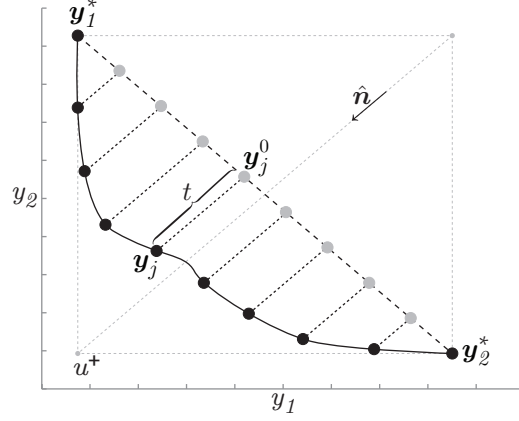


Figure 41: Normal boundary intersection for two attributes

For two-attribute problems, this approach is always able to search the entire Pareto frontier. However, with more than two attributes, NBI generally cannot search the entire Pareto frontier.

2. Although each black point in this example is globally non-dominated, in general NBI seeks out points *on the boundary of the objective space* without any guarantee of global Pareto optimality. If portions of the boundary are sufficiently concave as to be dominated, NBI will still sample from those portions.
3. The distribution of the basepoints on the CHIM is perfectly even, but because of curvature in the Pareto frontier, the distances between the corresponding sampled points vary. The variation is typically small, but it may be large in regions where the tangent to the Pareto frontier is nearly parallel to the search direction. The resulting irregularities will affect the quality of a coordinate system based on the weights.
4. The placement of the basepoints is determined entirely by the locations of the individual minima. If these are located incorrectly, e.g. if the optimizer converges to a local minimum, the entire sampled frontier will be affected.
5. Each of the subproblems is solved independently, so their solution can be easily parallelized to speed the computations.

Each of the above points has implications for using the basepoints' weights as coordinates to parameterize the Pareto frontier. Observation (1) is the most problematic, as it severely limits NBI's ability to represent higher dimensional Pareto frontiers. As an example, Figure 42 illustrates the inability of NBI to sample from the entirety of the sphere example problem's frontier. The excluded regions lie near the edges of the frontier; thus they are the regions that would be preferred if one attribute were relatively unimportant to the decision maker. It is this shortcoming that the modifications described in the remainder of this chapter seek to overcome.

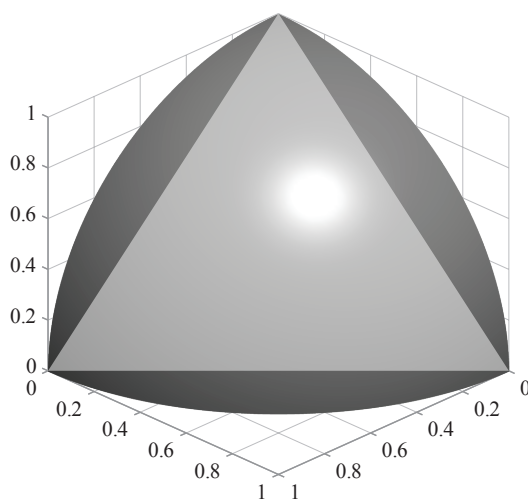


Figure 42: Three-attribute spherical Pareto frontier with portion searched by NBI lightened

Observation (2) notes that for Pareto frontiers with concave regions, NBI often samples points that are globally dominated. The implication for the present application is that the NBI-based coordinate system may parameterize portions of the boundary of the objective space that are not on the frontier.

Observation (3) notes that although the basepoints are evenly spaced along the CHIM, their mapping to the frontier necessarily introduces local stretching and compressing of the coordinate values depending on the local curvature of the frontier. This is a minor complication that potentially reduces the degree of nonlinearity of the eventual function $(\mathbf{x}, \mathbf{y}) = g(\mathbf{w})$.

Observations (4) and (5) will become relevant in the modified version of NBI described

in the next section. The general approach that will be used to enable sampling from the entire frontier is to place the basepoints based on the locations of the *lower-dimensional subfrontiers* instead of only considering the locations of the individual minima. Consequently, the “trickle down” effect described in the fourth point above becomes more prominent. A related issue is that the modified NBI approach requires that the subproblems be solved in a particular order, which makes parallelization more difficult.

6.2 *Modified NBI Approaches by Motta et al. and Mueller-Gritschneider et al.*

The most significant shortcoming of NBI, as described in the previous section, is that the search basepoints are arranged in a way that prevents sampling of the entire Pareto frontier. This effect is illustrated in the first column of Figure 49 for the five example problems.

Several authors have modified the NBI algorithm to enable sampling of the entire Pareto frontier. An early example is the (normalized) normal constraint (NC) method by Messac et al. [102, 104], which maintains NBI’s idea of distributing the basepoints along a simplex but enlarges the simplex such that it encompasses the entire Pareto frontier. However, this approach leads many of the basepoints to lie entirely outside the feasible objective space, which eliminates the possibility of a one-to-one mapping from the weights to the sampled Pareto frontier.

Two recent approaches, by Motta et al. [112] and Mueller-Gritschneider et al. [113], solve the problem of sampling the entire frontier while maintaining a one-to-one mapping from basepoints to sampled points. Each of these approaches is based on the same important observation about the subfrontiers: the projection of the k -attribute Pareto frontier into a $(k - 1)$ -dimensional space that leaves out one attribute is bounded by the corresponding $(k - 1)$ -subfrontiers.¹ Based on this observation, a strategy for sampling from the entire Pareto frontier was developed that entails alternately sampling from increasingly high-dimensional subfrontiers (to extend the region bounded by the sampling) and redistributing the basepoints inside the region of the objective space bounded by the current sampling (to

¹This is proven in Section 3.3.2.

ensure the next higher subfrontier is evenly sampled).

6.2.1 Successive Sampling of Subfrontiers

In Motta et al.'s and Mueller-Gritschneider et al.'s modified NBI approaches, the weight tuples are interpreted as corresponding to particular subfrontiers depending on which of their weights have non-zero values. Recall that this same interpretation was found for the non-domination level coordinates in Chapter 5. For example, a basepoint in a five-attribute problem whose coordinates are $[0, 0.4, 0.1, 0, 0.5]$ would be optimized to lie on the subfrontier of attributes 2, 3, and 5 – the indices of the non-zero coordinates.

This approach differs from the classic NBI method, in which the weights serve only to locate the basepoints according to the equation $\mathbf{y}_j^0 = \Phi \mathbf{w}_j$, and only the individual minima are optimized to particular subfrontiers, i.e. the 1-subfrontiers. Consequently, classic NBI is generally unable to sample points on, or even near, the higher-dimensional subfrontiers, as can be seen in Figure 42.

The strategy of Motta et al. and Mueller-Gritschneider et al. is to enable subfrontiers to be sampled by enforcing the NBI search direction constraint only in those dimensions whose corresponding weight in \mathbf{w}_j is non-zero. This is accomplished by multiplying each term in the equality constraint in Eq. 7 by a diagonal matrix A_j where $A_{j,ii} = 1$ if the i -th weight of \mathbf{w}_j is nonzero, i.e. if the solution to the subproblem should be optimized to lie on a subfrontier that omits the i -th attribute, and $A_{j,ii} = 0$ otherwise.

The modified NBI subproblem is then

$$\begin{aligned} & \max_{\mathbf{x}, t} t \\ & \text{s. t. } A\mathbf{y}_j^0 + tA\hat{\mathbf{n}} = A\mathbf{f}(\mathbf{x}) - A\mathbf{u}^+ \\ & \mathbf{x} \in D \end{aligned} \tag{8}$$

The effect of multiplying each term by A is to increase the dimensionality of the search space when the sampled point is to lie on a subfrontier. For points that lie only on the k -subfrontier (i.e. the full Pareto frontier), A_j is the identity matrix and the modified subproblem (8) becomes identical to the original NBI subproblem (7).

The result of expanding the search space on the distribution of sampled points is illustrated for the two-attribute subfrontiers of the sphere example problem in Figure 43. The basepoints are shown in gray, the corresponding sampled points in black, and the search directions as dotted lines for (a) the classic NBI approach and (b) the successive subfrontier optimization approach used by Motta et al. and Mueller-Gritschneider et al. Only the modified approach in (b) is able to sample from the true subfrontiers.

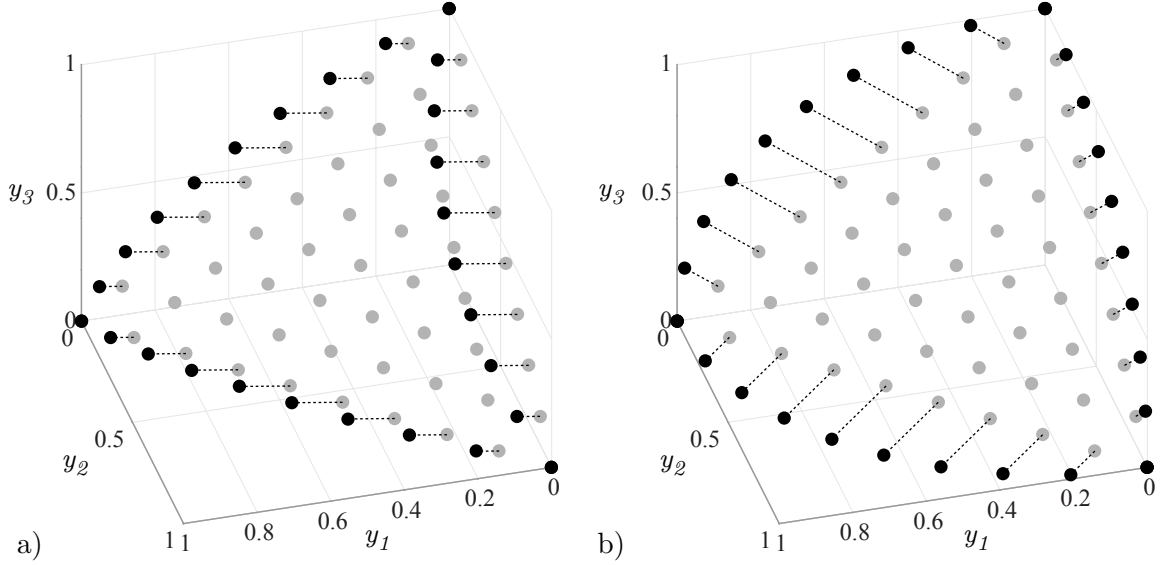


Figure 43: Optimization of 2-attribute subfrontiers by a) original NBI b) modification by Motta et al. and Mueller-Gritschneider et al.

6.2.2 Basepoint Redistribution

The successive sampling of subfrontiers using increased-dimensional search spaces enables the boundaries of the Pareto frontier to be sampled, but this effect alone does not lead to an even sampling of the entire frontier. An additional step is now needed: the basepoints must be adaptively redistributed to fill the region bounded by the subfrontiers. When redistribution is combined with successive subfrontier sampling, the resulting algorithm is able to evenly sample from the entire Pareto frontier.

The approaches developed by Motta et al. and Mueller-Gritschneider et al. both redistribute the basepoints that lie on the p -subfrontiers immediately after optimizing the

$(p - 1)$ -subfrontiers. The two approaches differ, however, in how the p -subfrontier's basepoints are redistributed. Motta et al. redistribute the points using a centroidal Voronoi tessellation design of experiments [42, 112], while Mueller-Gritschneider et al. [113] place each basepoint by solving a linear program. Comparing the results of these approaches that are presented in references [112] and [113] suggests that the linear programming approach of Mueller-Gritschneider et al. provides a more even distribution of the basepoints and better maintains a grid-like arrangement that can be easily translated into a coordinate system on the frontier. For this reason, the linear programming approach was selected for the present work.

In order to demonstrate the linear programming approach for locating basepoints, we consider a basepoint, \mathbf{y}_j^0 , whose weights, \mathbf{w}_j correspond to a p -subfrontier. Let W denote the set of weight tuples belonging to any subfrontier of *fewer than* p attributes², and let Y denote the set of sampled points for which the corresponding weights are in W .

For example, in Figure 43b the black points compose Y for $p=3$ because they represent the 1- and 2-subfrontiers³. The corresponding gray points that are linked by the dotted lines compose W .⁴ The remaining (unlinked) gray points are the basepoints \mathbf{y}_j^0 on the 3-subfrontier that must be redistributed.

The general strategy of Mueller-Gritschneider et al.'s redistribution approach is to represent the basepoint \mathbf{y}_j^0 as a linear combination of the points $\mathbf{y}_i \in Y$,

$$\mathbf{y}_j^0 = \sum_{i=1}^m \mathbf{y}_i a_{j,i} : \mathbf{y}_i \in Y, a_i \geq 0, \sum a_i = 0 \quad (9)$$

In this equation, both \mathbf{y}_j^0 and $a_{j,i}$ are unknown. In order to resolve this, Mueller-Gritschneider et al. define a parallel linear programming problem in the weights,

$$\mathbf{w}_j = \sum_{i=1}^m \mathbf{w}_i a_{j,i} | \mathbf{w}_i \in W, a_i \geq 0, \sum a_i = 0 \quad (10)$$

in which the only unknowns are the $a_{j,i}$ coefficients. The $a_{j,i}$ calculated from (10) are then used in (9) to solve for \mathbf{y}_j^0 .

²Equivalently, a weight tuple \mathbf{w}_i is in W if and only if \mathbf{w}_i has fewer non-zero weights than \mathbf{w}_j

³The 1-subfrontiers are simply the individual minima.

⁴The black points representing the three individual minima coincide with their corresponding gray basepoints; these are included in W .

When \mathbf{w}_j corresponds to a 3-subfrontier or higher the number of points in W , denoted by m , is necessarily larger than k , and (10) is underdetermined. It is this situation that requires a linear programming solution. Mueller-Gritschneider et al.’s particular approach is to select the a_i such that a_i is large for the \mathbf{w}_i in W that are most similar to \mathbf{w}_j , as this will lead to \mathbf{y}_j^0 being located according those \mathbf{y}_i in Y to which the eventual sampled point \mathbf{y}_j is presumed to be most similar. Mueller-Gritschneider et al. use the 1-norm to represent the similarity between \mathbf{w}_j and $\mathbf{w}_i \in W$,

$$\mathbf{d}_{j,i} = \|\mathbf{w}_j - \mathbf{w}_i\|_1, \mathbf{w}_i \in W$$

such that smaller values of $\mathbf{d}_{j,i}$ indicate greater similarity.

The objective of the linear program is thus to minimize $\mathbf{d}_j^\top \mathbf{a}_j$ while ensuring that \mathbf{a}_j is a solution to (10) above. The linear program is formulated as

$$\begin{aligned} \min_{\mathbf{a}_i} \quad & \mathbf{d}_j^\top \mathbf{a}_j \\ \text{s. t.} \quad & \mathbf{w}_j = \sum_{i=1}^m \mathbf{w}_i a_i | \mathbf{w}_i \in W \\ & a_i \geq 0 \\ & \sum a_i = 1 \end{aligned} \tag{11}$$

Finally, the solution of (11) is substituted into (9) to yield the basepoint \mathbf{y}_j^0 . Once the basepoint is obtained, the subproblem is solved according to (8).

6.2.3 Results

The results of using Mueller-Gritschneider et al.’s approach on the five example problems are shown in the second column of Figure 49. The resulting frontiers are certainly more representative of the true Pareto frontier than Das and Dennis’s method, but a characteristic flaw is visible in each plot. The linear programming approach results in each basepoint being placed slightly too uniformly around the nearest individual minimum. These small errors eventually build up, creating a characteristic “tear” through the center of the frontier where no points are sampled. The pattern of this tear is remarkably consistent across the different example Pareto frontiers, suggesting that it is not merely the effect on one particular geometry. The next section proposes two small modifications to Mueller-Gritschneider

et al.’s approach that typically result a more evenly spaced sampling.

6.3 Further Modifications to Improve Sampled Frontier Spacing

The distribution of points on the sampled frontier is a direct consequence of the distribution of basepoints, so improvements to the sampled frontier must be achieved by improving the basepoint locations. The goal here is to improve the basepoint locations without altering the basic premise of Mueller-Gritschneider et al.’s linear programming approach. Two elements of Eq. 11 are potential candidates for improvement: the vector of distances, \mathbf{d}_k , and the set of barycentric coordinates that are considered, W . This section describes possible modifications to these elements that typically result in more evenly sampled Pareto frontiers. The resulting NBI variant that uses Mueller-Gritschneider et al.’s linear programming approach with the modifications described in this section will be called the “mod-NBI” method for brevity.

6.3.1 Generalization to p -Norms

In Mueller-Gritschneider et al.’s approach, the similarity between \mathbf{w}_j and the weights in W is represented by the 1-norm. Without changing the general concept of using a vector of distances as the coefficient matrix in the linear program, a simple change is consider a different p -norm⁵ for calculating the distances. Figure 44 shows the sampled frontiers for the sphere example problem that result from using various p -norms. The frontiers are sampled using 50 levels for the weights. Interestingly, identical samplings are obtained for $p = 1$ and $p \rightarrow \infty$. Intermediate values of p yield significantly more even samplings; for $p \approx 1.5$ to $p \approx 6$ the sampling is very even and fairly insensitive to the exact value of p . A value of $p = 2$ will be used here, as it yields a very good sampling for all the example problems and it has an intuitive meaning of measuring the Euclidean distance.

6.3.2 Considering Distances to Additional Points

An additional improvement can be implemented by including additional points in W , the set of points whose distance to \mathbf{w}_j is considered in the linear program. In Mueller-Gritschneider

⁵ $\|\mathbf{z}\|_p = (\sum_{i=1}^k |z_i|^p)^{1/p}$

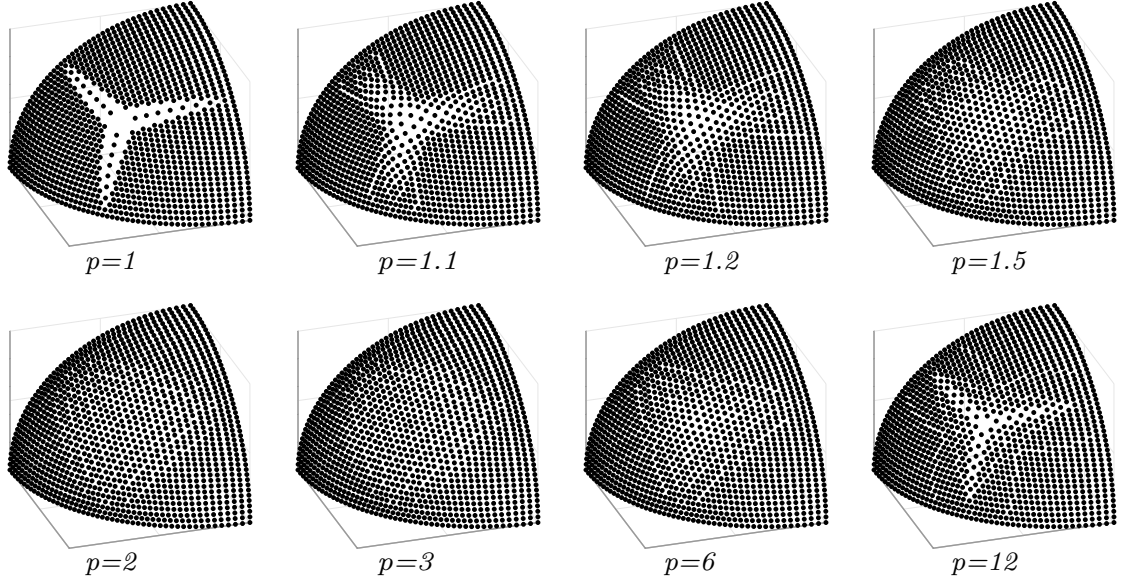


Figure 44: Comparison of p -norms on sampled frontier of sphere example problem

et al.'s approach, only points on the lower-dimensional subfrontiers are included in W . However, when the elements of \mathbf{w}_j are such that the corresponding \mathbf{y}_j^0 will be near the center of its subfrontier, the points in Y can be relatively distant from \mathbf{y}_j^0 . An example of this behavior is illustrated in Figure 45, which shows the state of the sphere example problem after the 2-subfrontiers have been sampled. The resulting sampled points are shown in black, while the basepoints whose subproblems have yet to be solved are shown in gray. These points are drawn in their classical-NBI locations so that the next step is to redistribute them according to the location of the black points.

Using Mueller-Gritschneider et al.'s approach, each gray point would be placed by solving a linear program with the black points composing W . Recall that this placement is in contrast with the classical NBI approach in which each gray point is placed based solely on the locations of the individual minima. By also including all lower-dimensional subfrontiers in W , more information is available, so the basepoints are able to be more evenly placed throughout the entire Pareto frontier.

The intent of the linear program is to place each \mathbf{y}_j^0 according to the most similar sampled points in Y . However, by examining Figure 45, it becomes clear that many sampled

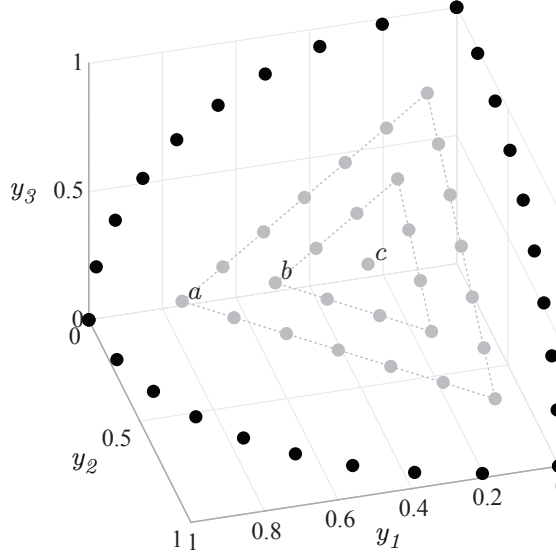


Figure 45: Basepoints (gray) to be redistributed based on sampled subfrontier (black)

points are being neglected by limiting Y to contain only sampled points that are on lower-dimensional subfrontiers. If, for example, the subproblem corresponding to basepoint c were to be solved last, then by the time it were solved, there would be many additional sampled points *on the same subfrontier* as c that could have been included in Y – those corresponding to the basepoints on rings a and b .

It seems likely that including additional sampled points in Y that are closer to \mathbf{y}_j^0 would improve the placement of \mathbf{y}_j^0 ; however, care must be taken with this approach. If each successive sampled point is added to Y , so that the corresponding weights are included in W when solving the next basepoint's linear program, the distribution of basepoints actually gets worse. The reason seems to be that this approach leads W to become “imbalanced” in sense that the mean of the weight tuples in W is no longer equal to $[1/k, 1/k, \dots, 1/k]$ as it is when W contains only weights corresponding to lower-dimensional subfrontiers. Although it is not intuitive that this should matter, it has been found that adding points to Y in such a way that W remains balanced yields a significantly more even distribution of basepoints.

For the example in Figure 45, maintaining balance in W requires adding every sampled point whose basepoint lies on the same ring (a or b) to Y at the same time. Using this approach, beginning with the state shown in Figure 45, the basepoints on ring a would be

redistributed with W comprising the black sampled points. The NBI subproblems corresponding to the points on ring a are then solved, and the resulting sampled points are added to W . This process is then repeated for ring b , and finally for point c . The important point here is that rather than adding the solutions to the ring a or b subproblems to W as each one is solved, they are not added until *all* of the subproblems on the ring have been solved – 18 subproblems for ring a , 9 for ring b . This procedure ensures that W remains balanced at all times.

The generalization of this procedure to problems with more than three attributes, or how it can be abstracted into an algorithm, may not be immediately clear. Surprisingly, however, the above approach is easily implemented in practice because of a simple pattern that becomes apparent when one considers the sequential solving of concentric rings of basepoints. Unlike Mueller-Gritschneider et al.’s approach, which requires a double-loop structure of looping over each subfrontier and then looping to solve the points that comprise that subfrontier, the concentric-ring solution requires only a simple ordering of the weights at the start of the algorithm, and then a single loop to solve the subproblems in the order of the sorted weights.

The required ordering of the weights is revealed by the defining properties of the concentric rings of basepoints described above: 1) every point on a ring belongs to the same subfrontier and 2) every point on a ring has the same value for its smallest non-zero weight. These properties are applicable to frontiers with any number of attributes, and to rings that lie on any subfrontier. In order to implement the concentric ring approach, the weights must be ordered such that, for a given subfrontier, no point on an inner ring is solved until every point on all outer rings have been solved. Additionally, the subfrontiers must be solved in order of increasing dimensionality, as in the original approach.

Fortunately, it is simple to sort an arbitrarily ordered list of weight tuples to meet the above requirements. The steps of the ordering process are illustrated in Figure 46 for a three-attribute problem with five levels per coordinate. Each table shows the three weights that comprise each tuple on the right and an identifying row number on the left. First, each weight tuple is sorted within itself so that the weights are listed from smallest to largest.

In doing this, the physical interpretation of the weights as relating to particular attributes is lost; however, this is merely an intermediate step in the ordering process. Next, the list of sorted weight tuples is ordered using a lexicographical ordering, i.e. the rows are sorted according to the first weight, ties are settled by the second weight, and remaining ties are settled by the third weight. The resulting twice-sorted weights are not used directly; rather, we are interested only in the *order* in which the weight tuples now appear. This order is indicated by the sorted row identifiers on the left side of the third table in Figure 46. The final step is to arrange the original (unsorted) weight tuples in this order, as shown in the rightmost table in Figure 46.

Original order				Sort each tuple				Lex. row sort				Unsort each tuple			
1	1	0	0	1	0	0	1	1	0	0	1	1	1	0	0
2	0.75	0.25	0	2	0	0.25	0.75	5	0	0	1	5	0	1	0
3	0.5	0.5	0	3	0	0.5	0.5	15	0	0	1	15	0	0	1
4	0.25	0.75	0	4	0	0.25	0.75	2	0	0.25	0.75	2	0.75	0.25	0
5	0	1	0	5	0	0	1	4	0	0.25	0.75	4	0.25	0.75	0
6	0.75	0	0.25	6	0	0.25	0.75	6	0	0.25	0.75	6	0.75	0	0.25
7	0.5	0.25	0.25	7	0.25	0.25	0.5	9	0	0.25	0.75	9	0	0.75	0.25
8	0.25	0.5	0.25	8	0.25	0.25	0.5	13	0	0.25	0.75	13	0.25	0	0.75
9	0	0.75	0.25	9	0	0.25	0.75	14	0	0.25	0.75	14	0	0.25	0.75
10	0.5	0	0.5	10	0	0.5	0.5	3	0	0.5	0.5	3	0.5	0.5	0
11	0.25	0.25	0.5	11	0.25	0.25	0.5	10	0	0.5	0.5	10	0.5	0	0.5
12	0	0.5	0.5	12	0	0.5	0.5	12	0	0.5	0.5	12	0	0.5	0.5
13	0.25	0	0.75	13	0	0.25	0.75	7	0.25	0.25	0.5	7	0.5	0.25	0.25
14	0	0.25	0.75	14	0	0.25	0.75	8	0.25	0.25	0.5	8	0.25	0.5	0.25
15	0	0	1	15	0	0	1	11	0.25	0.25	0.5	11	0.25	0.25	0.5

Figure 46: Steps for ordering weights for sequentially solving concentric rings

These sorting operations are standard routines in most programming languages. For example, in the MATLAB source code listed in Appendix C, generating and reordering the weights is accomplished with four lines of code,

```
weights=trigriddn(numpts,numobj); %generate weight grid
sortedweights=sort(weights,2); %sort each tuple
[sortedweights,order]=sortrows(sortedweights); %lexicographical sort
weights=weights(order,:); %arrange original weights in this order
```

The four tables in Figure 46 correspond to the state of `weights/sortedweights` after each of these four lines of code.

The resulting order in which the subproblems are to be solved is illustrated on a notional weight grid in Figure 47. This ordering automatically ensures that the subfrontiers are solved in the correct order of increasing dimensionality. The first three subproblems correspond to the individual minima, then subproblems 4-12 correspond to the 2-subfrontiers,

and finally subproblems 13-15 correspond to the 3-subfrontier. Similarly, within each subfrontier, points are solved from the outside inward. For example, subproblems 10, 11, and 12 each lie at the center of their respective subfrontier, so that they are each bounded by rings (in this two-attribute case, pairs) of previously solved subproblems that can be used to form the sets W and Y . Points that lie on the same ring, such as 13, 14, and 15, or on equivalent rings on different subfrontiers, such as 10, 11, and 12, can be solved in any order relative to each other. This is confirmed by the fact that the rows corresponding to these sets of points are identical in the second table of Fig 46 so that their relative lexicographical ordering in the third table is arbitrary.

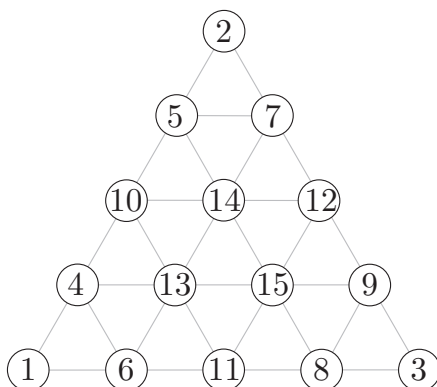


Figure 47: Three-attribute, five-level weight grid, labeled by order in which corresponding subproblems are solved

The final step of the implementation is to ensure that the set W contains all points on lower-dimensional subfrontiers and all points on outer-rings of the same subfrontier as the current w_j . Each of these criteria is easily checked from the sorted weights in the third table of Figure 46. In this table, rows are in the same order as the final ordered weights, and the subfrontier dimensionality is indicated by the number of leading zeros, i.e. a point w_i is on a lower dimensional subfrontier than w_j if the i -th row in this table has more leading zeros than the j -th row. The ring that a point lies on is indicated by the value of the smallest non-zero entry. I.e. a basepoint whose weights are w_i is on an outer ring relative to a basepoint whose weights are w_j if the first non-zero entry in the i -th row of this table is smaller than the first non-zero entry of the j -th row. The set W is formed by

simply checking each $w_i : i < j$ to see if it meets these criteria.

The effects of applying each of the above modifications to Mueller-Gritschneider et al.'s approach are illustrated in Figure 48. Either of the two modifications on its own effectively closes the large gaps that the original approach leaves in the center of the frontier. However, even with both modifications the point spacing remains slightly irregular, being sparser in the center of the frontier than near the edges. This appears to be a fundamental shortcoming of the linear programming approach.

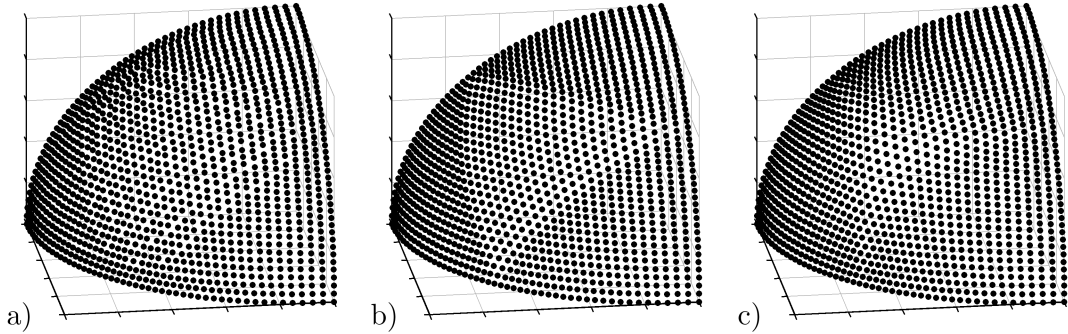


Figure 48: Comparison of improvements to Mueller-Gritschneider et al.'s approach. a) 2-norm only, b) expanded W only, c) 2-norm and expanded W

6.4 Application to Example Problems

The region of each example problem's objective space that is sampled by each NBI variant is illustrated in Figure 49. Conceptually, the classic NBI approach by Das and Dennis may be thought of as sampling a subset of the frontier that would project onto the CHIM. Consequently, NBI yields a simplex-like sampled frontier even if the true Pareto frontier is substantially different. This is most clearly seen in Figure 49b, for which NBI's sampling of the truncated sphere is extremely misleading. Only for the concave problem in Figure 49c is NBI able to sample the entire frontier; however, in this case over half the NBI subproblems yield unfeasible points that extend beyond the boundary of the Pareto frontier.

Another well known shortcoming of NBI is demonstrated in Figure 49e, which shows NBI sampling from regions of the boundary of the objective space that are globally dominated. This happens simply because each subproblem is solved in isolation, so no global non-domination checks are possible. In most cases this behavior is not particularly problematic.

The dominated points can be removed as a postprocessing step, or they may be retained in order to yield a more continuous representation of the frontier. In the present application of using the NBI weights as a coordinate system, these points must be retained in order to maintain a continuous coordinate system. However, the tradeoff is that the resulting coordinate system does not parameterize just the Pareto frontier but also includes some dominated designs.

The improvements to NBI made by Mueller-Gritschneider et al. are illustrated in the second column of Figure 49. This approach solves the problem of NBI being unable to sample regions of the Pareto frontier that project beyond the CHIM boundary. Similarly, in the case of the concave problem it solves the problem of wasting subproblems sampling unfeasible points. Across all the example problems, only five unfeasible points resulted from using Mueller-Gritschneider et al.’s algorithm: one near the maximum value of y_1 for the bumpy sphere problem and four on the $\{y_1, y_3\}$ subfrontier for the concave problem.

However, this method has a characteristic flaw that makes it unsuitable as the basis for a coordinate system. Each sampled frontier, with the possible exception of the concave example problem, shows a characteristic tear pattern near the center of the frontier. The cause of this pattern seems to be that Mueller-Gritschneider et al.’s approach locates each basepoint slightly too close to the nearest individual minimum. A single row of sampled points lies inside the tear; these are points whose weights are of the form $[ab/2b/2]$, the symmetry of which ensures that they are centrally located in the dimensions whose coordinates match. (However they are still located incorrectly in the non-matching coordinate’s dimension, as evidenced by the fact that these rows do not extend to meet in the center of the frontier.) The bumpy sphere problem shows two additional tears extending upward from the bottom-center of the frontier. These seem to be due to the relative sparsity of sampled points along the $\{y_1, y_2\}$ subfrontier in this region. This initial error then propagates to the interior of the sampled frontier.

The sampled frontiers using the final “mod-NBI” approach are shown in the third column of Figure 49. These frontier have none of the major shortcomings of Das and Dennis’s approach or Mueller-Gritschneider et al.’s approach. Additionally, every subproblem resulted

in a feasible sampled point. The sampling is still not perfect, however. A characteristic sparsity pattern can be seen near the center of the frontier. This seems to arise because the modifications suggested in Section 6.3 improve upon, but fundamentally do not solve, the underlying shortcoming of the linear programming approach.

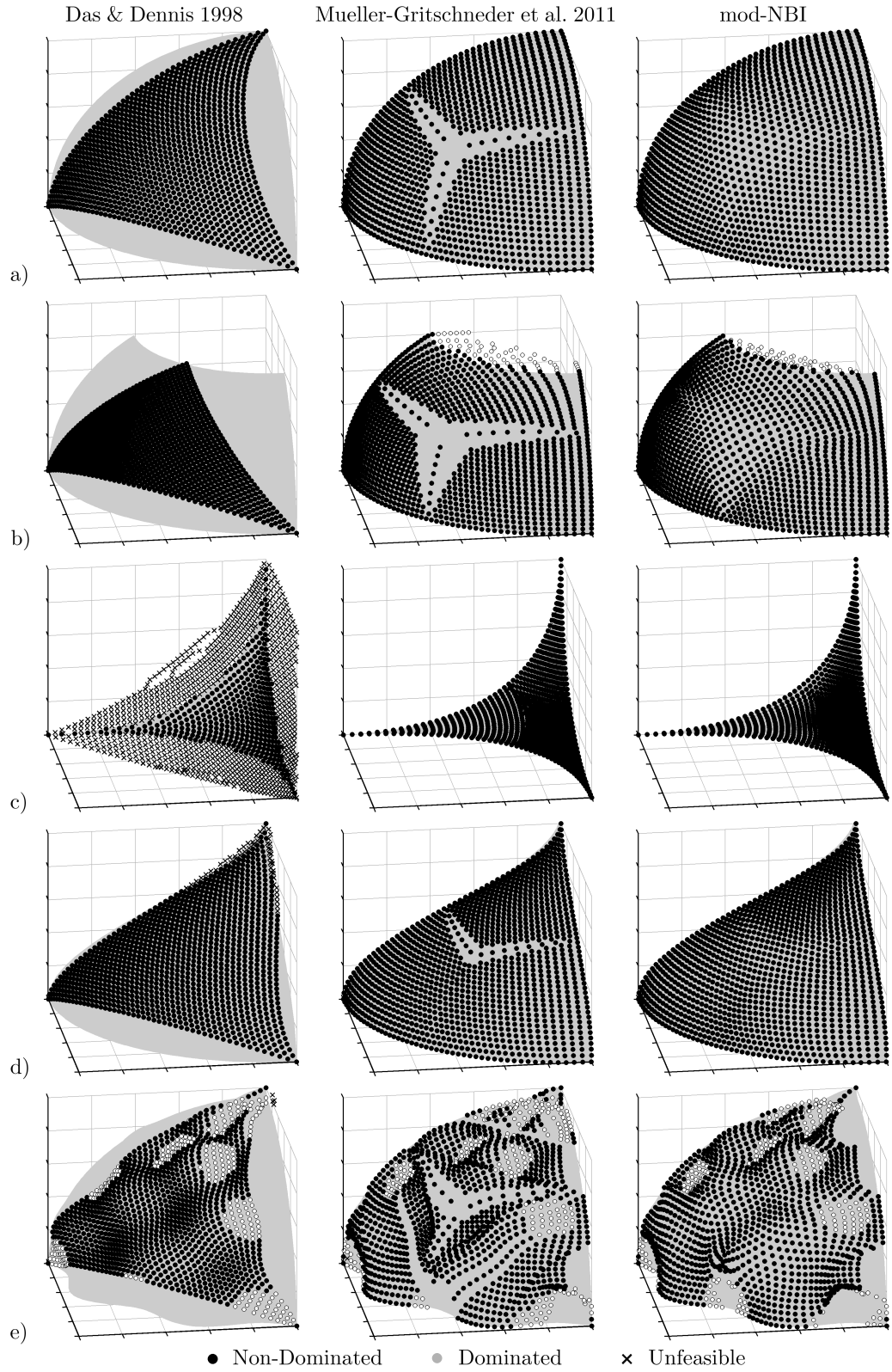


Figure 49: Comparison of normal boundary intersection methods applied to test problems. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

Figure 50 shows the mod-NBI sampled frontier colored according to each points corresponding weights. The plots reveal that the weights follow the correct trends to serve as coordinates on the frontier. Because the sampling is directly based on the grid of weights, the mapping of weights to sampled points is noise free. However, this also causes the sparser sampling near the center of the frontiers to affect the weight-based coordinate system. The result is a noticeable change in the curvature of the iso-coordinate curves where they enter and exit the sparsely sampled centers. The effect is small, and, as will be seen in Chapter 8, does not have much impact on the coordinate-based exploration techniques.

Rather than using the mod-NBI weights as coordinates, an alternative approach is to use mod-NBI to sample the frontier and then calculate NDLCs . This approach is illustrated in Figure 51. Compared to the NDLCs calculated for frontiers sampled using NSGA-II, shown in Figure 39 on page 112, the NDLCs based on mod-NBI sampled frontiers are significantly smoother. However, the NDLCs are negatively affected by mod-NBI's sparse sampling in the center of the frontiers. Comparing Figure 51 with Figure 50 does not suggest any reason to use NDLCs in place of the mod-NBI weights; consequently the pairing of mod-NBI sampling with NDLCs will not be considered further in this work.

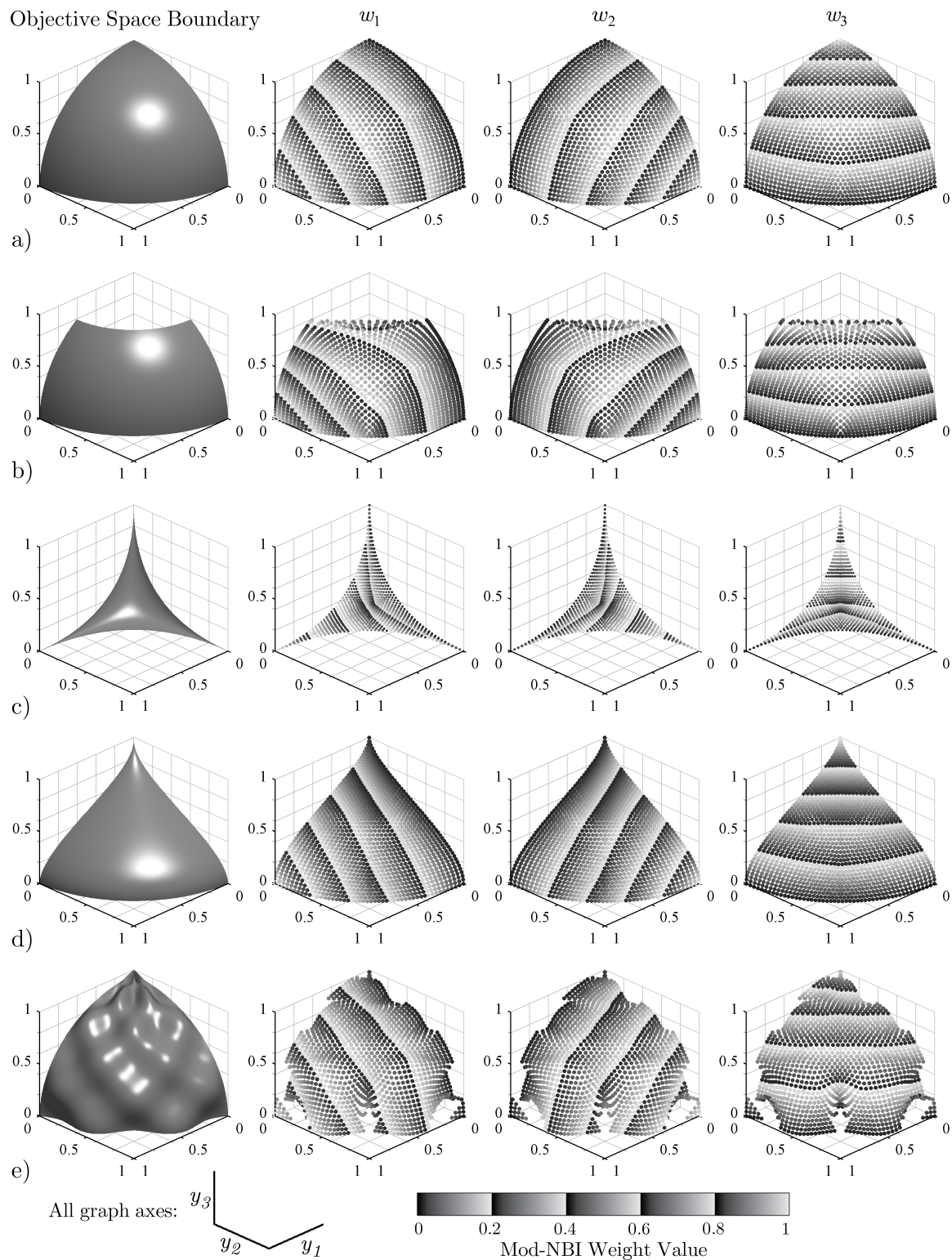


Figure 50: Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding mod-NBI weights. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

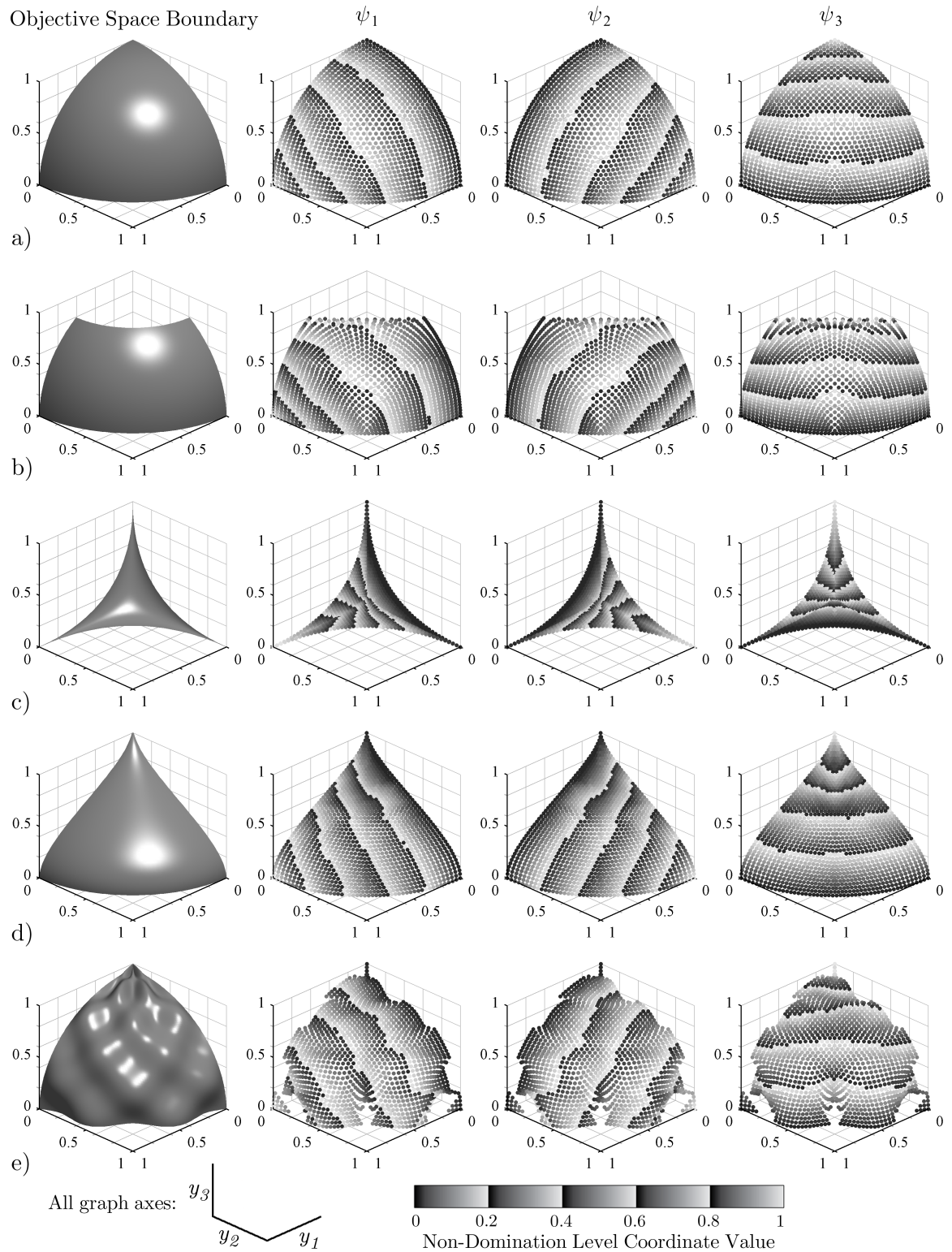


Figure 51: Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding NDLCs. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

CHAPTER VII

A SELF-ORGANIZING MAP COORDINATE SYSTEM

The non-domination level coordinate system and the mod-NBI coordinate system described in the previous chapters have been found to work well, but each has room for improvement. The NDLCs are very sensitive to local irregularities in the distribution of the sampled frontier from which they are constructed. The mod-NBI coordinate system is slightly irregular near the center of the frontier and depends on the success of its particular multi-objective optimization approach.

This chapter introduces a method for defining coordinates on a sampled Pareto frontier based on self-organizing maps that largely solves the shortcomings of the previously described coordinate systems. Self-organizing maps are a type of artificial neural network invented by Teuvo Kohonen in the 1980s for discovering patterns and structure in data sets by conforming a “topology” of nodes to the sampled data [77]. This process has clear parallels to the present objective of conforming a coordinate system to a sampled Pareto frontier. The general approach taken in this chapter is to use an unsupervised learning algorithm to train a simplex-like topology, whose vertices are labelled with barycentric coordinates, to a sampled Pareto frontier. As with non-domination level coordinates, this coordinate system is calculated from a discrete sampling of the frontier, which may be obtained from any multi-objective optimizer. A special learning algorithm is used to ensure that the coordinate values are evenly distributed throughout the Pareto frontier and that 0-coordinates are correctly mapped to the corresponding subfrontiers.

The chapter begins with an overview of the classical SOM algorithm and then proceeds to introduce a new variant of SOMs, named “Pareto simplex self-organizing maps,” (PSSOMs) that is more closely related to the problem of mapping a continuous coordinate system to a surface. The PSSOM method is demonstrated on example Pareto frontiers that have been sampled using NSGA-II and the mod-NBI algorithm introduced in Chapter 6.

MATLAB source code for creating PSSOMs and their associated coordinates is provided in Appendix D. The reader may find it helpful to reference this appendix while reading this chapter to help clarify certain points. The source code utilizes MATLAB’s Neural Network Toolbox for creating the self-organizing maps, thus some details of the overall process are not included in the code. However, the modifications that differentiate Pareto simplex SOMs from classical SOMs are all included.

7.1 *Applications of Self-Organizing Maps*

Self-organizing maps were originally developed for vector quantization, which is the mapping of an arbitrary input vector to a member of a discrete, *viz.* quantized, set of vectors called “codebook vectors” [77]. This approach can be thought of as a data compression or simplification approach. For example, an arbitrary color, defined by the wavelength of a corresponding photon, could be quantized to the name of the most similar color from the set {red, orange, yellow, green, blue, indigo, violet}.¹

Self-organizing maps are unique in that they perform this quantization based on the spatial distribution of the training data so that the codebook vectors² become evenly distributed in the training data space. The particular approach taken by self-organizing maps is to train, i.e. fit, a sheet-like grid of connected nodes (the codebook vectors) to the basis data in such a way that the distribution of the SOM nodes mimics the distribution of the basis data. The result of the training is thus the vector of data-space coordinates for each SOM node, which are called the “input weights.” When the trained SOM is presented with a new input vector, it outputs the index of the closest (as defined by some metric) SOM node; the weights of this node then serve as the quantization of the input vector [77].

Today, a more common application of SOMs is for high-dimensional data visualization. In this application, the trained SOM is not presented with new data vectors to be quantized;

¹This example highlights the meaning of quantization, but it is not typical of the type of quantization problems typically solved using SOMs. Self-organizing maps generally predefine the number and topology of the codebook vectors, but not their labels (in this case, the color names). The discovery of meaningful labels for the codebook vectors is one of the desired outcomes of using SOMs.

²The “codebook vector” of vector quantization is generally called a “model vector” in the context of self-organizing maps, however I will use “codebook vector” in both contexts to avoid overloading the word “model” in this discussion.

instead, the weights of the codebook vectors are visualized using a colored-tile plot that is endemic to SOMs. As a result of the training process, each node's n -th weight is equal to the mean of x_n , the n -th dimension of the training data, for the subset of the training data that the SOM would quantize to that node. Because of the minimal-distance condition by which the training data is quantized to the SOM nodes, visualizing the weights of the codebook vectors is equivalent to visualizing the mean values of the training data points when they have been binned according to a Voronoi tessellation of the SOM nodes. In many cases this visualization is more meaningful than the conceptually similar “heatmap” visualization, which bins data according to a rectangular grid of cells. A comparison of the bins defined using a Voronoi tessellation of an SOM trained to a 2-D data set vs. the rectangular grid of a heatmap is shown in Figure 52. The heatmap grid (b) is constructed independently of the data distribution (a). Consequently the distribution of data points to heatmap bins is highly uneven, with many bins containing no data points. If instead of using a rectangular grid, a SOM is first fit to the data (c), the bins formed by the Voronoi tessellation of the SOM nodes are distributed such that assignment of points to Voronoi cells is much more uniform (d).

7.2 A Coordinate System Based On Self-Organizing Maps

The present application of mapping coordinates to the Pareto frontier has some similarities with the classical SOM, but also some significant differences. The most important similarities are that the barycentric coordinate system can be represented as a node-link diagram, as shown in the left side of Figure 33 on page 96, and that the goal is to evenly distribute the coordinate system across the Pareto frontier in much the same way that the SOM nodes become distributed throughout the training data. However, a significant departure from the classical SOM is that the present goal is to assign precise coordinate values to *every point* on the Pareto frontier, not merely to define coordinates at a few discrete locations, i.e. the codebook vectors. This assignment requires some way of interpolating values along the SOM. A second difference is that the assignment of coordinates to subfrontiers must obey the rule that 0 coordinate values map to the corresponding subfrontiers. This rule

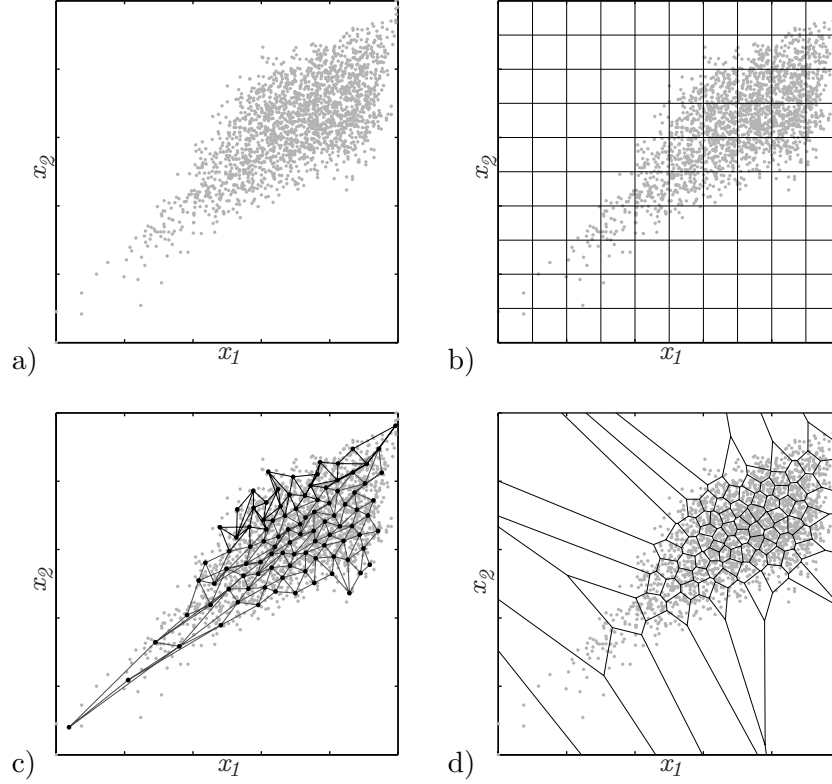


Figure 52: Comparison of data binning by 10×10 rectangular heatmap grid and 10×10 SOM. a) Distribution of training data. b) Uniform heatmap-style bins. c) Trained SOM, showing nodes and edges. d) Voronoi tessellation of SOM nodes in (c).

is effectively a constraint on the location of certain nodes. The final significant difference is that in order for the coordinates to be meaningful to a decision maker, they must be uniformly spread across the frontier. Although the classical self-ordering map achieves a fairly uniform distribution of nodes, it is not sufficiently uniform for this application.

In order to illustrate the need for changes to the classical SOM, Figure 53(a) shows the result of fitting a SOM to the teardrop Pareto frontier that has been sampled using the modified NBI method of Chapter 6. Despite the very uniform distribution of the sampled frontier, the SOM shows characteristic irregularities in its node placement. For comparison, the final coordinate grid created using the Pareto simplex self-organizing map developed in this chapter is shown in Figure 53(b). The PSSOM achieves a very uniform assignment of coordinates to the Pareto frontier, better than either the NDLC or the modified NBI approaches. The remainder of this chapter describes the SOM training process in detail

and the modifications needed for PSSOMs.

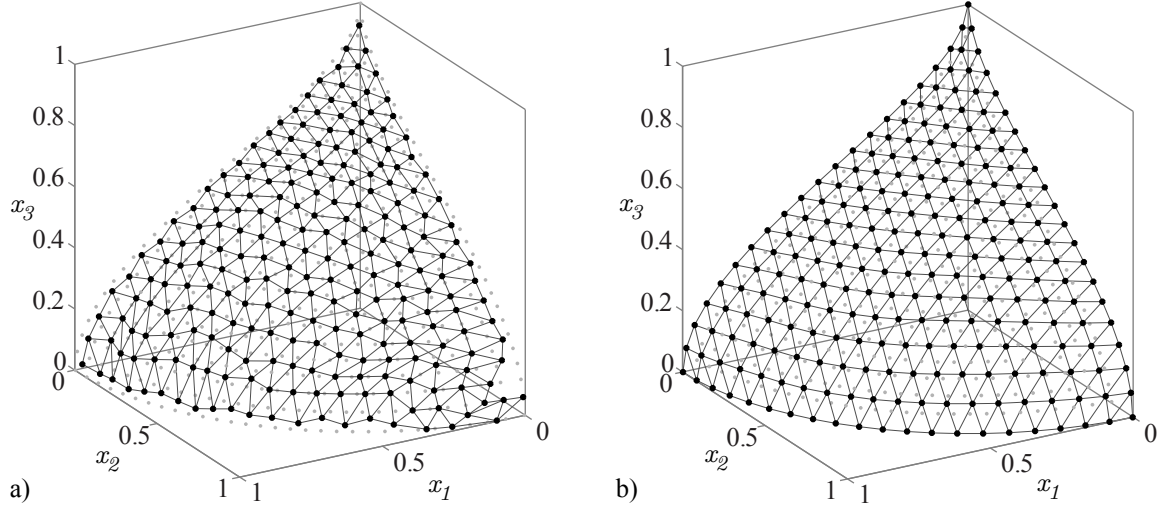


Figure 53: a) “Classical” SOM fit to teardrop Pareto frontier. b) Pareto simplex SOM fit to teardrop Pareto frontier

7.3 Classical Approach to Creating a Self-Organizing Map

Training is the process by which the nodes attain their final codebook vectors, i.e. their final locations in the training data space. The description of the classical training process given here is based on Kohonen’s original approach described in references [77, 78, 79]. Section 7.4 then describes the modifications to the classical approach made in the present work.

7.3.1 Self-Organizing Map Topologies

Before training a SOM, the “topology” of the nodes must be specified. The topology consists of the number of nodes, m , and a symmetric $m \times m$ adjacency matrix whose value is 0 or 1 depending on whether the nodes indexed by the corresponding row/column are connected with an edge.

A common claim regarding SOMs is that they are able to represent a high-dimensional data set in fewer dimensions while “preserving its topology” (see e.g. [60]). However, when the SOM’s topology differs from the data set’s topology, especially because of differing dimensionality, the SOM’s representation will necessarily suffer from some distortion [83].

Consequently, the choice of a particular topology strongly affects the resulting SOM's usability for various goals.

In nearly all applications of SOMs found in the literature, the topology is a 1-D chain or a 2-D regular grid of nodes and edges. One-dimensional topologies may be used to form a non-parametric “curve fit” through a data set, a notional example of which is shown in Figure 54. McClain et al. have demonstrated the use of this technique for approximating the shape of ice accumulations on wing leading edge [97]. Another application of 1-D SOMs is for generating space filling (Peano) curves in higher dimensional data sets (see e.g. [137]). The two most common topologies for two-or-more-dimensional SOMs are the hexagonal and square grids, the 2-D versions of which illustrated in Figure 55. When the SOM is being used to visualize a high-dimensional data set, the 2-D hexagonal grid is applied almost exclusively. Higher-dimensional SOMs are uncommon, but may be used in particular applications, such as vector quantization, that do not require visualization of the SOM.

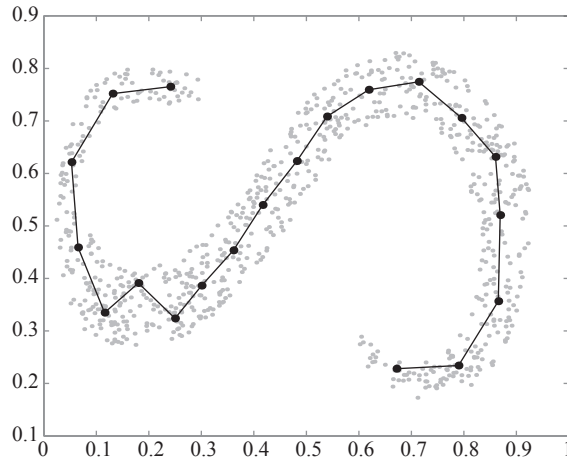


Figure 54: A 1-D, 20-node SOM trained to an S-shaped data distribution.

The number of nodes in the topology, and the topology’s “aspect ratio,” must be chosen based on the user’s particular goals and what is known about the training data. If the user has knowledge of the training data set, the aspect ratio may be chosen to match the distribution of the training data. For example, the aspect ratio could be selected as

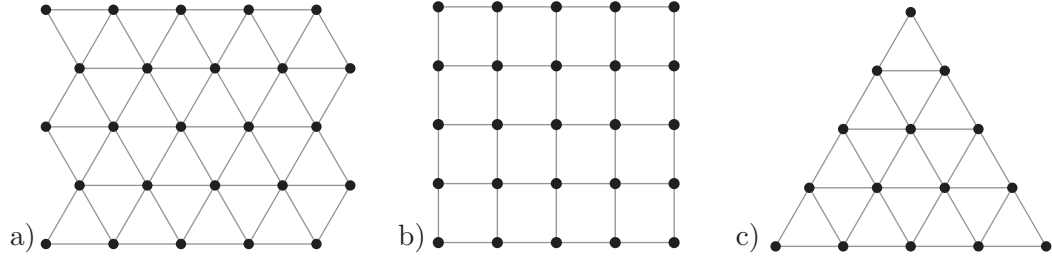


Figure 55: Example 2-D self-organizing map topologies: a) hexagonal, b) square, c) simplex

the ratio of the largest principal components, although for complicated data distributions such as in Figure 54, the principal components may be misleading. The number of nodes generally selected to be small compared to the number of training data points because, as in Figure 52, a common goal of using SOMs is to aggregate the training data in order to represent it more simply [79]. With too many nodes, the SOM may over-fit the training data and lose its smoothing properties. The training process is computationally inexpensive, and the required training time is generally not a constraint when selecting the number of nodes.

7.3.2 Training the Self-Organizing Map

The training process begins by initializing each node's codebook vector. Kohonen suggests two options: the codebook vectors may be randomly assigned, or the nodes of a k -dimensional topology may be uniformly distributed along the hyperplane spanned by the k largest principal components of the data set [79]. The latter option, called linear initialization, can greatly reduce the number of training steps required for the SOM to converge; however, even with random initialization the SOM will converge to a similarly ordered final state.

Kohonen proposes two algorithms for iteratively updating the codebook vectors. The first is an incremental updating process in which a single training data point is considered at each increment. This algorithm is inefficient, but it more clearly demonstrates the updating process. The second algorithm is a batch training algorithm in which all training data points are considered simultaneously. The batch training method is used by essentially all modern SOM implementations. Both algorithms are described below.

7.3.2.1 Incremental SOM Training

The incremental training algorithm considers a sequence $\{\mathbf{x}(t)\}$ of training data points, where t is an index into the sequence. For practical purposes, this sequence may be formed by randomly ordering the set of training data points, and then repeating this ordering n times such that each training data point is eventually presented to the training algorithm n times.

At the t -th training increment, the single training point $\mathbf{x}(t)$ is compared with each codebook vector, \mathbf{m}_i , to find the “winning” codebook vector, \mathbf{m}_c , which is closest to $\mathbf{x}(t)$ [78, p. 110], i.e. \mathbf{m}_c is found by solving

$$\|\mathbf{x}(t) - \mathbf{m}_c\|_2 = \min_i \{\|\mathbf{x}(t) - \mathbf{m}_i\|_2\} \quad (12)$$

In order to ensure the distances in (12) are meaningful, the training data points should first be normalized so that each component of \mathbf{x} has the same dynamic range.

Next, every codebook vector is updated according to the equation

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t) (\mathbf{x}(t) - \mathbf{m}_i(t)) \quad (13)$$

In this equation, $h_{ci}(t)$ is called the “neighborhood function”; its purpose is to scale the move that \mathbf{m}_i makes in the direction of $\mathbf{x}(t)$. For example, when $h_{ci}(t) = 1$, $\mathbf{m}_i(t+1) = \mathbf{x}(t)$ and when $h_{ci}(t) = 0$, $\mathbf{m}_i(t+1) = \mathbf{m}_i(t)$. The neighborhood function’s value depends on the “topological distance” from node i to the winning node c , denoted $\|r_c - r_i\|$, which is simply the minimum number of edges in the topology that must be traversed to move from node i to node c ; consequently the topological distance is always an integer. Figure 56 illustrates the topological distance for a hexagonal topology.

A common form of the neighborhood function is [78, p. 111]

$$h_{ci}(t) = \begin{cases} \alpha(t) & \text{if } \|r_c - r_i\| \leq N_d(t), \\ 0 & \text{if } \|r_c - r_i\| > N_d(t). \end{cases} \quad (14)$$

where $N_d(t)$ is the limiting topological distance. Both $\alpha(t)$ and $N_d(t)$ are chosen to be

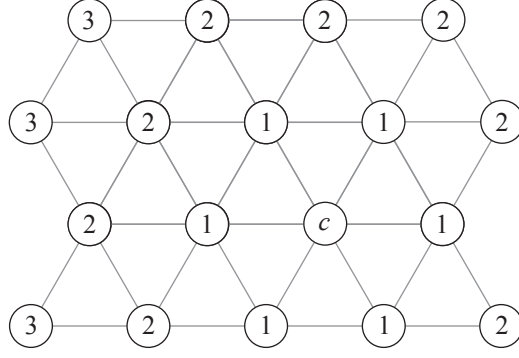


Figure 56: The topological distance of nodes relative to node c .

monotonically decreasing in t . With this simple form, all nodes within the winner's neighborhood (defined by the limiting topological distance) are updated to move the same distance toward $\mathbf{x}(t)$, while nodes outside the neighborhood do not move. An alternative, smoother neighborhood function is the Gaussian function,

$$h_{ci}(t) = \alpha(t) \cdot \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) \quad (15)$$

in which case $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing in t [78, p. 111].

Kohonen describes the training process described above as proceeding in two phases. Both phases use the same training process, described above, but they differ in the extent to which a winning node influences its neighbors. The first phase is the *ordering phase*, in which the influence of the winning node on its neighbors is large, corresponding to a large $N_d(t)$ in Eq. 14 or large $\sigma(t)$ in Eq. 14. In this phase the SOM topology becomes ordered so as to approximate the topology of the training data. When the initial codebook vectors have been chosen to already approximate the topology, for example by using the principal component method, this phase may require relatively few iterations. The second phase is the *fine tuning phase*, in which the winning node has little or no influence on its neighbors, corresponding to a $N_d(t) = 1$ in Eq. 14 or small $\sigma(t)$ in Eq. 14. In this phase the individual nodes work to better model their respective training data points. This phase typically requires an order of magnitude more iterations than the ordering phase [78, p. 112].

7.3.2.2 Batch SOM Training

The batch training algorithm is significantly faster than the incremental algorithm, and all practical SOM implementations use this method. The essential difference of the batch algorithm compared to the incremental algorithm is that in each iteration of the batch algorithm all the training data points are presented to the codebook vectors simultaneously, such that the codebook vector update step accounts for the influence of every training data point simultaneously. Because the training data are presented in a batch, the idea of a sequence of training data points is no longer relevant. Instead, the training proceeds for a set number of iterations, with every training data point considered during each iteration.

The first step in each training iteration is to loop through every point in the training data set and find its nearest codebook vector, which is called the “winner.” This process is delineated in Algorithm 3, in which \mathbf{c} denotes the vector of indices of the winning nodes. The length of \mathbf{c} is equal to the number of training points, as each training point has exactly one winning node. As in the incremental algorithm, the training data should be normalized such that each component x_i has the same dynamic range for calculating the distances.

Algorithm 3 $\mathbf{c} = f(X, M)$

```

1: for all  $\mathbf{x}_i \in X$  do
2:   for all  $\mathbf{m}_j \in M$  do
3:      $dist(j) \leftarrow \|\mathbf{x}_i - \mathbf{m}_j\|_2$ 
4:   end for
5:    $\mathbf{c}(i) \leftarrow \arg \min_j(dist)$ 
6: end for
```

The second step in each training iteration is the update step in which each codebook vector is updated to better model the subset of the training data for which it is the winner. The procedure for this step can be better understood by first assuming that the SOM will indeed converge to a final state. In order for the codebook vectors to converge to a final state, the updates to the codebook vectors must eventually average out to zero as the number of iterations goes to infinity. Letting τ denote the iteration number, we have the requirement that $\mathbf{m}_i(\tau + 1) = \mathbf{m}_i(\tau)$ as $\tau \rightarrow \infty$ [78, p. 138]. By modeling the update term

of Eq. 13 as a random variable we can express this requirement as

$$E(h_{ci}(\mathbf{x} - \mathbf{m}_i^*)) = 0 \text{ as } \tau \rightarrow \infty \quad (16)$$

where \mathbf{m}_i^* is the convergence limit of \mathbf{m}_i and E is the expected value function [78, p. 138]. Denoting by \mathbf{x}_{mj} the mean of all the training points \mathbf{x} whose winner is \mathbf{m}_j , and by n_j the number of training points whose winner is \mathbf{m}_j , Eq. 16 can be rewritten as

$$\mathbf{m}_i^* = \frac{\sum_j n_j h_{ji} \mathbf{x}_{mj}}{\sum_j n_j h_{ji}} \quad (17)$$

According to Eq. 17, the values of \mathbf{m}_i^* are equal to the weighted average of its “neighborhood set” – the subset of the training data for which \mathbf{m}_i is the winner or is in the neighborhood of the winner – with the weighting defined by the neighborhood function h_{ji} . An important property of Eq. 17 is that it cannot be used to directly solve for the final weights \mathbf{m}_i^* because \mathbf{x}_{mj} and n_j are functions of the codebook vectors. Instead, Eq. 17 is applied iteratively over several training iterations. These iterations can be interpreted as being divided into ordering and fine tuning phases as in the incremental training algorithm.

An insightful result of the batch training algorithm is that if h_{ji} is chosen to be equal to $\delta_{i,j}$ (the Kronecker delta function) such that the winning nodes have no effect on their neighbors, the batch training algorithm becomes identical to the Linde-Buzo-Gray algorithm for k -means clustering [85]. Consequently, the effect of using a node topology in SOMs (which becomes irrelevant when $h_{ji} = \delta_{i,j}$) can be understood as being the difference between SOMs and k -means clustering [79].

7.4 *Modified Training for Pareto Simplex Exploration*

As is apparent in Figure 53(a), the classical SOM training method described above does not define a useful coordinate system for the Pareto frontier. However, with minor changes to the algorithm, a very good coordinate system can be obtained, as shown in Figure 53(b). This section describes the particular modifications that have been found to be most effective. The application of these modifications to the original SOM algorithm constitutes the PSSOM training algorithm.

As with NDLCs, this coordinate system is constructed based on the image of the Pareto frontier in the objective space. Consequently, the training data set for PSSOMs consists of the \mathbf{y}^* vectors of a sampled frontier. The corresponding \mathbf{x}^* vectors are not directly used.

7.4.1 The Simplex Topology

For assigning coordinates to the Pareto frontier, the choice of a SOM topology is straightforward: it should match the frontier’s $(k-1)$ -dimensional simplex topology. As is apparent in Figure 55(c), a simplex topology can readily be constructed as a subset of the more commonly used hexagonal topology. Like the hexagonal topology, the simplex topology can be extended to any dimensionality. The only parameter that must be chosen is the number of nodes along each edge of the topology, which must be the same for each edge.

As will be described in subsequent sections, the codebook vectors of the trained SOM form the basis for an interpolating model of the Pareto frontier. The number of nodes per edge thus determines the number of basis points for the interpolation. Unlike the choice of the number of weight levels in the mod-NBI approach, the number of SOM nodes may be large without significantly affecting the overall computational complexity. Consequently, a good strategy is to choose the number of nodes per edge such that the total number of SOM nodes is large but less than the number of points in the sampled Pareto frontier. The reasoning behind this strategy is that a larger number of nodes will generally yield a more accurate continuous model of the frontier; however, in the limit in which the number of nodes equals the number of points on the sampled frontier, the SOM training will typically place the codebook vectors to coincide with the sampled frontier points. This behavior is not necessarily problematic; however, if this number of nodes is exceeded, some of the codebook vectors will necessarily be located at the same frontier point—an arrangement that is incompatible with the node redistribution process described below. As an example, the problem shown in Figure 53 has a 496-point sampled Pareto frontier, corresponding to an NBI search grid with 31 points per edge, and a 231-node SOM, corresponding to 21 points per edge.

In the classical SOM algorithm, each node is indistinguishable from any other node

except by its relative location in the topology, and through training each node attains a particular identity in the form of its final codebook vector. In the Pareto simplex SOM, however, each node is assigned a particular identity *a priori* in the form of a barycentric coordinate tuple that encodes the node’s relative position in the topology. These coordinates are assigned by the trivial interpretation of the topology as a grid in a barycentric coordinate system in which the corner nodes have coordinates of the form $[0, \dots, 0, 1, 0, \dots, 0]$ and nodes in the interior have coordinates that are representative of their relative position within the topology. Consequently, each node is identified by three attributes: a fixed position within the topology, which may be represented with an adjacency matrix and is used in the SOM training process; a corresponding barycentric coordinate tuple, which is used to interpret the trained SOM as a coordinate system for the frontier; and a codebook vector, which is attained through training and represents the node’s final location on the Pareto frontier.

By labeling the nodes with coordinate tuples, the SOM becomes a one-to-one map from a discrete subset of the barycentric coordinate system (the particular coordinate tuples assigned to the nodes) to a discrete subset of the objective space (the corresponding codebook vectors). This mapping captures the essence of the present approach, but it is incomplete. We must extend the mapping to the entire frontier, and ensure that it has the desirable properties of a preference-like coordinate system. These issues are discussed in the remainder of this section.

7.4.2 Initializing the Codebook Vectors

The classical SOM begins by initializing the codebook vectors randomly or according to the principal components of the training data. For training the Pareto simplex SOM, however, a more direct approach is apparent: the initial codebook vectors are placed along the CHIM in exactly the same way as the basepoints for (the unmodified) NBI are arranged. This arrangement ensures that the vertices/edges/faces of the SOM topology begin training aligned with the corresponding subfrontiers. It has been found that this initial arrangement is so effective that only about ten ordering-phase iterations are needed during the batch training process for the SOM to converge.

7.4.3 Sequential Training of Subfrontiers

The classical SOM training algorithm generally avoids placing codebook vectors along the boundaries of the training data set, as each codebook vector is located at the centroid of its corresponding training points. This effect can be seen in Figure 53(a), in which the bounds of the sampled frontier extend beyond the edges of the trained SOM. For the PSSOM, however, we must ensure that the SOM covers the entire Pareto frontier.

The solution to this problem is conceptually the same as the mechanism by which the modified NBI algorithm in Chapter 6 expands the NBI grid to enable sampling from the entire frontier. Rather than train all the SOM codebook vectors simultaneously, the training will proceed in stages: at the i -th stage, the nodes whose barycentric coordinates lie on an i -subfrontier, i.e. the nodes with i nonzero barycentric coordinates, are trained. Thus a k -attribute Pareto frontier requires k stages to train. The nodes that are trained at each stage for a 3-dimensional Pareto frontier using a SOM with 5 nodes per edge are shown in Figure 57, with black points indicating nodes trained during the current stage and gray points indicating nodes that have already been trained.

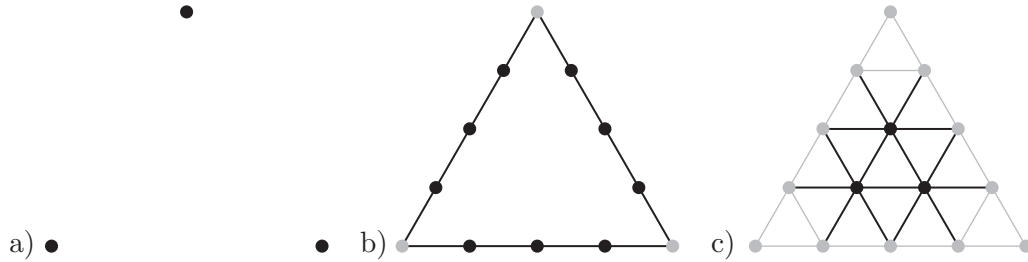


Figure 57: Nodes and edges considered at each stage in Pareto simplex SOM training. a) 1st stage, b) 2nd stage, c) 3rd stage

Correspondingly, the entire sampled Pareto frontier is not used for training during each stage. Rather, only those points on the sampled frontier belonging to a subfrontier of dimensionality *less than or equal to* i are considered in the i -th stage.³ The use of training points on lower-dimensional subfrontiers in addition to those on the i -dimensional subfrontiers may

³See section 7.4.6 for details of a potential complication when determining which training data points lie on subfrontiers.

seem incongruous. It has been observed, however, that by including these additional points the resulting distribution of codebook vectors is more even than if they are excluded.

During the first stage the SOM training algorithm is not actually used. Since there are k nodes being trained to the k corresponding individual optima of the sampled frontier, the codebook vectors are simply placed at the corresponding optima.

The effect of using sequential training of the subfrontiers is illustrated in Figure 58. The sequential training solves the problem of the SOM not filling the entire Pareto frontier, but the spacing of the codebook vectors along the frontier remains uneven.

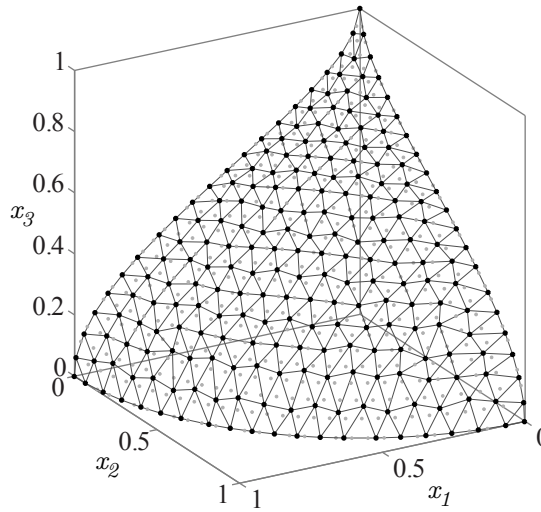


Figure 58: SOM trained to teardrop test problem using hierarchical training of subfrontiers.

7.4.4 Evenly Distributing the Codebook Vectors on the Frontier

As shown in Figure 58, the classical SOM training algorithm does not result in an even distribution of codebook vectors across the Pareto frontier. Although the distribution is acceptable, and clearly representative of the frontier's geometry, it is uneven enough that the resulting coordinate system may be no better than the NDLCs. Two possible approaches for forming a better coordinate system from the codebook vectors are apparent: 1) Reassign the barycentric coordinate-vector labels to the nodes such that although the codebook vectors remain unevenly distributed, their corresponding barycentric coordinates have the same distribution. 2) Keep the barycentric coordinates the same and modify the codebook

vectors to be more evenly spaced on the Pareto frontier.

The first approach could be implemented by simply measuring the length of each SOM edge that belongs to a given iso-barycentric-coordinate curve, summing these lengths to obtain the total curve length, and reassigning the coordinate values based on the cumulative fraction of the total length at which a given node lies. (The coordinate values of the first and last nodes on this curve are fixed by the subfrontier-membership boundary conditions). Unfortunately, this solution has two problems. First, there is no guarantee that the resulting coordinates of each node would sum to one; they almost certainly would not. This would require an additional normalization as is used in calculating the NDLCs. Second, as seen in Figure 58, the iso-coordinate curves are not straight, particularly near their endpoints. This curvature would cause a distortion of the coordinate values. A second possible solution would be to use a linear programming approach of assigning coordinates, similar to the approach used in Chapter 6 to redistribute the NBI basepoints. However, this approach would likely yield results no better than those of Chapter 6. In the interest of improving on both the NDLC and mod-NBI methods of assigning coordinates, the second approach, in which the codebook vectors are redistributed, is taken here.

The codebook redistribution approach works by attempting to make every edge that connects to a given node have equal length. Algorithm 4 describes the steps for redistributing the codebook vectors. In this algorithm, N_i is the set of codebook vectors that are connected to \mathbf{m}_i by an edge in the SOM topology and either belong to the same subfrontier as \mathbf{m}_i or have already been trained in a previous stage as described in the previous section.⁴

The general approach of this algorithm is conceptually simple. For each codebook vector, \mathbf{m}_i , the edges connecting \mathbf{m}_i to its topological neighbors are considered as a set. The average length of these edges is calculated and subtracted from their individual lengths in order to determine how much each edge is longer or shorter than the average length. The codebook vector \mathbf{m}_i is then moved in such a way as to shorten the edges that are longer than average and lengthen the edges that are shorter than average. The final move vector is

⁴In this context, two codebook vectors being on the same subfrontier is defined by having identical patterns of 0-values in their corresponding barycentric coordinate vectors.

Algorithm 4 Codebook vector redistribution; $\mathbf{m}_{new} = f(\mathbf{m})$

```
1: for all  $\mathbf{m}_i$  do
2:   for all  $\mathbf{m}_j \in N_i$  do                                ▷ For each edge that connects to  $\mathbf{m}_i \dots$ 
3:      $\mathbf{edge}_{ij} = \mathbf{m}_i - \mathbf{m}_j$ 
4:      $\mathbf{edgeLength}_{ij} \leftarrow \|\mathbf{m}_i - \mathbf{m}_j\|$                 ▷ Calculate length of edges
5:      $\mathbf{unitEdge}_{ij} = \mathbf{edge}_{ij} / \mathbf{edgeLength}_{ij}$         ▷ Calculate unit vector of edges
6:   end for
7:    $\mathbf{meanLength} \leftarrow \text{mean}(\mathbf{edgeLength})$ 
8:    $\mathbf{move} = [\mathbf{0}]$                                           ▷ Initialize  $\mathbf{move}$  to 0 vector
9:   for all  $\mathbf{m}_j \in N_i$  do
10:     $\mathbf{diffLength}_{ij} = \mathbf{edgeLength}_{ij} - \mathbf{meanLength}$     ▷ Is edge too long or too short?
11:     $\mathbf{move}_i = \mathbf{move}_i + \mathbf{diffLength}_{ij} \cdot \mathbf{unitEdge}_{ij}$  ▷ Move to shrink/stretch this edge
12:   end for
13: end for
14: for all  $\mathbf{m}_i$  do
15:    $\mathbf{m}_{i,new} = \mathbf{m}_i + \gamma \cdot \mathbf{move}_i$                 ▷ Move a small step toward equalizing edge lengths
16: end for
```

expressed as a sum of individual move vectors, each of which points in the direction of one edge (i.e. toward the \mathbf{m}_j that this edge connects to \mathbf{m}_i) and has a magnitude equal to the difference between the edge's length and the average edge length. As indicated by line 15 of Algorithm 4, the final moves are applied after the move of each \mathbf{m}_i has been calculated, and the move vector is scaled by a factor $\gamma < 1$ before it is added to \mathbf{m}_i . If $\gamma = 1$, then $\mathbf{m}_{i,new}$ would yield equal length edges if only a single \mathbf{m}_i were being relocated. However, because each codebook vector is being moved, each edge's length is affected by two moves, one for each of its endpoint nodes. Consequently, applying the full move ($\gamma = 1$) to every codebook vector will not yield equal edge lengths. This fact requires that Algorithm 4 be applied iteratively in order for the edge lengths to eventually converge, but it does not necessarily imply that a value for γ other than 1 would improve the convergence. Experiments, however, have shown that $\gamma = 1$, often leads to changes that are so large that the redistribution never converges (rather, the codebook vectors are pushed “out of the plane” of the frontier and diverge). The best convergence has been observed when a small fractional move is taken for each repetition of Algorithm 4, for example $\gamma = 0.05$, and the algorithm is repeated approximately $2/\gamma$ times. This approach generally leads the codebook vectors to converge to evenly distributed points across the frontier.

The effect of Algorithm 4 on a 1-D topology is illustrated in Figure 59. The gray points

show the path taken by the center codebook from its initial point, $\mathbf{m}_{initial}$, to its final point, \mathbf{m}_{final} , at which the edges have equal length. Note that the sum of the lengths of the two edges stays nearly equal at each step, so that the path of the gray points is almost an elliptical arc. This behavior helps ensure that the redistribution keeps the codebook vectors very close to the Pareto frontier, i.e. it avoids solutions that achieve an equal edge length by simply pushing \mathbf{m}_{final} a very large distance away from the frontier.

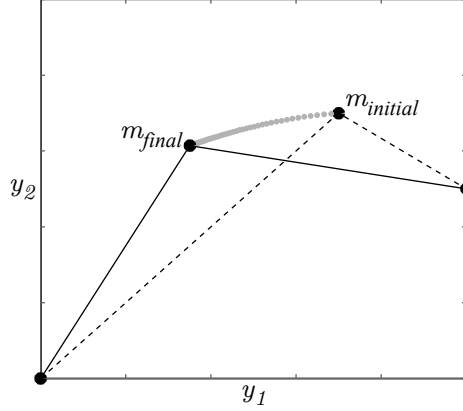


Figure 59: Notional codebook vector redistribution, showing iterations from $\mathbf{m}_{initial}$ to \mathbf{m}_{final} , at which the edges have equal length

The redistribution algorithm is applied at the end of each subfrontier-training stage and is applied only to the nodes that were trained during that phase. These nodes and the associated edges are shown in black in Figure 57 on page 150. An important point must be emphasized about which edges are considered in the set N_i at each stage. The edges that compose the set N_i are determined by the union of the sets of edges that connect nodes on the *individual* subfrontiers being considered; this is different than simply taking all edges that connect any points that were trained in this stage. In particular, if an edge connects two nodes on *different* subfrontiers of the same dimensionality, this edge is *never* included in the set N_i . This important exclusion ensures that Algorithm 4 works as intended. As an example, Figure 60 indicates the edges that are never included in N_i for a 2-D simplex topology.

In some cases, it has been observed that the redistribution algorithm does not converge at a few points on the Pareto frontier. This typically happens in regions where the Pareto

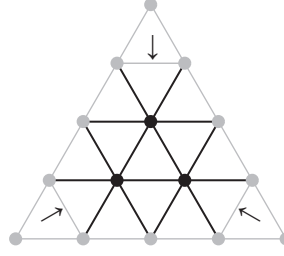


Figure 60: Detail of Figure 57 with arrows indicating edges that are never included in N_i

frontier tapers sharply (such as at the top of the teardrop Pareto frontier), as this causes some nodes to have edges that are both short (along the narrow axis of the frontier) and long (along the opposing axis). For these few points the algorithm described above tends to push the points away from the training data in such a way that all of the nodes' edges become longer.

In order to minimize any complications that may arise from this effect, it is recommended that at each iteration of the redistribution algorithm, the mean of the absolute values of all moves be computed and stored. While the redistribution is converging, this value will decrease at each iteration. It is thus recommended that the process be stopped when this value begins to increase. As indicated above, this has been observed to generally occur after approximately $2/\gamma$ repetitions.

7.4.5 Summary of SOM Training Process

The full SOM training process is summarized graphically in Figure 61. In (a), vertices of the untrained SOM grid have been placed at the corresponding individual optima of the sampled frontier. The remaining codebook vectors are linearly distributed between the individual optima. In (b), the 1-D subfrontiers have been trained, while the interior nodes remain in their initial positions. In (c), the codebook vectors have been redistributed along the 1-D frontiers to be more evenly spaced. In (d), the 1-D frontiers' codebook vectors have been locked in place, and the interior codebook vectors have been trained to the full sampled frontier. Finally, in (e), the interior codebook vectors have been redistributed, resulting in an even grid on the entire Pareto frontier.

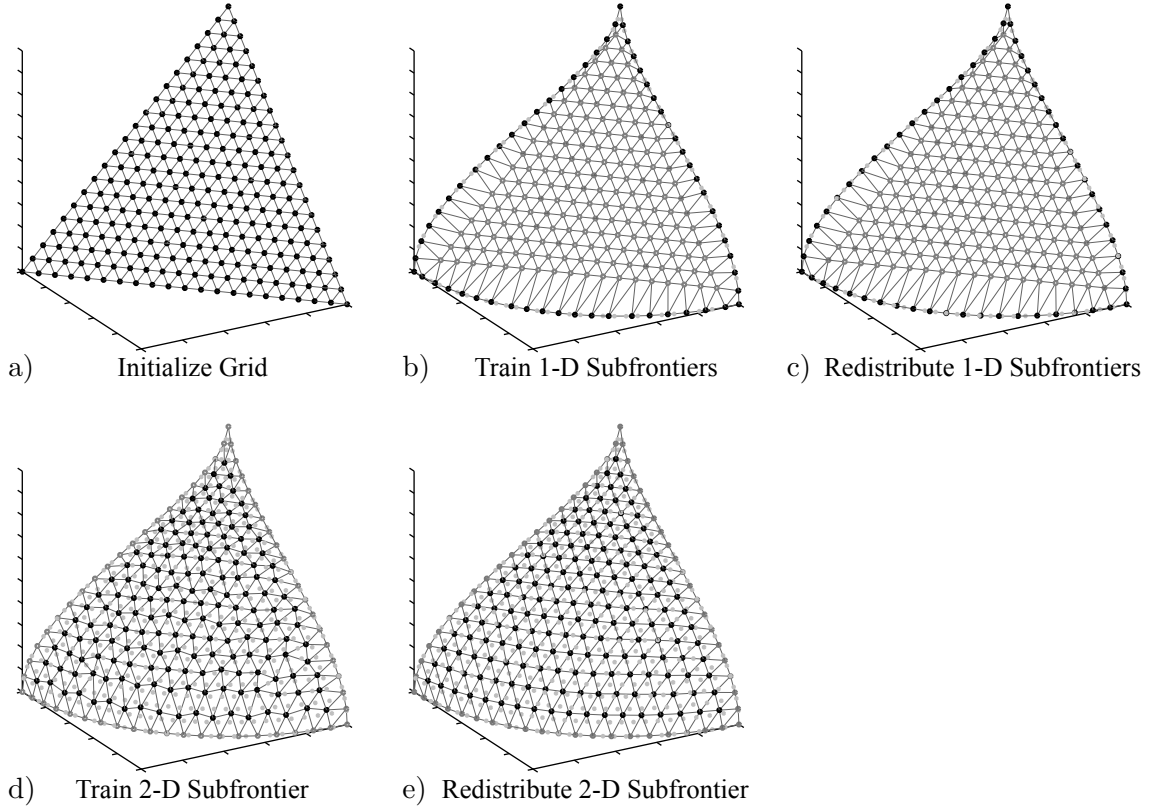


Figure 61: Visualization of Pareto simplex SOM training process

7.4.6 Complications of Identifying Sampled Subfrontiers

The PSSOM method requires the points on the sampled frontier that belong to subfrontiers to be identified so that they can be used as training data during the appropriate stages. Typically this can be accomplished by checking the Pareto dominance criteria after projecting the sampled data into the appropriate lower-dimensional space. In some cases, however, applying the dominance criteria to the sampled frontier will result in points being identified as being on a subfrontier when a simple visual examination reveals that they are on the interior of the frontier.

This problem is demonstrated in Figure 62a for the teardrop frontier sampled using the modified NBI algorithm. Here, ten points on the interior have been identified as belonging to the subfrontiers. By examining the sampled frontier, however, we can easily infer that these ten points should not actually be counted as belonging to the subfrontiers. Clearly

there *should be* points that dominate these; those points just have not been sampled.

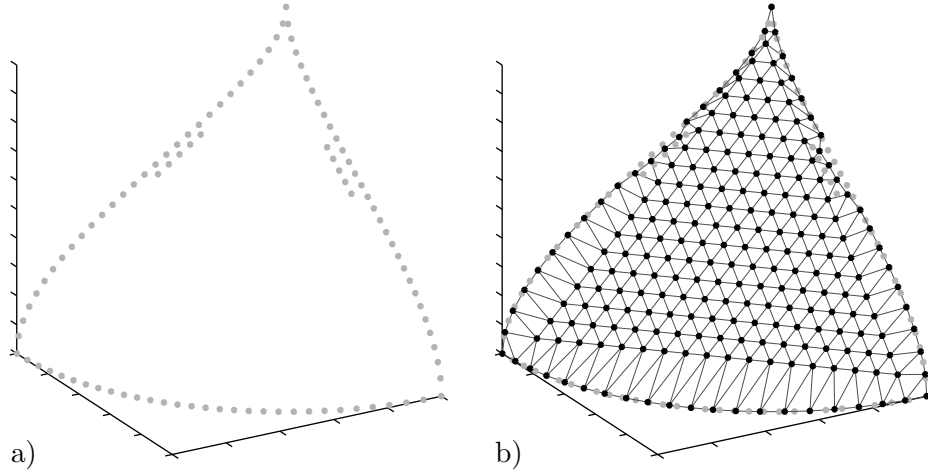


Figure 62: a) 1-D subfrontiers found by applying dominance criteria to projected frontier
b) Effect on SOM training, cf. Figure 61c

The effect of including these points in the subfrontiers while training the SOM is shown in Figure 62b. Their inclusion causes the edges of the SOM to taper inward, distorting the shape of the underlying coordinate system. Two approaches can be used to help avoid this complication. First, before applying the Pareto dominance criteria to identify the subfrontiers, the sampled frontier may be projected onto the CHIM hyperplane as described in Section 5.4. This generally results in a modified distribution of the sampled data that better identifies the true subfrontiers. If projecting the data is ineffective, then the recommended approach is to resample the subfrontiers using separate multi-objective optimization runs instead of simply applying the dominance criteria to the sampled (full-dimensional) Pareto frontier. This second approach is the most effective, and essentially guarantees that the SOM will be trained to the true subfrontiers; however, for high-dimensional Pareto frontiers the number of multi-objective optimization runs becomes large.

7.5 Associating SOM Coordinates with the Sampled Pareto Frontier

If the user desires to explore the Pareto frontier in only the objective space, with a function of the form $\mathbf{y} = g(\boldsymbol{\beta})$, then the trained SOM created in the previous section contains all the information that is needed. However, if a function of the form $(\mathbf{x}, \mathbf{y}) = g(\boldsymbol{\beta})$ is desired, an

additional step that connects the SOM's barycentric coordinates with the design space is needed.

This section describes a process for combining the two mappings we have available – the sampled Pareto frontier, which relates \mathbf{x} to \mathbf{y} , and the SOM nodes, which relate \mathbf{y} to $\boldsymbol{\beta}$ – to form a basis data set that relates \mathbf{x} , \mathbf{y} , and $\boldsymbol{\beta}$. The strategy taken here is to project each point in the sampled frontier onto the *SOM manifold* – an abstraction that “fills in” the empty space between the codebook vectors to interpret the trained SOM as forming a faceted (hyper)surface. Because each cell⁵ of the SOM topology is itself a simplex, each point on the SOM manifold can be defined as a weighted sum of some cell's vertices. Figure 63 illustrates this concept. The point \mathbf{y}_0 is in the cell whose vertices are \mathbf{m}_1 , \mathbf{m}_2 , and \mathbf{m}_3 . Consequently, $\mathbf{y}_0 = \alpha_1 \mathbf{m}_1 + \alpha_2 \mathbf{m}_2 + \alpha_3 \mathbf{m}_3$ with $\sum \alpha_i = 1$ and $\alpha_i \geq 0$. After solving for the α_i weights, the point \mathbf{y}_0 can be assigned a tuple of barycentric coordinates that encode its relative position within the entire SOM: $\boldsymbol{\beta}_{y_0} = \alpha_1 \boldsymbol{\beta}_1 + \alpha_2 \boldsymbol{\beta}_2 + \alpha_3 \boldsymbol{\beta}_3$.

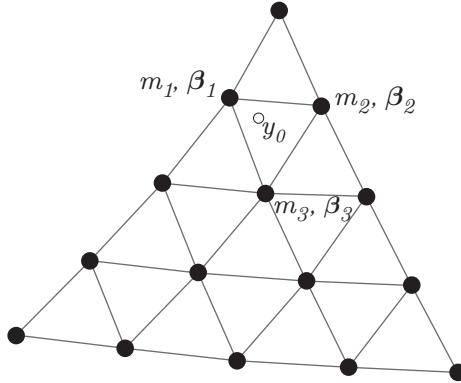


Figure 63: SOM manifold. Point \mathbf{y}_0 can be expressed as a weighted sum of \mathbf{m}_1 , \mathbf{m}_2 , \mathbf{m}_3

Each point on the sampled frontier can thus be projected onto the SOM manifold and represented in the form $\mathbf{y}_{proj} = \alpha_1 \mathbf{m}_1 + \dots + \alpha_k \mathbf{m}_k$. The strategy taken here is to find, for each point on the sampled frontier, the closest projected point on the SOM manifold. This point is expressed as a weighted sum of the codebook vectors, $\mathbf{m}_1, \dots, \mathbf{m}_k$, that are the vertices of some SOM cell, and the barycentric coordinates of this point are calculated by using those

⁵In this context, a *cell* is one of the $(k - 1)$ -dimensional simplices enclosed by edges/faces that connect exactly k nodes in the SOM topology.

same weights in a weighted sum of the vertices' barycentric coordinates, β_1, \dots, β_k . Each point on the sampled frontier then inherits the barycentric coordinates of its projection on the SOM manifold. Since the points on the sampled frontier have corresponding preimages in the design space, \mathbf{x} , we can use these barycentric coordinates to create a function of the form $(\mathbf{x}, \mathbf{y}) = g(\beta)$.

The calculation of the projection of each point in the sampled frontier onto the SOM manifold is non-trivial and is complicated by the fact that the SOM manifold may be non-convex. In general, we must project each sampled point onto every cell of the SOM and then check which cell contains the closest projected point. The distances from each point on the sampled frontier to each cell are then calculated, and the closest projection for each sampled point is identified. As described below, this process can be expedited by ruling out (and therefore not projecting onto) cells that cannot possibly contain the closest overall projected point.

Given a point on the sampled frontier, \mathbf{y}_0 , and a cell of the SOM defined by its vertices, $\mathbf{m}_1, \dots, \mathbf{m}_k$, the calculation of the closest projection onto the cell can be found by solving the quadratic programming problem

$$\begin{aligned} \min_{\alpha} \quad & d = \|\mathbf{y}_0 - [\mathbf{m}_1 \dots \mathbf{m}_k]^T \alpha\| \\ \text{s. t.} \quad & \sum_{i=1}^k \alpha_i = 1 \\ & \alpha_i \geq 0 \end{aligned} \tag{18}$$

The closest projection on the cell is then equal to $[\mathbf{m}_1 \dots \mathbf{m}_k]^T \alpha$ and its distance from \mathbf{y}_0 is d . Problem 18 can be converted to a standard quadratic programming form

$$\begin{aligned} \min_{\alpha} \quad & d = \alpha^T M^T M \alpha - 2(M^T \mathbf{y}_0)^T \alpha \\ \text{s. t.} \quad & [1 \dots 1] \alpha = 1 \\ & I \alpha \geq 0 \end{aligned} \tag{19}$$

where M is the matrix whose columns are the vertices of the SOM cell, $\mathbf{m}_1, \dots, \mathbf{m}_k$.

As described above, this quadratic program would generally be solved in a double loop in which each point on the sampled frontier is compared to each SOM cell. However, the

inner loop can be simplified by only solving the program for those cells that could possibly contain the overall closest point. This can be done by determining upper and lower limits on the distance from a sampled point to a given cell. Then any cell whose lower-limit distance is greater than the smallest (as calculated over all cells) upper-limit distance cannot possibly contain the overall closest point.

Simple rules for calculating upper and lower limits on the distance from a point to a SOM cell based only on the cells vertices are:

- *Upper limit* The distance from a sampled point to a cell cannot be larger than the distance from the sampled point to the nearest vertex of the cell.
- *Lower limit* The distance from a sampled point to a cell cannot be smaller than the distance from the sampled point to the nearest vertex of the cell minus the length of the longest dimension of the cell.

The upper limit rule is self evident. The lower limit rule is an overly conservative bound based on sweeping out a hypersphere in the objective space that is centered at the closest vertex of the SOM cell and whose radius is equal to the longest dimension of the SOM cell. The rule states that the closet point on the cell cannot possibly be closer than the closest point on this hypersphere, at which the distance is equal to the bound stated in the rule. This distance is illustrated conceptually in two dimensions in Figure 64. The longest dimension of the cell, in this case \overline{ab} , is always equal to the longest edge connecting two vertices of the cell; thus it is easily calculated from just the vertex positions.

The true lower limit distance is necessarily lower than the limit imposed by the above rule. Suppose, for example, that \overline{ab} pointed directly back toward the sampled point. Then b would be nearer than a , contradicting the original premise that a is closest. However, considering that the precise lower limit is in fact the solution to the quadratic program, this rule seems to strike good balance between the computational effort required to carefully bound the limits in order to avoid solving certain quadratic programs, and the computational effort of simply solving those quadratic problems.

Even with these overly conservative limits, the range between the upper and lower

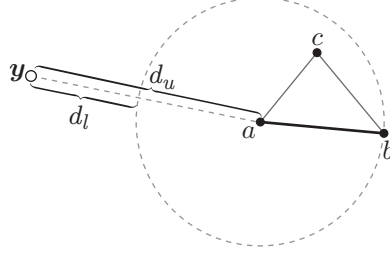


Figure 64: Upper (d_u) and lower (d_l) limits of distance from a point, y , to the SOM cell $\triangle abc$.

limits is generally quite small – the SOM training process ensures this – and the vast majority of cells can be ruled out beforehand. As an example, solving the projection of the 496 sampled frontier points in Figure 53(b) onto the trained SOM required approximately 16,900 quadratic programs, an average of 34 per sampled point versus the maximum of 413 SOM cells. The projection and coordinate assignment process for all 496 points was solved in approximately 15 seconds on a modern, high-end desktop computer.

7.6 Application to Example Problems

Pareto simplex self-organizing maps were created for the example problem frontiers sampled using NSGA-II and mod-NBI. Before creating the PSSOMs, each sampled frontier was normalized such that each attribute ranged from 0–1. This is primarily relevant to the truncated sphere problem and the concave problem. To resolve the non-unique optimum for y_3 in the truncated sphere problem, the codebook vector with coordinates $(0, 0, 1)$ was placed at $\mathbf{y} = [0.4243, 0.4243, 1]$, which is the center of the constrained upper edge of the frontier.

Each example problem’s NSGA-II sampling comprises a 10,000 point population optimized to all three attributes, and three 200 point populations optimized to the three two-attribute subfrontiers, for a total of 10,600 points. While training the two-attribute subfrontier nodes of the PSSOM, only the corresponding 200 points were used. All 5600 points were while training the nodes on the three-attribute frontier, i.e. the nodes with all non-zero coordinates. Separate subfrontier training sets were used to avoid the problems described in Section 7.4.6 and because NSGA-II does not explicitly seek points on the

subfrontiers, resulting in jagged edges that would negatively affect the PSSOM. (See, for example, the boundaries of the sampled sets in Figure 39.)

The mod-NBI samplings are based on a weight grid with 51 points per edge, which results in 1275 total sampled points. The globally dominated sampled points for the truncated sphere and bumpy sphere problems were discarded before training the PSSOMs. (These points are shown in Figure 49.) In order to avoid the problems described in Section 7.4.6, the training data sets for the two-attribute subfrontiers were limited to those sampled points whose mod-NBI weights place them on the corresponding subfrontier.

The PSSOMs were trained using simplex node topologies with 21 points per edge for all example problems except the bumpy sphere, for which 41 points per edge were used to better capture its small-scale features. A step neighborhood function was used in which the neighbors of the winning nodes (i.e. those nodes whose topological distance is less than the limiting neighborhood distance) are moved half the distance that the winning node itself is moved. Nodes whose topological distance exceeds the limiting neighborhood distance are unaffected by the winning node. The limiting neighborhood distance was initially set to 8, and decreased linearly to 0 over 300 training iterations. These were followed by up to 40 redistribution iterations with $\gamma = 0.5$, which may have been stopped early according to the convergence criterion defined at the end of Section 7.4.4.

The final trained PSSOM topologies are plotted in the objective space in Figure 65 for the NSGA-II sampled frontiers and in Figure 66 for the mod-NBI sampled frontiers. The resulting maps are extremely similar, and aside from the truncated sphere, are nearly identical. This similarity demonstrates PSSOMs insensitivity to the distribution of the sampled frontier points, which is one of its main benefits over either the NDLCs or the mod-NBI weights. For either sampling method, the PSSOM is seen to have difficulty mapping the upper portion of the truncated sphere, with many nodes being squeezed into the region near the $y_3 < 0.8$ constraint. As with the NDLCs and mod-NBI weights, however, this effect is largely confined to the region around the constraint and does not negatively affect the PSSOM in regions away from the constraint.

Figure 67 shows the NSGA-II sampled frontiers colored according to the PSSOM coordinates. These plots generally show the same large-scale trends as the NDLCs plotted in Figure 39, but are significantly less noisy. This indicates that the linearity of the coordinate system over small scales is improved by the PSSOM coordinates. An important difference between the PSSOM coordinates and the NDLCs can be seen in the bumpy sphere plots in Figure 67. Whereas the NDLCs accounted for the presence of holes in the frontier by conforming around the holes, the PSSOM coordinates are less effective at accounting for the holes. This is seen in Figure 67 by the relative straightness of the iso-coordinate curves in the regions immediately surrounding the holes in the frontier.

Figure 68 shows the mod-NBI sampled frontiers colored according to the PSSOM coordinates. At a glance, the results look comparable to the results of using the mod-NBI weights as coordinates in Figure 50, and it is not immediately apparent that the PSSOM has created a better coordinate system. A more revealing look at the PSSOM coordinates is shown in Figure 69, which plots the mod-NBI sampled points on ternary grids of the PSSOM coordinates. If these sampled points were plotted on a ternary grid of the mod-NBI weights, then by definition each plot would show a regular grid of points. The extent to which the points in Figure 69 differ from a regular grid is representative of the difference between using the PSSOM coordinates versus using the mod-NBI weights as coordinates. Most interesting is Figure 69a, which clearly reveals the mod-NBI sampling pattern of increased sparsity near the center of the frontier. The presence of this pattern reveals that the PSSOM coordinate system has accounted for, and corrected for, the slightly irregular mod-NBI sampling.

The resulting PSSOM coordinate systems are extremely well behaved, and have almost no irregularities. The primary cause of difficulty in the PSSOM method seems to be frontier geometries that taper to sharp points, such as at the tips of the concave frontier or the top of the teardrop frontier. As described in Section 7.4.4, this appears to be a result of the very different edge lengths along the different axes of the frontier, which cause the nodes to push outward during the distribution process. The final effect on the coordinate system is minor; it is essentially undetectable in Figures 68 and 67, but it can be seen in Figure 69d which

shows a gap at the top of the grid which corresponds to the tapered point of the teardrop frontier. The effect of one node in this region pushing outward from the local plane of the frontier is to correspondingly push the coordinates of sampled points near this node toward the neighboring nodes, leaving a small gap in Figure 69d.

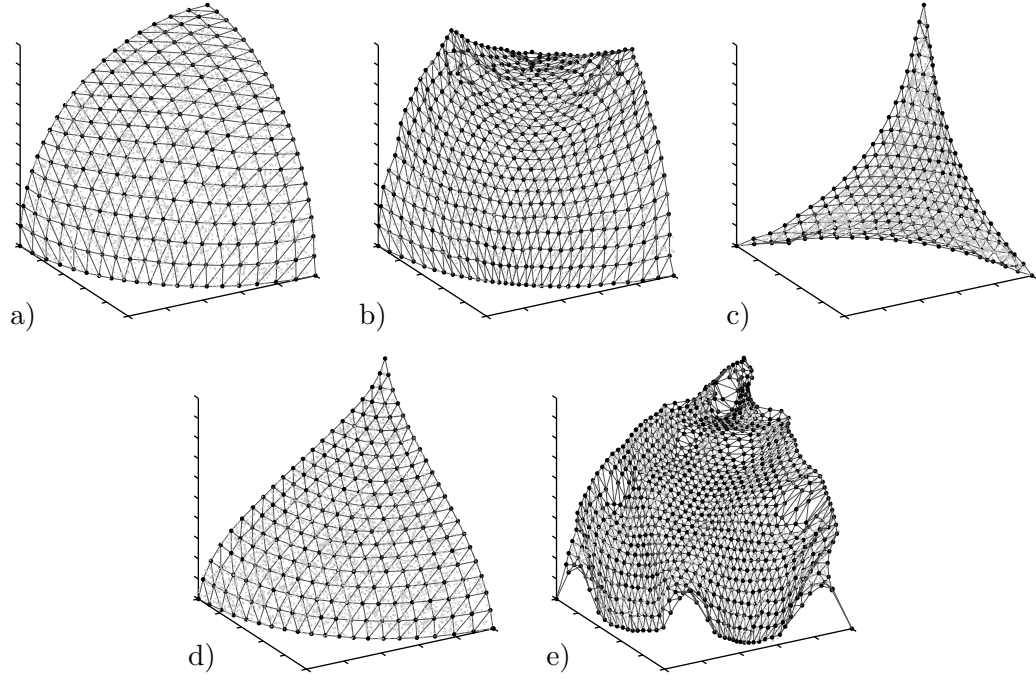


Figure 65: Final trained SOMs for exemplar Pareto frontiers sampled with NSGA-II. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

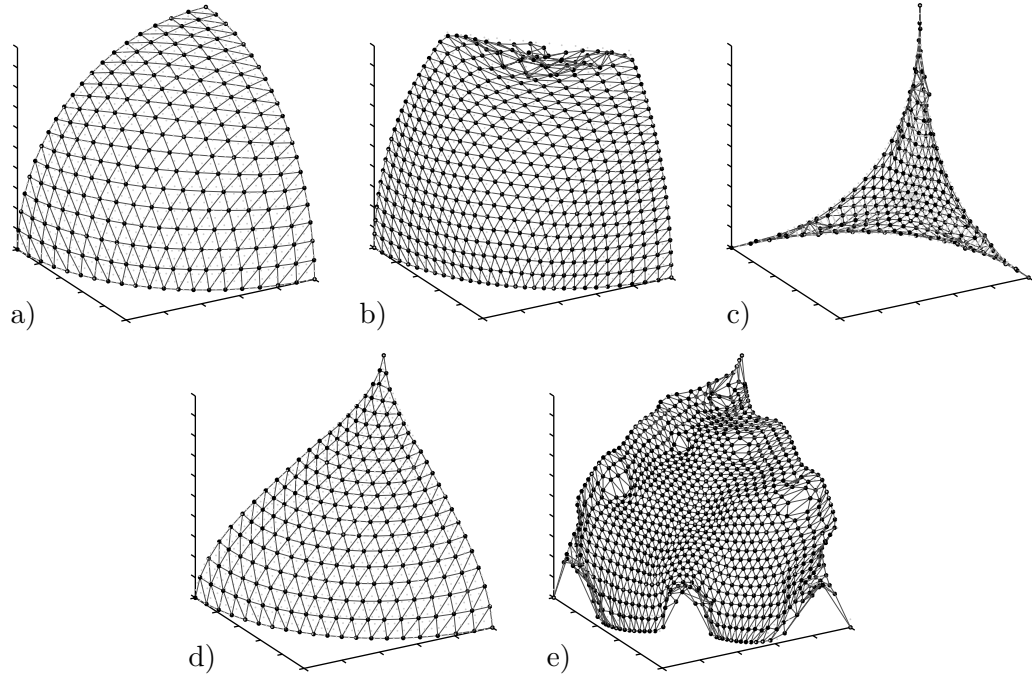


Figure 66: Final trained SOMs for exemplar Pareto frontiers sampled with mod-NBI. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

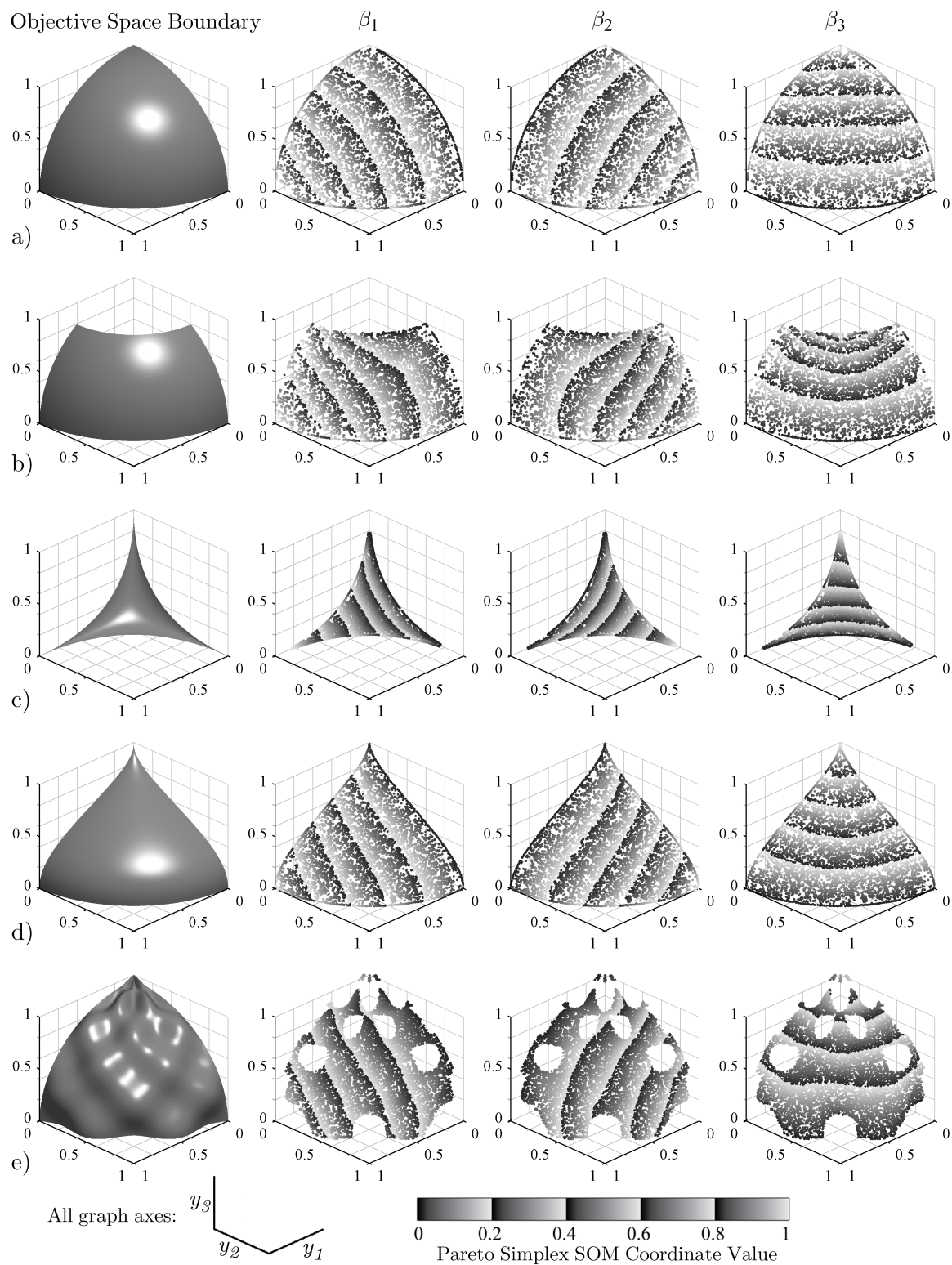


Figure 67: Example problem boundaries and sampled frontiers; sampled by NSGA-II, colored by corresponding PSSOM coordinates. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

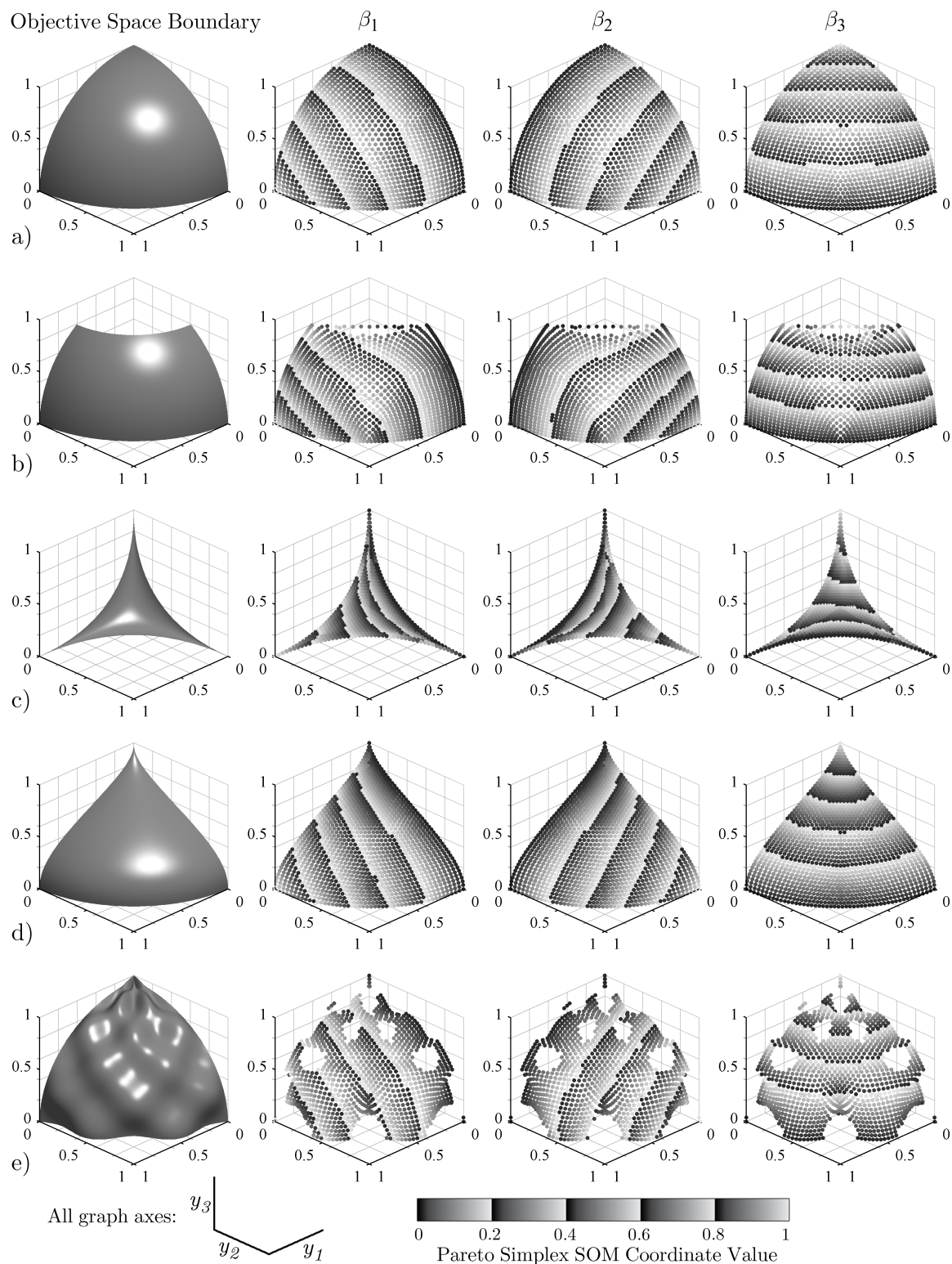


Figure 68: Example problem boundaries and sampled frontiers; sampled by mod-NBI, colored by corresponding PSSOM coordinates. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

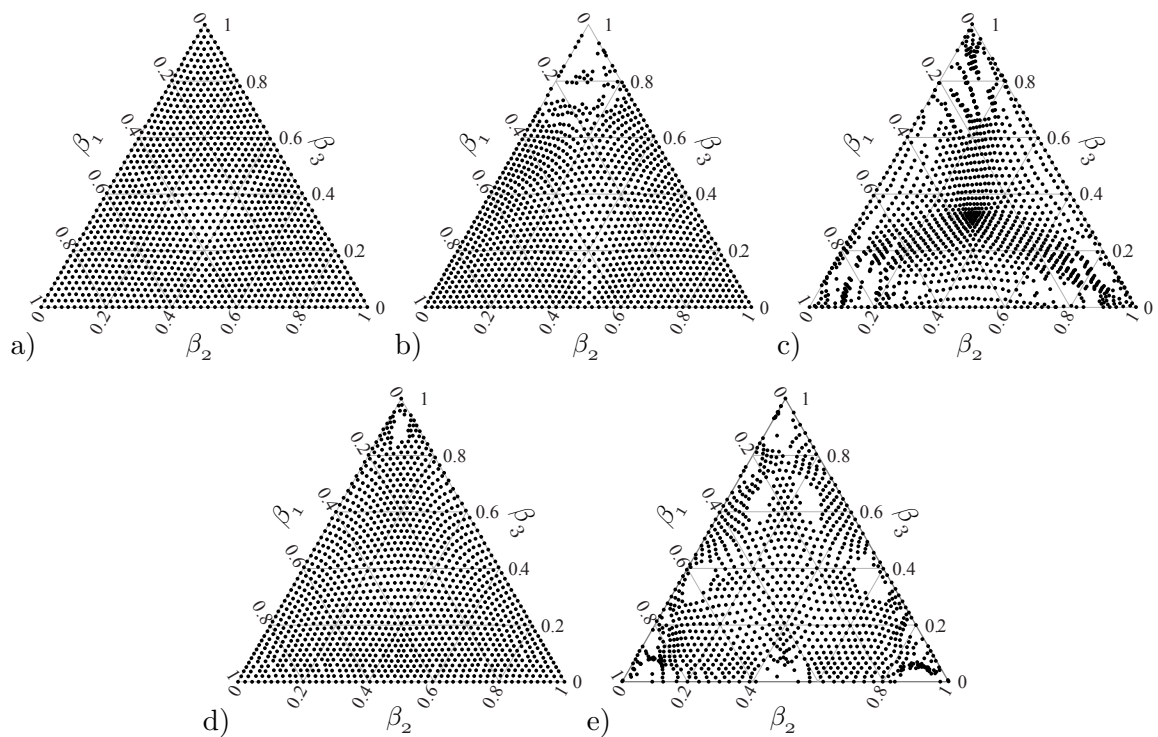


Figure 69: Distribution of mod-NBI sampled frontiers in PSSOM coordinate spaces. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

CHAPTER VIII

RESULTS

The motivation for developing the coordinate systems in Chapters 5–7 is to enable new design space exploration methods that leverage the coordinates as independent variables. This chapter describes the realization of these methods, first by creating interpolating functions of the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$, and then by exploring these functions visually to learn about design problems. Throughout this chapter, the four methods for constructing coordinate systems that were introduced in previous three chapters are compared: non-domination level coordinates based on NSGA-II sampling, Pareto simplex self-organizing map coordinates based on NSGA-II sampling, mod-NBI weight coordinates based on a mod-NBI sampling, and Pareto simplex self-organizing map coordinates based on mod-NBI sampling. (As demonstrated at the end of in Chapter 6, it would also be possible to construct an NDLC coordinate system based on a mod-NBI sampling, however this method is not demonstrated here because it generally performs worse than simply using the mod-NBI weights.) This chapter thus serves the dual purpose of illustrating exploration techniques that may be useful in a Pareto Simplex Exploration approach and using these techniques to assess the quality of these four coordinate systems.

8.1 Modeling the Pareto Frontier with Radial Basis Functions

Having completed Steps 1 and 2 of the Pareto Simplex Exploration approach in the previous three chapters, the next step is to use the sampled frontiers and their associated coordinates to create interpolating functions of the form $\mathbf{y} = g(\boldsymbol{\psi})$. As described in Section 4.2, these functions can be composed with functions of the form $\mathbf{x} = h(\mathbf{y})$. However, because the example problems considered in this section have trivial mappings from the design variables to the attributes, only the functions $\mathbf{y} = g(\boldsymbol{\psi})$ are considered here. Section 8.3 introduces a more interesting wing design example problem, which will be used to demonstrate the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$.

The interpolating functions used here are simple radial basis function networks (RBFs). One RBF is created for each attribute or design variable to be modeled. The RBFs are created by first calculating the square matrix of distances, R , from every barycentric coordinate tuple that has been assigned to a sampled frontier point to every other such barycentric coordinate tuple. For the purpose of this calculation, the Euclidean distance is used.

$$R_{i,j} = \|\psi_i - \psi_j\|$$

These distances are next transformed using a particular basis function. For the present work, the trivial basis function $\Phi(r) = r$ is used. This has been found to give good results and there is no apparent incentive to use a more complicated basis function.

The coefficients of the RBF, ω , are then calculated by solving the linear system

$$R\omega = \mathbf{y}_i$$

where \mathbf{y}_i represents the attribute or design variable to be modeled by this RBF.

The final RBF has the form $y = \omega^\top \Phi(r(\psi))$, where $r(\psi)$ returns a vector of distances from the arbitrary input barycentric coordinate tuple, ψ , to each of the barycentric coordinate tuples of the sampled frontier. (Consequently, these tuples are hard-coded into the RBF function). $\Phi(r)$ is the particular basis function, which in this case is simply the identity function.

Radial basis function networks are interpolating functions, meaning that exactly reproduce the sampled frontier data on which they are based. In order to assess the RBFs generalizability to arbitrary barycentric coordinate tuples, they were created based on a random subset of 80% of the sampled frontier points. The remaining 20% of the sampled frontier points can then be used to assess the quality of the RBFs by comparing the actual values of the attributes/design variables in this data set with the values predicted by the RBFs when they are presented with the corresponding barycentric coordinates. The residual values for y_3 , i.e. $y_{3,predicted} - y_{3,actual}$, are plotted in Figure 70 against $y_{3,predicted}$. The attribute y_3 was chosen for this plot because it generally yielded the worst residuals. Thus the values in Figure 70 may be interpreted as the worst case across all attributes. Points that exceed 4% error are drawn as an \times along the upper or lower limit of the plot.

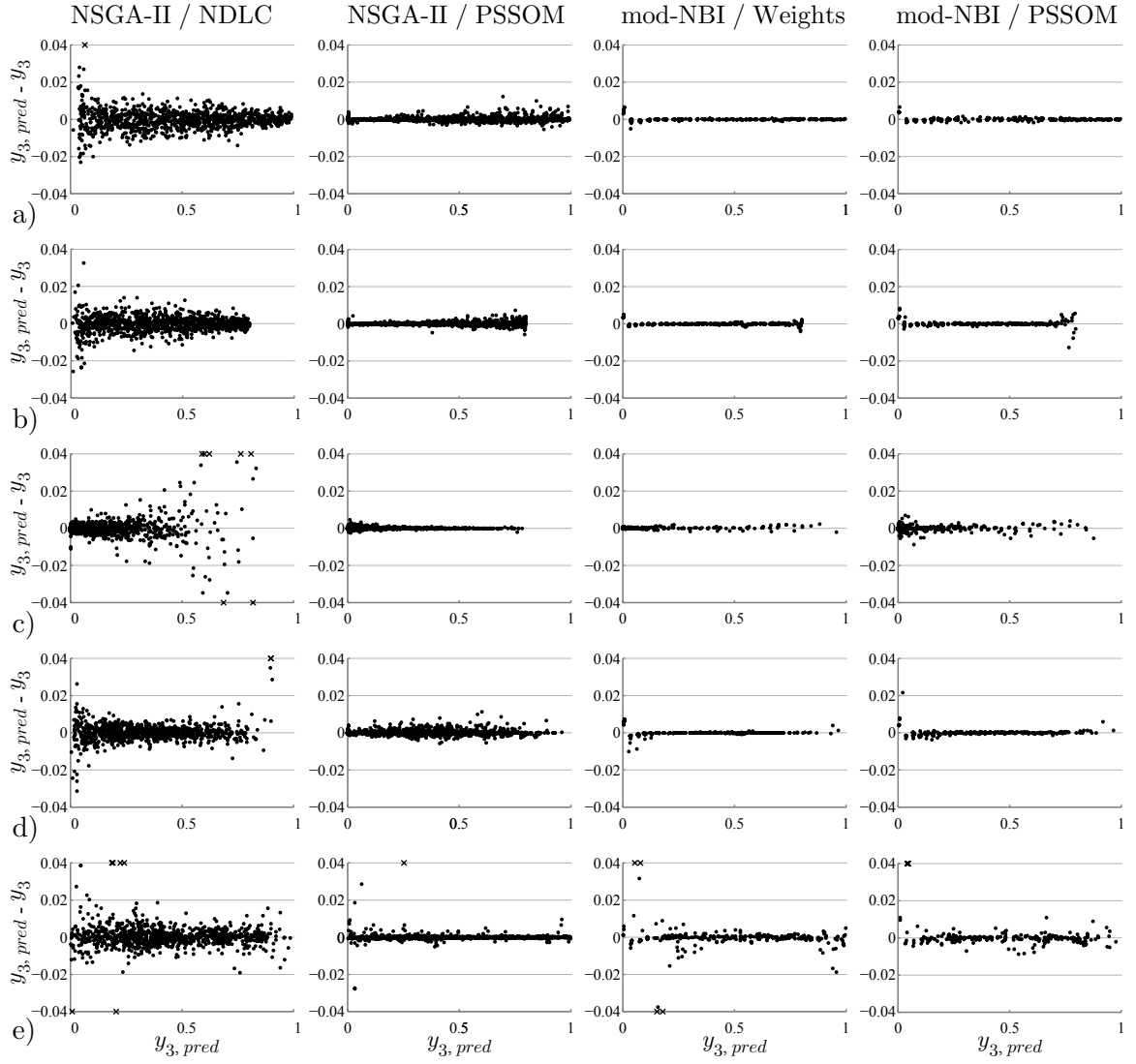


Figure 70: Comparison of radial basis functions' residual vs. predicted values for y_3 . Plots show only the subset of the data not used to train the RBFs. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

Several interesting observations can be made from Figure 70. First, the residuals are generally small. In almost all cases they are less than 2%, and in many cases they are effectively 0. The largest residuals are seen in the NSGA-II/NDLC approach. This is not surprising, as the results presented in Chapter 5 showed that the NDLCs are locally noisy. Yet even for this case the errors are within 1% almost everywhere. The second column shows the same NSGA-II data set parameterized with the PSSOM coordinate system instead of NDLCs. The difference between the first and second columns is thus indicative of the improvements from using PSSOM instead of NDLCs. The fourth column shows the residuals of the mod-NBI/PSSOM approach; thus the difference between the second and fourth column is indicative of the improvements gained from using mod-NBI to sample the frontier instead of NSGA-II. The smallest residuals, however, are seen in the third column, which shows the mod-NBI coordinate system. It is not surprising that these residuals are slightly better than the mod-NBI/PSSOM approach, as using the mod-NBI weights as coordinates avoids possible imprecisions that may arise from training the PSSOM. In either case, however, these residuals are small enough to be inconsequential; they are almost certainly smaller than the error inherent in the original design analysis.

8.2 *Coordinate-Enabled Visualizations of the Pareto Frontier*

This section describes several visualization techniques that leverage the functions of the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ created in the previous section. First, prediction profilers of these functions are examined to give a local view of the Pareto frontier. Then, ternary contour plots and re-sampled scatter plots are examined to give a global view of the frontier.

8.2.1 Prediction Profiler

As described in Section 2.2.5.7, the prediction profiler is a matrix plot that shows how each dependent variable varies as each independent variable is swept through its range of allowable values while the remaining independent variables are held constant. For the present application, the dependent variables are the attributes¹, and the independent variables are

¹Generally the design variables would also be treated as dependent variables in the prediction profiler.

the barycentric coordinates.

As previously described, the barycentric coordinates cannot actually be varied in isolation because the constraint $\sum \psi_i = 1$ must always be satisfied. Consequently, the interpretation of the prediction profiler is slightly modified for this case. Rather than the plots showing variations due to changing one coordinate while the rest are held constant, they show variations due to changing one coordinate while the *ratios of* the other coordinates are held constant.

This rule can be better understood by examining the coordinate grids in Figure 71. The grid on the left shows a Cartesian coordinate system of truly independent variables, while the grid on the right shows a barycentric coordinate system. In each coordinate system, a baseline point is indicated by a white dot. This corresponds to the “current design” about which each plot in the prediction profiler shows some variation. In a traditional prediction profiler, those variations would correspond to changing a single independent variable while the remainder are held constant. This is shown in Figure 71a by the black paths through the current design. Each path is parallel to a corresponding axis, indicating that along these paths only one variable changes at a time. In the barycentric coordinate system, it is impossible to draw a path along which only one coordinate changes at a time. A simple choice of paths to visualize in a prediction profiler thus seems to be the straight-line paths that connect each vertex to the current design, as illustrated in Figure 71b. These paths have the property that all coordinates but one have constant ratios to each other, the excluded coordinate being the one whose corresponding vertex is one endpoint of the path.

Prediction profilers of each example problem are shown in Figures 72–76. Each profiler shows variations around the current design whose coordinates are $\psi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The horizontal axes of these graphs thus correspond to motion through the coordinate space along the paths drawn in Figure 71b. The current design’s coordinate values are indicated by the vertical dashed line in each plot. Each plot shows four solid colored curves. These indicate the variation in the corresponding attribute (rows of the figure) due to motion along the path of the corresponding coordinate (columns of the figure). One colored curve

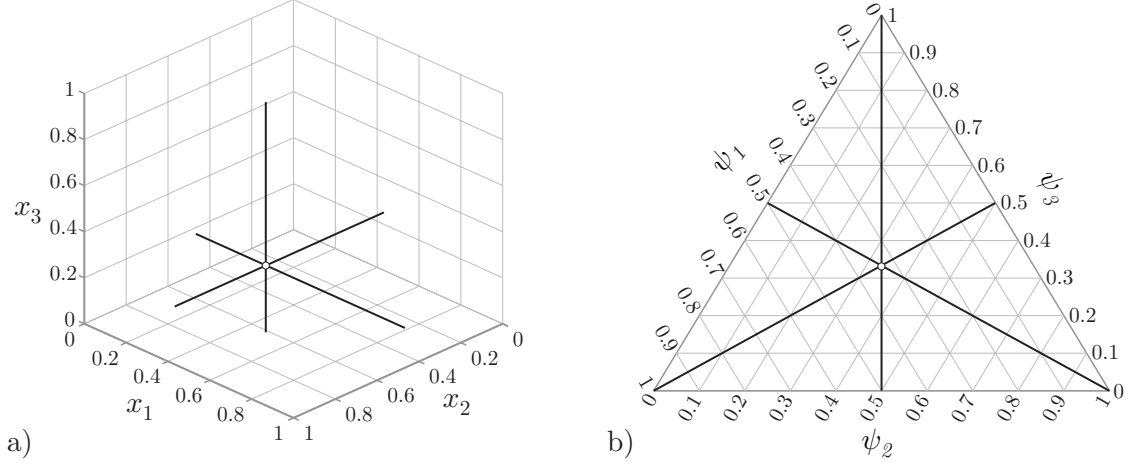


Figure 71: Example prediction profiler paths. a) With independent variables; b) With barycentric coordinates

is shown for each of the four optimizer/coordinate system pairings.² The differences in these curves reflect the differences in each optimizer/coordinate system pairing. The curves were generated by evaluating the radial basis functions constructed in the previous section at many discrete points along the corresponding path through the coordinate space. Where the dashed vertical line indicating the current coordinate crosses the solid colored paths, dashed colored lines are drawn to the corresponding value on the vertical axis. These lines indicate the attribute values corresponding to the coordinate tuple $\psi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

The prediction profilers are best interpreted by examining one column of plots at a time. Consider, for example, the first column of plots in Figure 72. These plots show how each attribute will vary as the ψ_1 coordinate is varies from 0 to 1 while ψ_2 and ψ_3 are held at a constant ratio (in this case, 1 : 1). At the current design point, $\psi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, all coordinates are equal to each other. Roughly speaking, this should correspond to an even balance between all three attributes. As seen by the horizontal dashed lines in Figure 72, this is indeed the case – y_1 , y_2 , and y_3 all have current values that are approximately equal to $\sqrt{1/3}$, which is the center of the frontier. As we move to the right along the colored paths in the first column, the value of y_1 increases, while the values of y_2 and y_3 decrease.

²The prediction profilers in this dissertation show four curves in order to allow the four coordinate systems to be compared side by side. In practice, however, the user will generally be working with a single coordinate system, so each plot in the prediction profiler will contain only one colored curve.

This happens because this motion corresponds to increasing ψ_1 while decreasing ψ_2 and ψ_3 . At the far right edge of the plots, the coordinates are $\boldsymbol{\psi} = (1, 0, 0)$ and correspondingly, $\mathbf{y} = [1, 0, 0]^T$, which is the univariate optimum of y_1 . Conversely, at the far left edge, the coordinates are $\boldsymbol{\psi} = (0, 0.5, 0.5)$ and the attributes are approximately $\mathbf{y} = [0, \sqrt{1/2}, \sqrt{1/2}]^T$, which is the center of the subfrontier $\mathcal{P}(\{y_2, y_3\})$. Because of the symmetry of the spherical Pareto frontier, each column of plots repeats this same behavior, with the ordering of plots within each column altered to reflect the particular path through the barycentric coordinate system that is being viewed.

Prediction profilers are continuous visualizations that show information about local variations around the current design, which may be thought of as looking along k paths through the coordinate space as drawn in Figure 71. Consequently, at any moment the prediction profiler shows only a small portion of the Pareto frontier. In practice, prediction profilers are used interactively in appropriate visualization software to allow the entire frontier to be explored. The vertical dashed lines indicating the current design can be dragged horizontally to set a new current design. As one column's coordinate is changed in this way, the graphs in the other columns change to reflect the new current design. For the present application based on barycentric coordinates, the vertical dashed lines in the other columns also change to reflect the $\sum \psi_i = 1$ constraint. By interactively setting the coordinate values in this way, the user can trace paths along the Pareto frontier with the goal of eventually arriving at a design that offers an acceptable balance of the attribute values.

The process of converging to such a design is helped by the curves displayed in the prediction profiler. Consider again Figure 72. Suppose that the decision maker wishes to move from the current design, whose attribute values are approximately $\mathbf{y} = [\sqrt{1/3}, \sqrt{1/3}, \sqrt{1/3}]$, to a design that more strongly optimizes y_1 . Suppose, for example, that the decision maker wishes to achieve a value of $y_1 \geq 0.8$. The most straightforward way to achieve this is to increase the ψ_1 coordinate, since this coordinate is monotonically increasing with y_1 . From the first column in Figure 72 it can be seen that in order to achieve $y_1 \geq 0.8$, ψ_1 must be greater than approximately 0.5 (from the top left plot). From the remaining plots in this column, it can be determined that $\psi_1 = 0.5$ will result in y_2 and y_3 being at most equal to

about 0.45. If this is acceptable, the decision maker could make this move by dragging the dashed vertical line to $\psi_1 = 0.5$. Then perhaps the decision maker would wish to explore the relative balance between y_2 and y_3 . This could be accomplished by examining the second and third columns of the prediction profiler, to determine, for example, how much of y_2 is acceptable to give up in order to improve y_3 .

We now turn our attention to the particular trends observed in Figures 72–76 and what they reveal about the coordinate systems. For the spherical frontier’s prediction profiler in Figure 72, all the coordinate systems yield very similar curves. Only the NDLCs’ curves are occasionally jagged, indicating the local noise inherent in the NDLCs.

The prediction profiler of the truncated sphere in Figure 73 shows more interesting results. In the first column, as ψ_1 approaches zero, the NSGA-II/PSSOM curves for y_2 and y_3 diverge. The implication is that the coordinate system is shifted in this region such that even though ψ_2 and ψ_3 are equal (both being 0.5 at this point), the corresponding point in the objective space is approximately $\mathbf{y} = [0, 0.65, 0.75]$. Although this behavior would almost certainly be undesirable, an important point to be emphasized is that it is not “wrong” in the sense that the point $\mathbf{y} = [0, 0.65, 0.75]$ is still on the Pareto frontier. This is an example of the subjective quality that may be ascribed to various coordinate systems. The various coordinate systems similarly diverge in the third columns of Figure 73 for $\psi_3 > 0.7$. This is not surprising, as this region corresponds to the irregularity at the top of the frontier. From the bottom right graph, it can be seen that for ψ_3 greater than about 0.7, y_3 is effectively “saturated” at the constrained value $y_3 = 0.8$. Consequently, further increases in ψ_3 correspond to motion along this constraint, which none of the coordinate systems is designed to handle. As such, each coordinate system’s response is effectively random, causing the paths to diverge.

The prediction profiler of the concave problem in Figure 74 is generally well behaved, similarly to the spherical frontier. An interesting result that can be observed is that the NSGA-II sampling does not fully explore the tips of the frontier, but is limited to attribute values less than about 0.8. Consequently, the NSGA-II curves on the diagonal plots are shallower than the mod-NBI curves, and never reach the true univariate optima. This

underscores an important property of Pareto Simplex Exploration: it explores the portion of the Pareto frontier spanned by the discrete sampling used to construct the coordinate system.

The prediction profiler of the teardrop problem in Figure 75 is generally well behaved. Similar effects as those previously described can be seen: slight divergence of the opposing attributes as one coordinate approaches zero, a noisier curve for the NDLCs, and a slight lag in the NDLC curves in approaching the univariate maximum of y_3 .

The prediction profiler of the bumpy sphere problem in Figure 76 again underscores that the goodness of a particular mapping of the coordinates to the frontier is subjective. On account of the bumpy sphere's complicated geometry, each coordinate system shows very different trends along the prediction profiler paths. However, none of these may be identified as being the "correct" trend; each is simply a different path traced along the frontier.

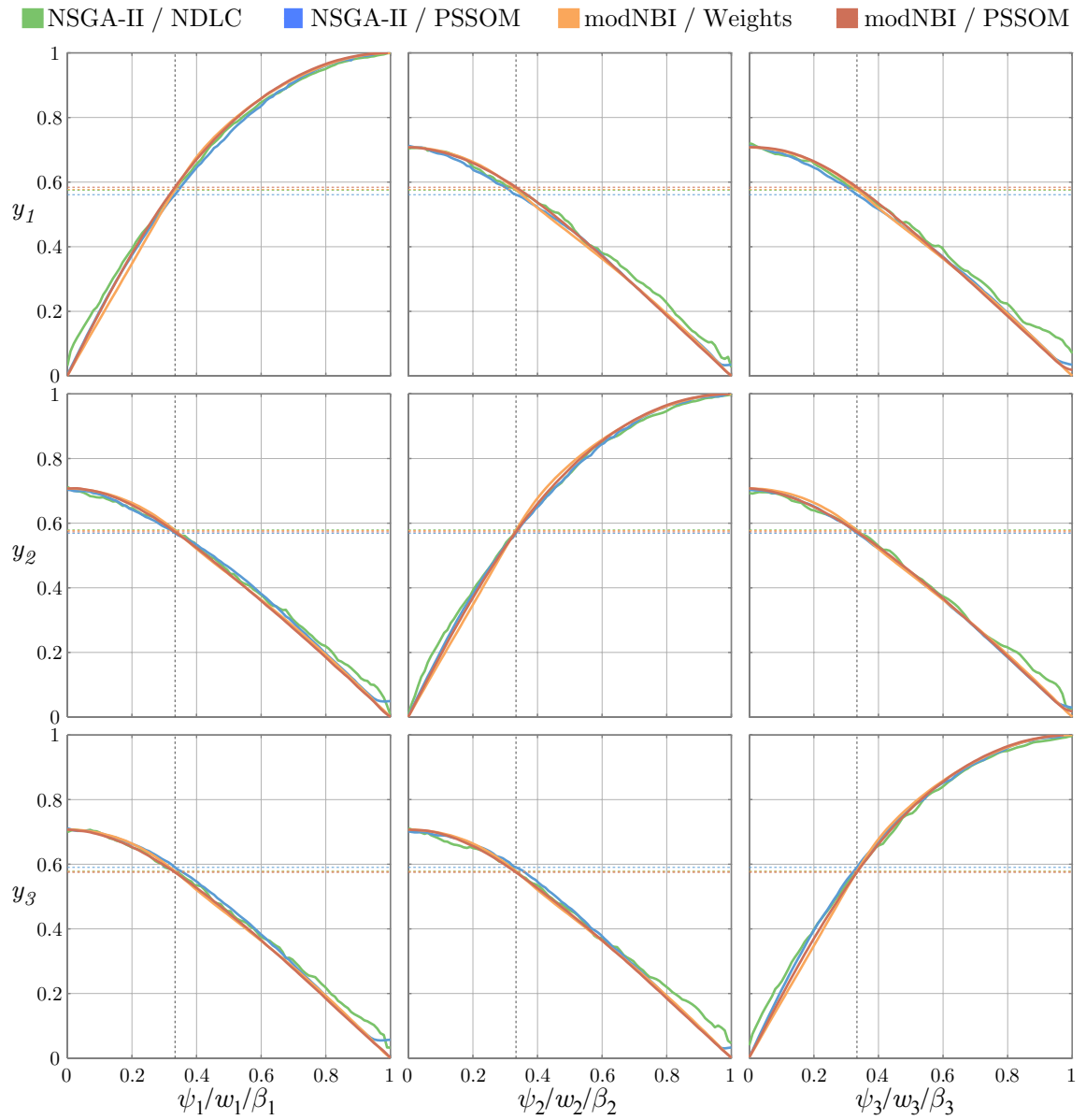
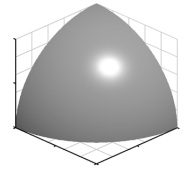


Figure 72: Prediction profiler for sphere example problem

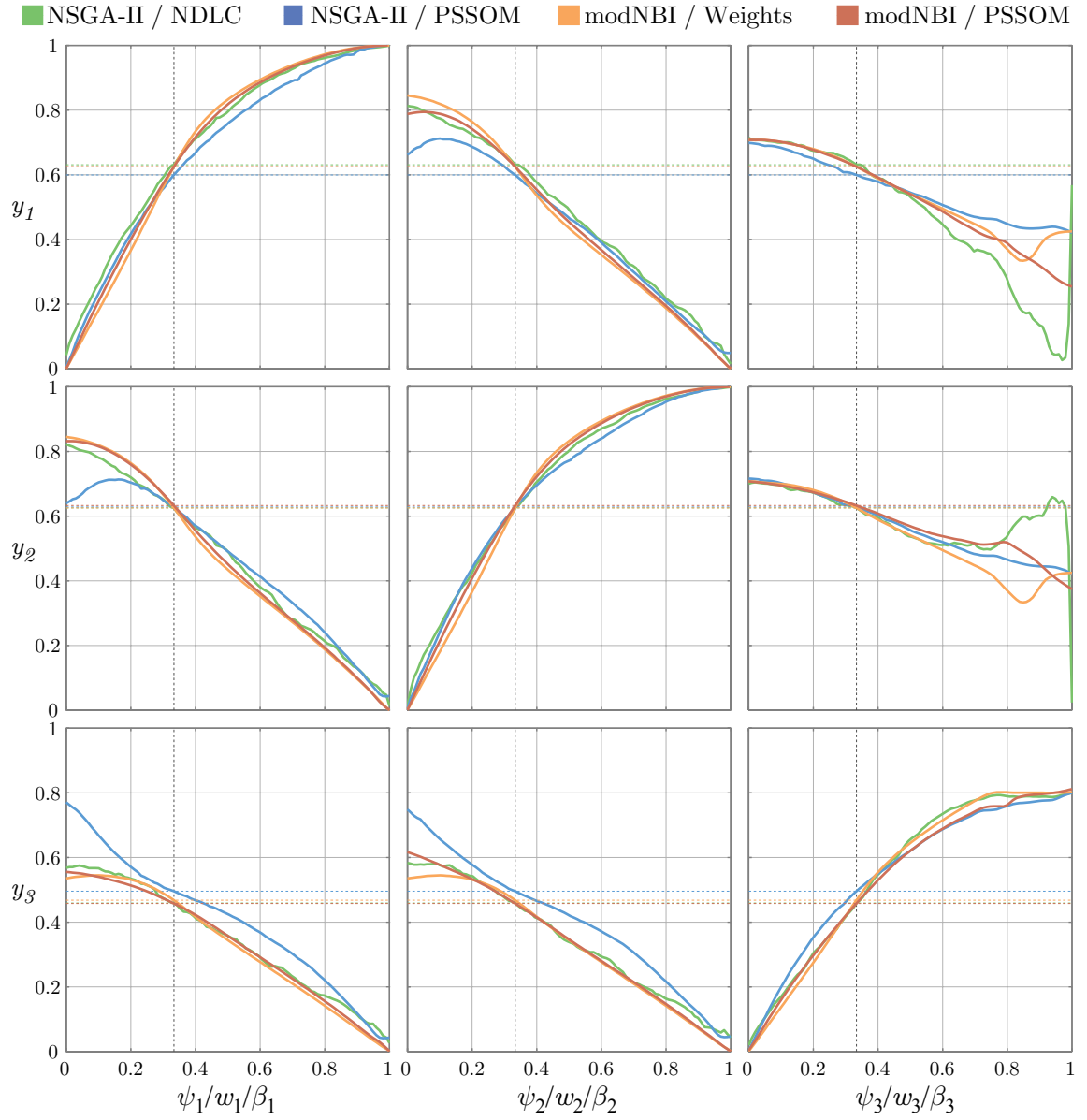
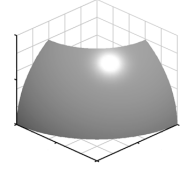


Figure 73: Prediction profiler for truncated sphere example problem

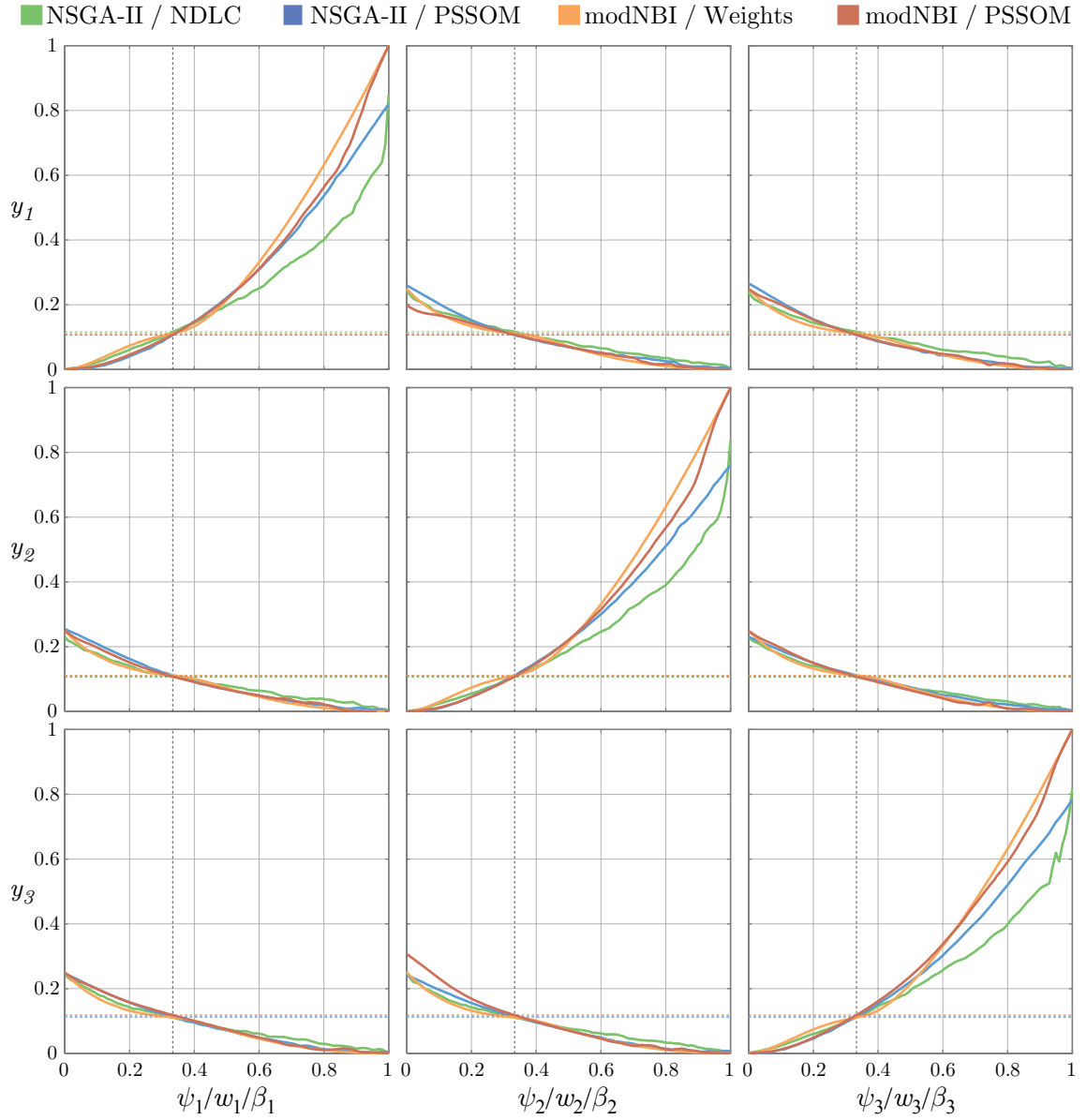
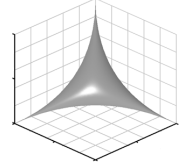


Figure 74: Prediction profiler for concave example problem

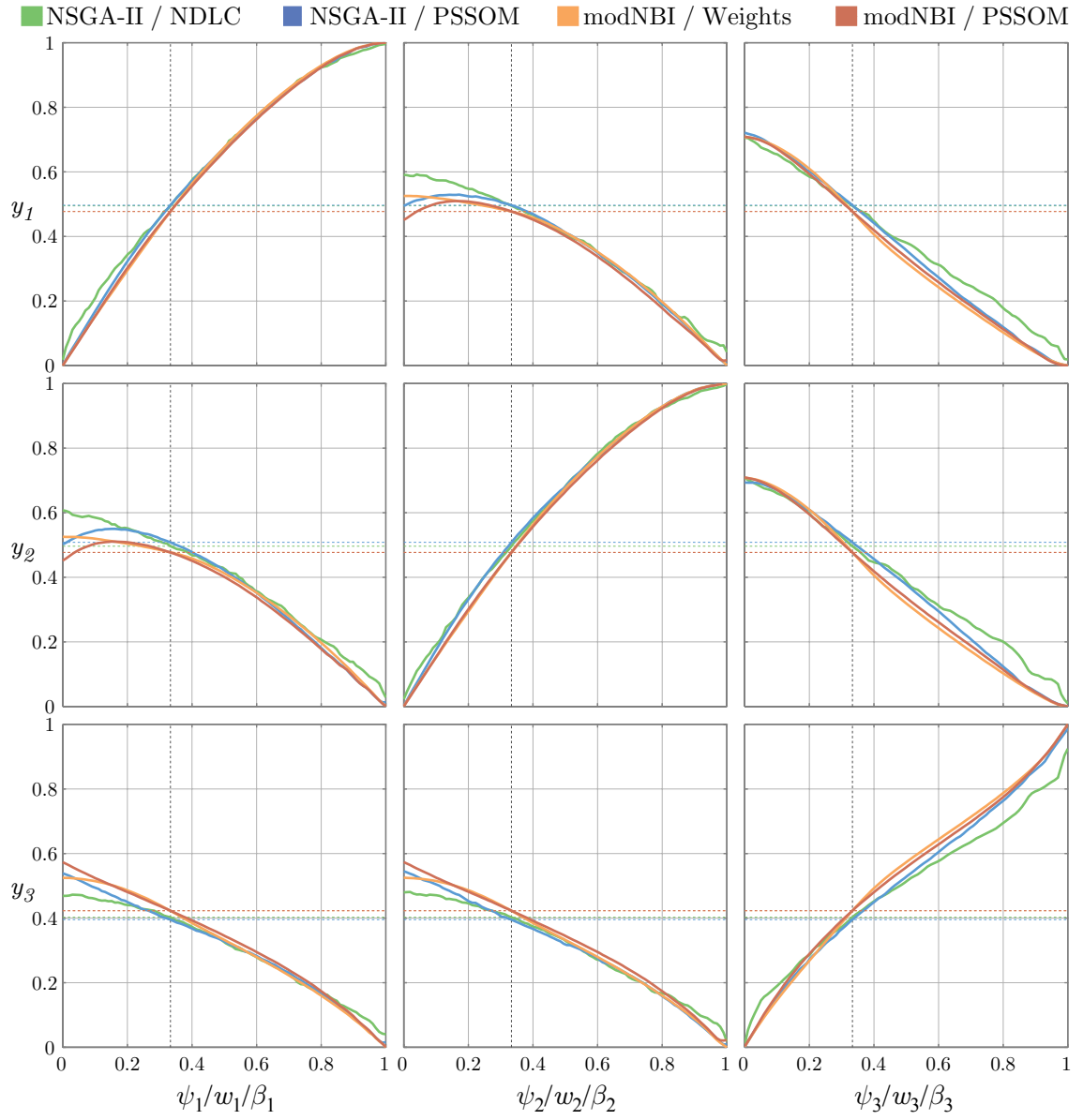
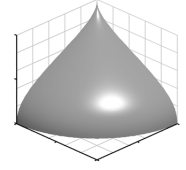


Figure 75: Prediction profiler for teardrop example problem

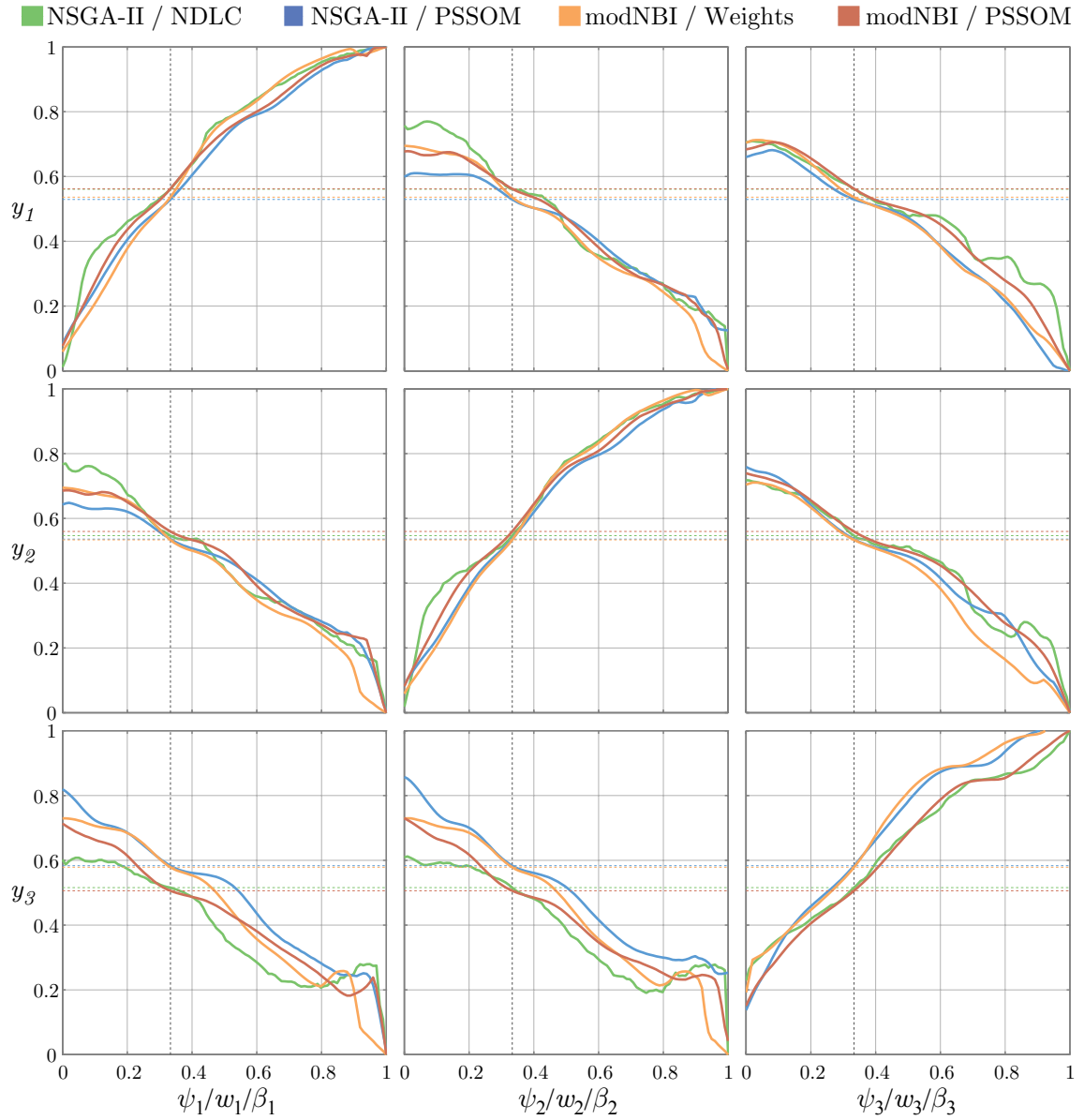
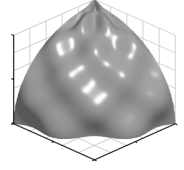


Figure 76: Prediction profiler for bumpy sphere example problem

8.2.2 Ternary Plots

We next examine the Pareto frontiers from a “global” perspective using visualizations that show the entire Pareto frontier in a single view. Two types of plots are considered here. First, a ternary contour plot is used to show the variations of the attribute values plotted directly in the coordinate space. Second, a scatter plot of the attribute values that result from sampling the radial basis functions with an evenly spaced grid of coordinates is shown. A potentially useful, although subjective, measure of the quality of each coordinate system is that this evenly spaced grid of coordinates should yield an evenly spaced sampling of the frontier.

Each of these visualizations is constructed by sampling the radial basis functions using an evenly spaced coordinate grid with 50 points per edge, as shown in Figure 77. The scatter plots directly show the resulting sampling of the frontier, whereas the contour plots interpret the resulting sampling as a continuous manifold by using linear interpolation.

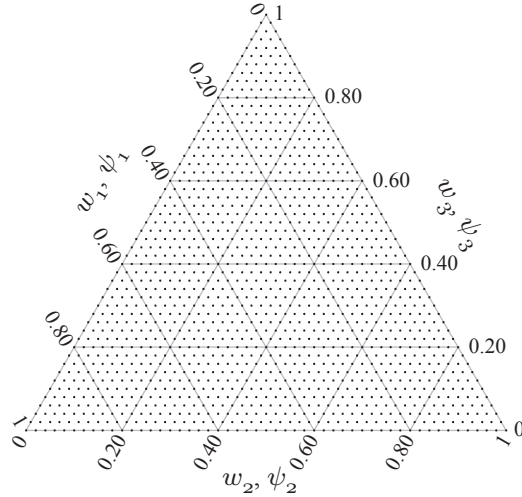


Figure 77: Grid of 1326 weights/coordinates used to assess methods for example problems

Figures 78–82 show the resulting visualizations of each example problem. Each figure shows one row of plots for each optimizer/coordinate system pairing. The first column shows the ternary contour plot of y_2 , the second column shows the ternary contour plot of y_3 , and the third column shows the scatter plot of the sampled points in the objective space. Contour plots of y_1 are not shown since each example problem is symmetric in y_1 .

vs. y_2 . The contour plots are colored such that red = 1 and purple = 0. The black contour lines are drawn at increments of 0.1.

A striking observation from Figures 78–82 is that although we have introduced three very different approaches to constructing coordinate systems on the frontier, the resulting coordinate systems are remarkably similar to each other as indicated by the similarities in the contour plots. For the sphere problem in Figure 78, which admittedly is a highly idealized Pareto frontier, the contour plots are almost indistinguishable. Differences become apparent when viewing the scatter plots, however. Notably, the noise inherent in the NDLCs obscures the grid structure of the sampled coordinates so that the resulting sampling looks fairly random. Each of the other coordinate systems, however, clearly reveals the grid structure of the sampled coordinates. The NSGA-II/PSSOM pairing distributes the sampled points very evenly except for small discrepancies near the univariate optima. The mod-NBI weight coordinates show the same sparseness near the center of the frontier that was observed in Chapter 6. The mod-NBI/PSSOM sampling is, for all practical purposes, perfectly spaced.

The plots of the truncated sphere problem in Figure 79 show interesting views of how each coordinate system deals with the irregularity caused by the $y_3 < 0.8$ constraint. The NDLCs seem to have the least difficulty with the irregularity, as indicated by the fact that the sampling remains entirely within the Pareto frontier and remains fairly evenly spaced. The PSSOM-based approaches clearly reveal the sheetlike structure of the self-organizing map, which results in folds or ripples in the sampling near the irregularity. The mod-NBI-based samplings are seen to have resulting in many weakly non-dominated points that fill in the flat portion on the top of the Pareto frontier. Consequently, the coordinate systems based on these samplings also sample from that portion of the objective space.

The plots of the concave problem in Figure 80 show that none of the coordinate systems is “perfect” in this case. The NDLCs are slightly noisy as always, the NSGA-II/PSSOM approach is slightly irregular near the tips of the frontier. The mod-NBI weight coordinates show their characteristic uneven sampling near the center of the frontier, although for this concave frontier the center is more densely sampled instead of being sparsely sampled. The

mod-NBI/PSSOM sampling also shows slight irregularities near the tips of the frontier. Interestingly, the rippling appears to be less severe for the NSGA-II/PSSOM than for the mod-NBI/PSSOM approach despite the latter's more even initial sampling of the frontier.

The plots of the teardrop problem in Figure 81 show that all the coordinate systems are generally well behaved. As before, slight irregularities in the PSSOM coordinates can be detected near the point at $y_3 = 1$. Aside from these, the PSSOM coordinate systems seem to yield the best results, although the differences between the mod-NBI weight coordinate system and the mod-NBI/PSSOM coordinate system are negligible.

The plots of the bumpy sphere in Figure 82 show how each coordinate system deals with holes in the Pareto frontier. None of the coordinate systems is able to completely avoid sampling from the holes. The mod-NBI weight coordinate system makes no attempt to avoid the holes; as described in Chapter 6 it does not check for global non-domination. The PSSOM coordinate systems avoid the holes to the extent that the local lack of sampled frontier points effectively repels the codebook vectors from the holes, but the partially undone by the redistribution algorithm. The NDLCs do the best job of avoiding the holes but again suffer from their characteristic noisiness.

In order to emphasize that the capability to explore Pareto frontiers in this way is newly enabled by the coordinate systems developed in this work, Figures 83 and 84 show conceptually similar ternary contour and scatter plots based on using the weights in a weighted sum aggregate objective function and a Chebychev norm aggregate objective function as coordinates. The weighted sum is known to only be able to sample from the convex hull of the Pareto frontier, while the Chebychev norm has infinite curvature that allows it to sample from the entire frontier regardless of convexity. In both cases, the results are unsatisfactory. The weighted sum AOF, as shown in Figure 83, fails completely on the Pareto frontiers with concave portions. In the extreme case of the concave example problem, it is able to sample only the individual optima. Even for the convex sphere example problem, however, the spacing of the sampled points is irregularly clustered near the individual optima. For the truncated sphere, the weighted sum AOF is essentially unable to detect and adapt to the presence of the $y_3 < 0.8$ constraint. Instead, the sampling below this constraint is identical

to the spherical frontier, and the remaining points pile up along the constraint.

The Chebychev norm AOF seems appealing because it is able to sample from any portions of the Pareto frontier regardless of the local convexity. However, Figure 84 reveals that this alone does not lead to a suitable coordinate system on the frontier. The mapping of weights to sampled frontier points is extremely uneven. Almost all the sampled points cluster along the 2-attribute subfrontiers, leaving the interior of the frontier sparsely and irregularly sampled.

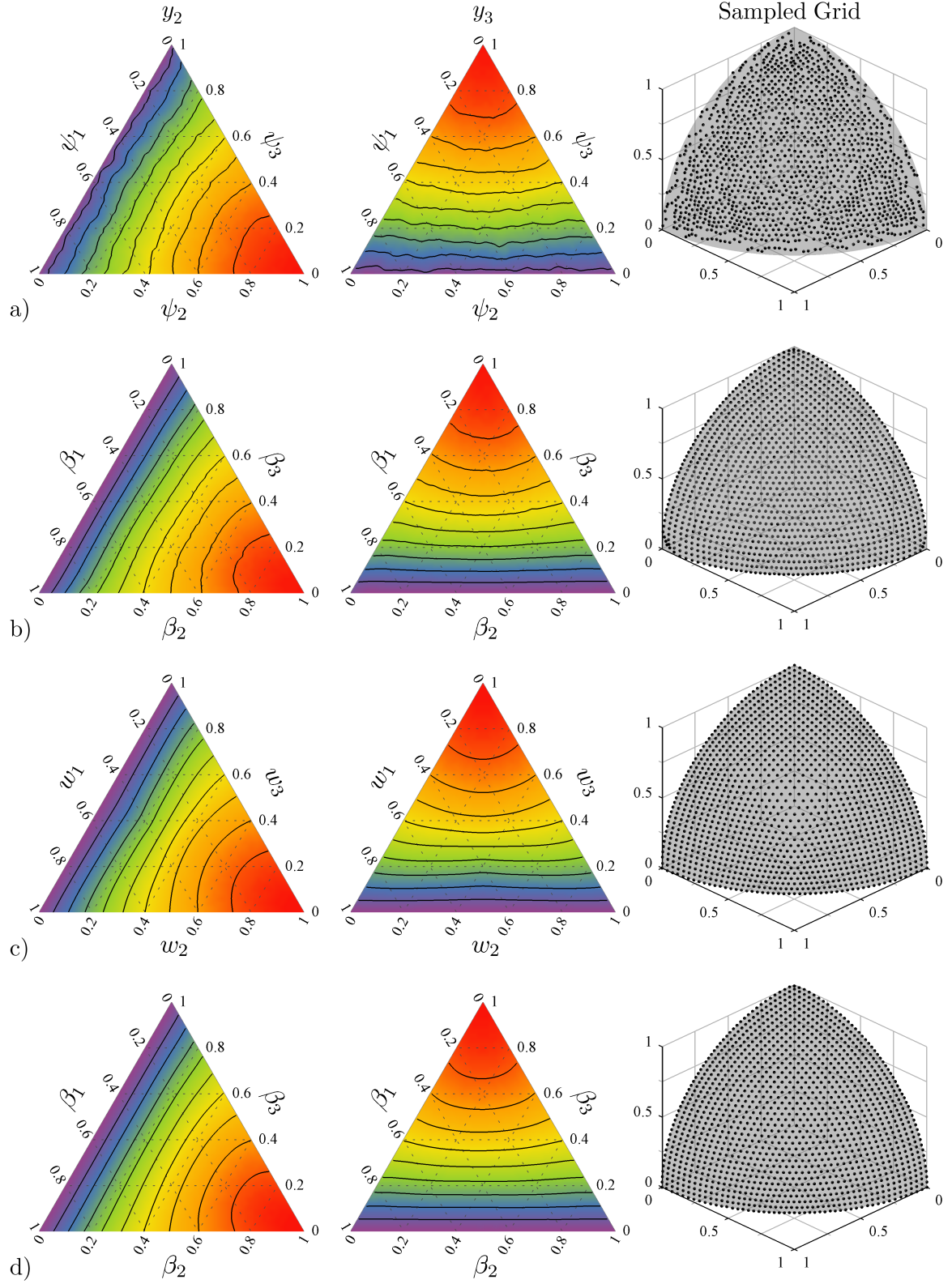


Figure 78: Comparison of coordinate systems for sphere example problem. a) NSGA-II / NDLC, b) NSGA-II / PSSOM, c) mod-NBI / Weights, d) mod-NBI / SOM

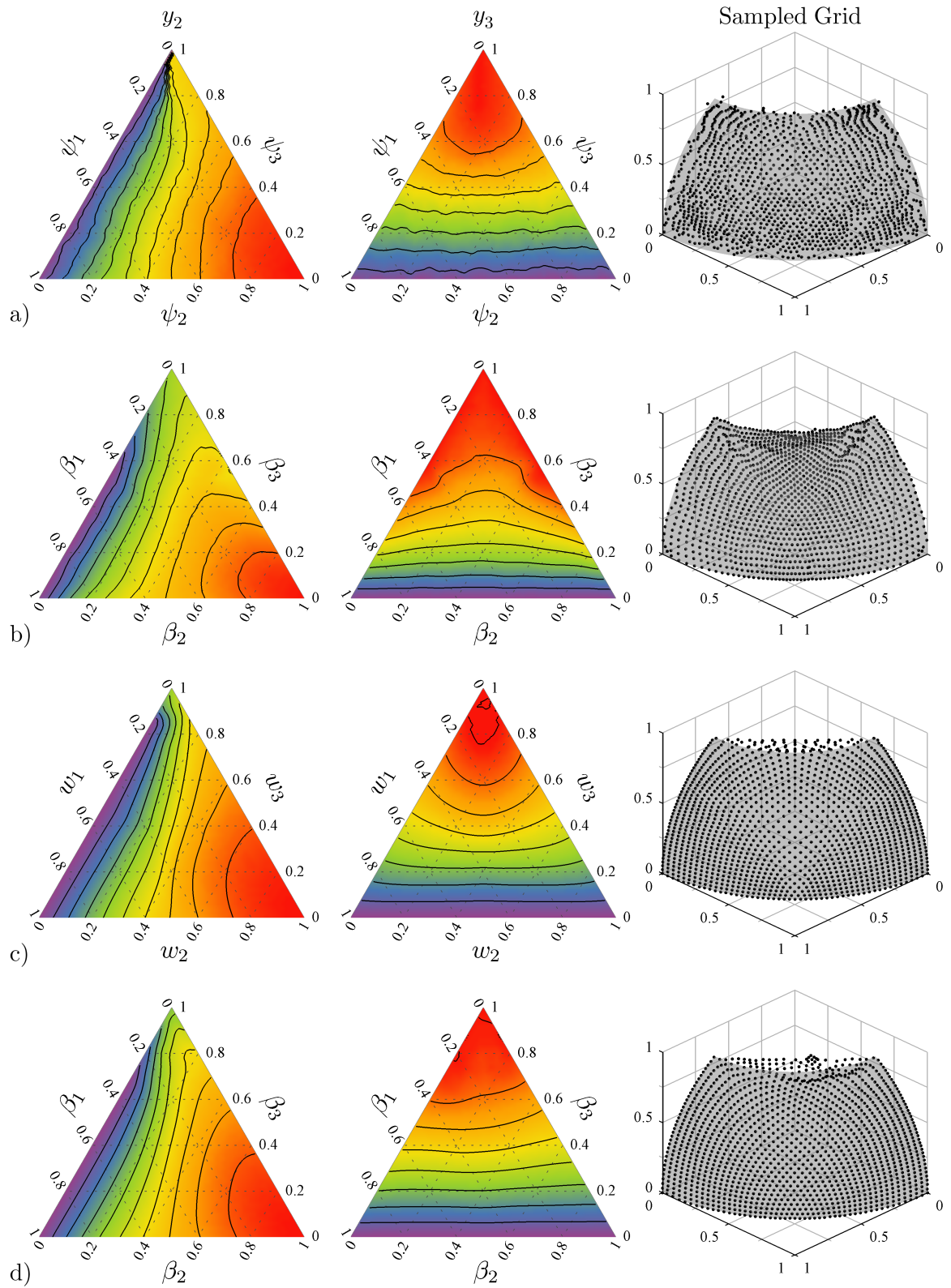


Figure 79: Comparison of coordinate systems for truncated sphere example problem. a) NSGA-II / NDLC, b) NSGA-II / PSSOM, c) mod-NBI / Weights, d) mod-NBI / SOM

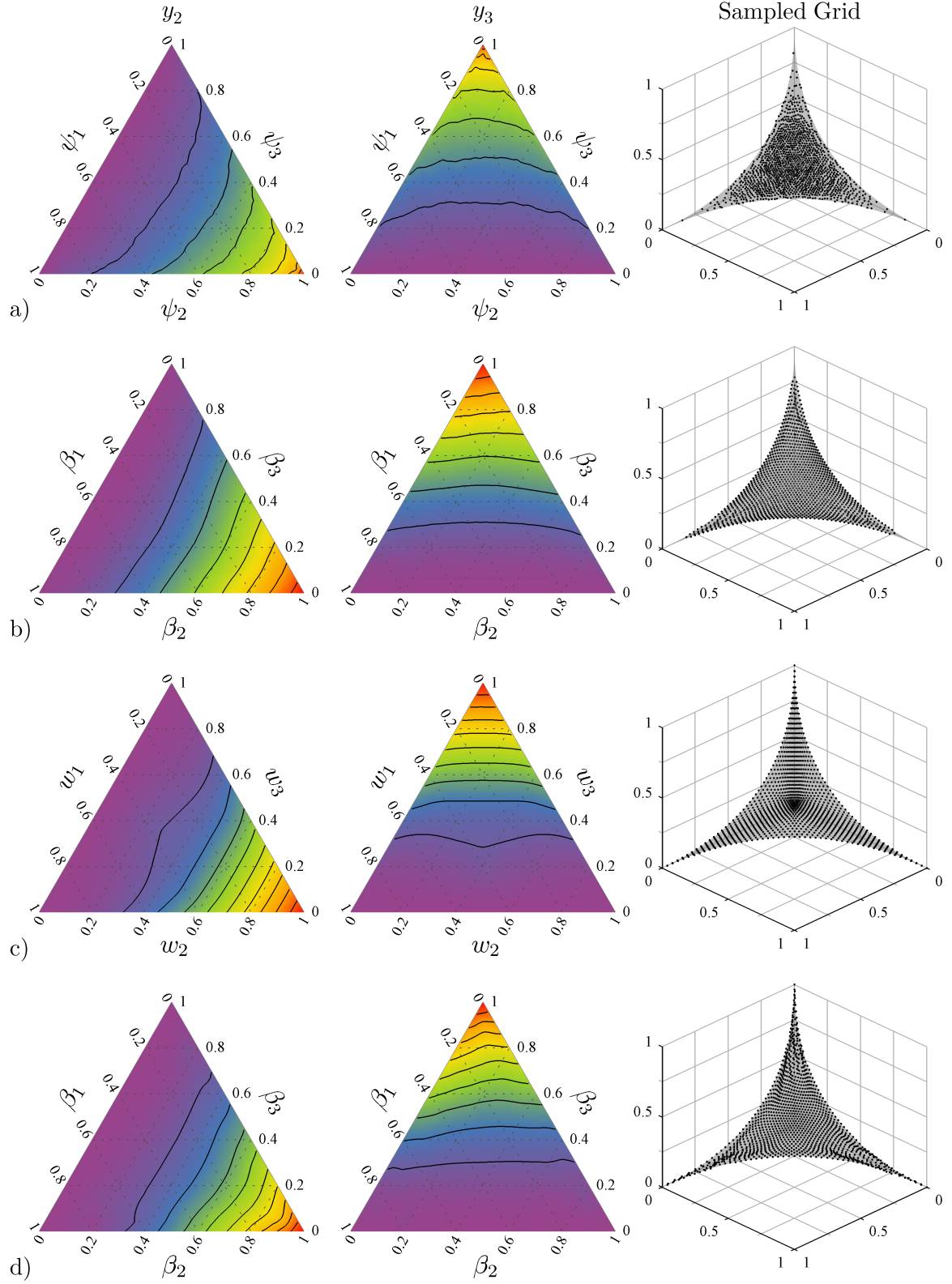


Figure 80: Comparison of coordinate systems for concave example problem. a) NSGA-II / NDLC, b) NSGA-II / PSSOM, c) mod-NBI / Weights, d) mod-NBI / SOM

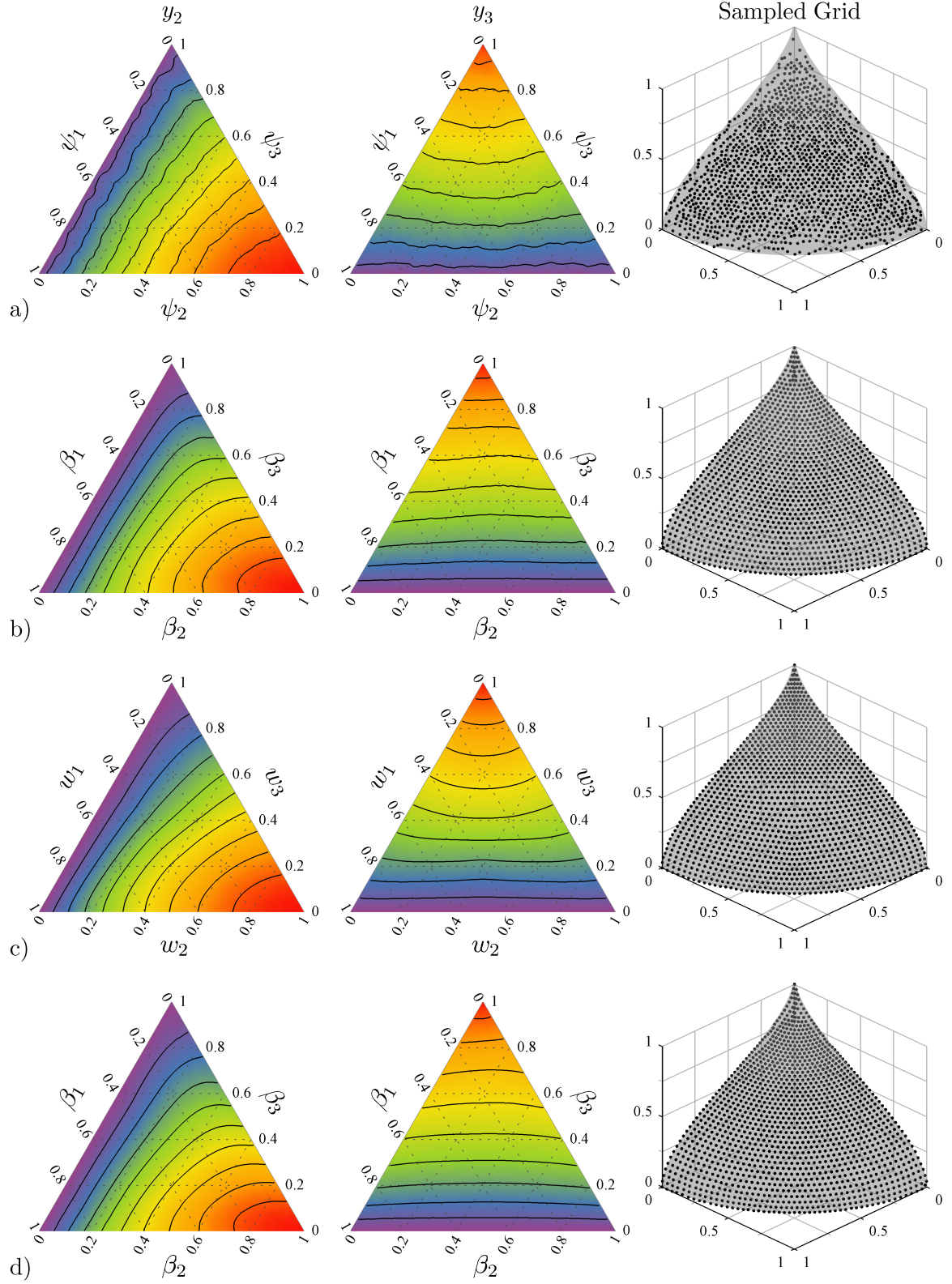


Figure 81: Comparison of coordinate systems for teardrop example problem. a) NSGA-II / NDLC, b) NSGA-II / PSSOM, c) mod-NBI / Weights, d) mod-NBI / SOM

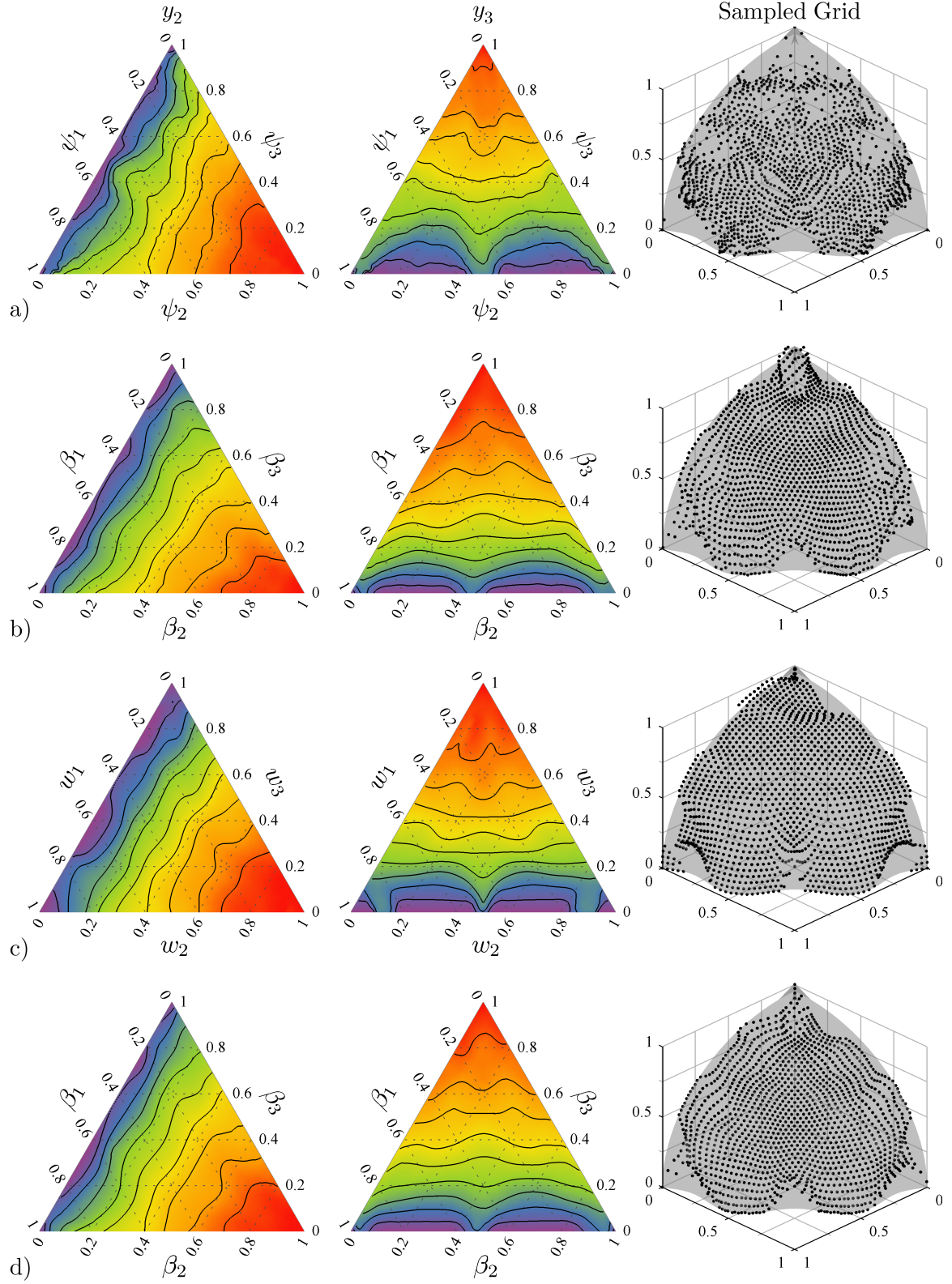


Figure 82: Comparison of coordinate systems for bumpy sphere example problem. a) NSGA-II / NDLC, b) NSGA-II / PSSOM, c) mod-NBI / Weights, d) mod-NBI / SOM

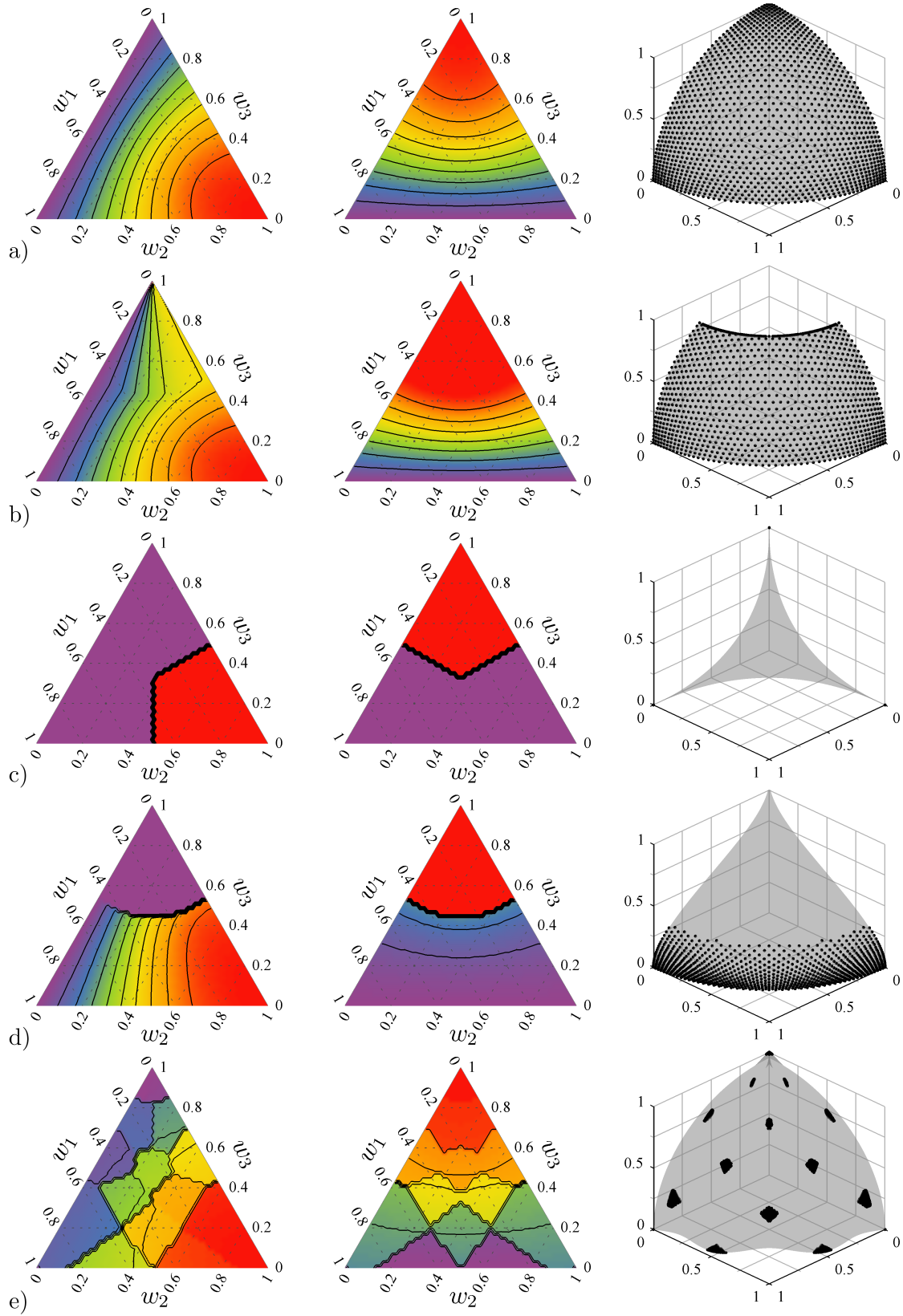


Figure 83: Effect of a weighted-sum coordinate system on example problems. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

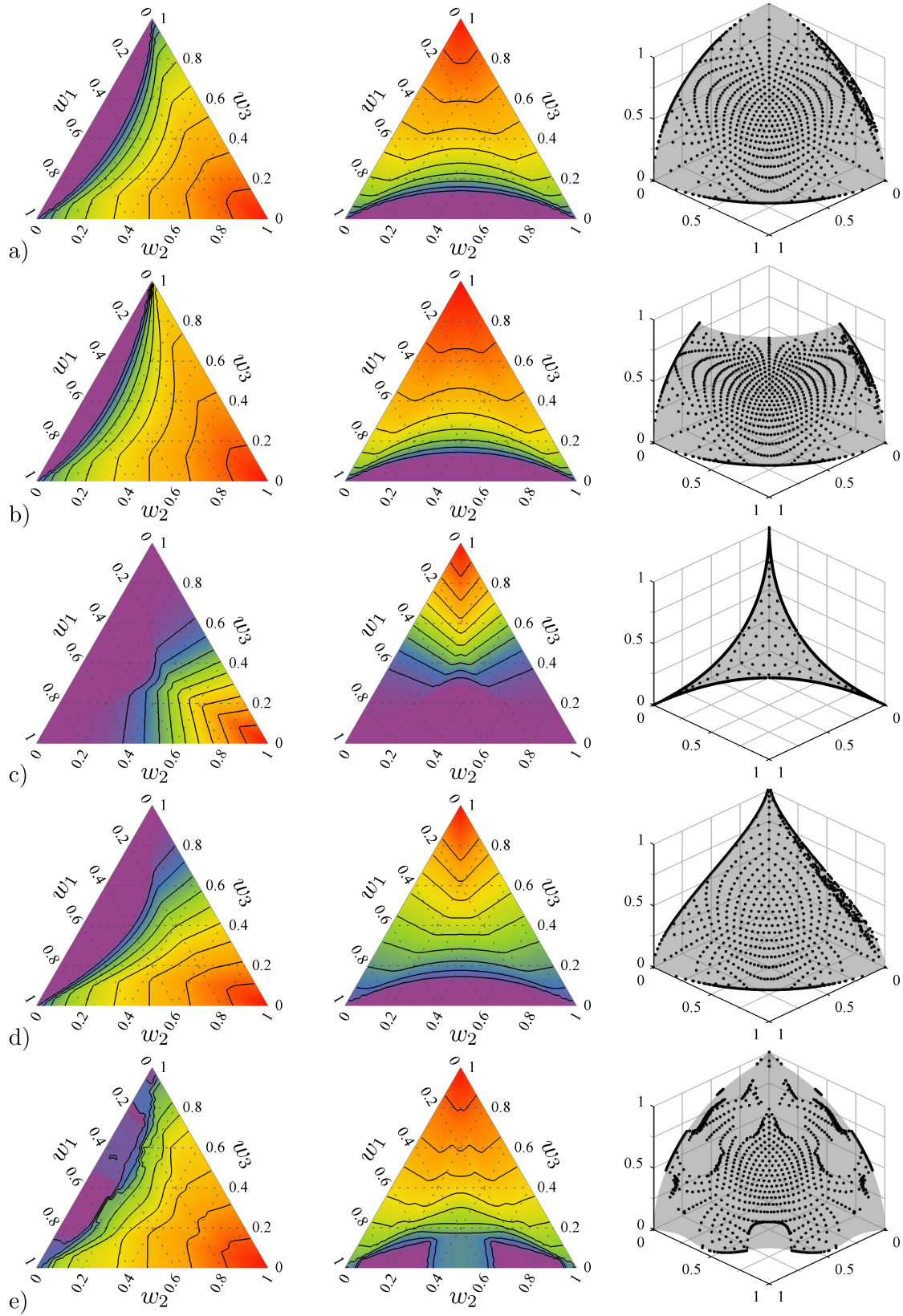


Figure 84: Effect of a Chebyshev norm coordinate system on example problems. a) Sphere, b) Truncated sphere, c) Concave, d) Teardrop e) Bumpy sphere

8.3 *Wing Design Problem Case Study*

The example problems considered thus far have been useful for demonstrating the behaviors of the coordinate systems on various possible Pareto frontier geometries in the objective space. However, they lack interesting design analysis functions that reflect the complexity of real engineering design problems. In order to enable demonstrations of the coordinate systems' suitability for exploring not only the Pareto frontier in the objective space, but its preimage in the design space, a more typical engineering design problem based on the conceptual design of a transonic aircraft wing is introduced in this section.

The particular problem formulation considered here was developed by German and Takahashi [52]. The problem has five independent variables, listed in Table 8, and three attributes:

- To maximize the product of the cruise Mach number and the lift-to-drag ratio (ML/D), a measure of aerodynamic efficiency in cruise
- To minimize the wing weight
- To maximize the wing fuel capacity.

Aircraft design engineers often inspect the best attainable values of these metrics as part of the design space exploration activity. From a decision making perspective, these attributes are not typically used directly to select a design; however, the metrics are known to be monotonic with higher-level attributes such as aircraft range performance. As shown by Malak and Paredis [88], this monotonicity allows Pareto frontiers resulting from a subsystem-level multi-objective optimization to be used to reduce the search space for a system-level single-objective optimization.

The aerodynamic design analysis for the wing is based on a two-parameter quadratic drag polar. The induced drag is determined by the wing aspect ratio and the span efficiency, which is estimated by a correlation with root bending moment relief (RBMR), a measure of the deviation of the lift distribution from ellipticity [40, 125]. The zero-lift drag component is calculated by a semi-empirical buildup based on skin friction coefficient, form factor due

Table 8: Independent Variables for Wing Design Problem

Variable	Symbol	Lower Limit	Upper Limit
Aspect Ratio	\mathcal{R}	6	12
Taper Ratio	TR	0.1	1
Leading Edge Sweep	Λ_{LE}	15°	40°
Planform Area	S	800ft^2	2000ft^2
Root Bending Moment Relief	RBMR	0%	16%

to thickness, and typical values for non-wing wetted area for transport aircraft in the 150 klb size class. Wing thickness is allocated with the Korn equation based on the specified leading edge sweep angle and a given critical Mach number. A tip stall constraint is included, which limits the maximum section lift coefficient at 80% semispan to 0.6. Wing weight is estimated from a semi-empirical regression that includes one term that scales with wing area and a second term that scales with root bending moment. The effects of aspect ratio, taper ratio, and t/c are also included in the wing weight calculation. Finally, fuel capacity is determined from the volume of the equivalent prismatic wing and a typical fuel volume fraction. The complete design analysis functions are detailed in Appendix E.

8.3.1 Sampled Frontiers and Coordinate Systems

As for the previous example problems, two samplings of the wing design problem were created: a 5000 point NSGA-II population and a 50-level mod-NBI sampling. Figure 85³ shows the samplings plotted in the objective space. The sampled Pareto frontier is seen to have a complicated shape that is not particularly simplex like. The two samplings differ slightly in their exploration of the bounds of the frontier. The mod-NBI sampling more crisply defines the edges, especially along the top and bottom of the frontier as it is oriented in Figure 85. The NSGA-II sampling, however, more thoroughly samples the “tail” of high ML/D and high wing weight points. Many of the mod-NBI sampled points in this region are infeasible or dominated.

As for the above example problems, four coordinate systems were constructed: NDLCs based on the NSGA-II sampling, the mod-NBI weight coordinates, and PSSOM coordinates

³All figures for this section are located at the end of the chapter.

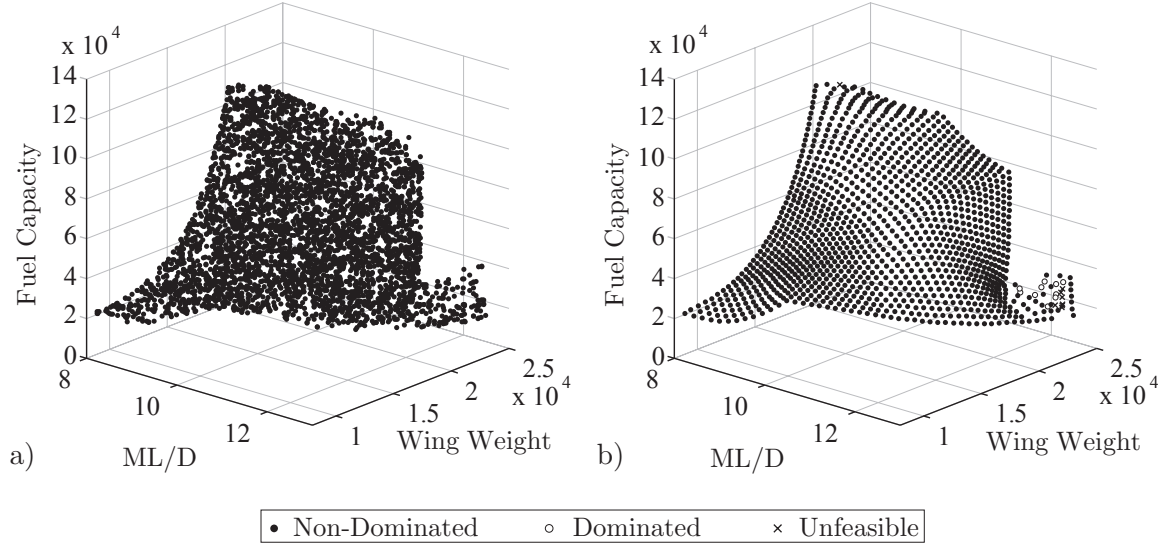


Figure 85: Sampled frontiers for wing design problem, plotted in the objective space. a) 5000 point NSGA-II b) 1326 point mod-NBI

for both the NSGA-II and mod-NBI samplings. The infeasible and dominated points were removed from the mod-NBI sampling before calculating the PSSOM coordinates. The coordinate values for each method are plotted in Figure 86. The results are very encouraging. Despite the complicated Pareto frontier geometry, each method is able to produce a coherent coordinate system that adheres to the guidelines suggested in Chapter 4. Unlike the previous example problems, for which the four coordinate systems produced very similar results, Figure 86 shows a stronger difference in the NDLCs compared to the other mappings. The two mod-NBI coordinate systems are similar to each other, with the mod-NBI / PSSOM coordinate system perhaps removing some spurious waves in the ML/D and Wing Weight coordinates.

For comparison, Figure 87 shows the sampled frontiers colored according to the actual attribute values at each point, rather than the coordinate values as in Figure 86. Comparing these two Figures reveals the extent of the non-linearity of the coordinate mappings to their corresponding attributes. This is most pronounced in the wing weight coordinates' contours, which are sharply curved to conform to the Pareto frontier.

In order to demonstrate the quality of the PSSOM training, Figure 88 shows the final trained self-organizing map topologies superimposed on the sampled frontiers. For both

the NSGA-II and the mod-NBI sampling, the self-organizing map is able to conform to the frontier quite well despite the unusual frontier geometry. The only difficulty is seen in the tail of the mod-NBI frontier, which is very sparsely sampled.

8.3.2 Radial Basis Functions and Continuous Visualization with Prediction Profilers

Radial basis functions of the form $y_i = g(\psi)$ and $x_i = g(\psi)$ were created to calculate each attribute and design variable as functions of each coordinate system. As with the previous example problems, 20% of the sampled frontier points were withheld to assess the predictive power of the radial basis functions. Figure 89 shows the error of the radial basis functions predicted values vs. the actual values of these withheld sampled points. All values have been normalized so that the predicted values for each variable range from 0–1. The vertical axes of the plots showing the attributes range from -5% to 5% error, while the vertical axes for the plots showing the design variables range from -20% to 20% error.

As before, the residuals for the mod-NBI methods are small enough almost everywhere to be negligible. The prediction of the design variable values for the mod-NBI sampling are also very good, with a few high-error outliers. By comparison, the NSGA-II samplings yield much higher residuals. This is due in large part to the fact that the NSGA-II does not use derivative information to precisely converge its sampled points to the true Pareto frontier, resulting in relatively large local fluctuations in the design variable values. This effect was illustrated in Figure 23 on page 64. Consequently, the residuals of the RBFs that calculate the design variables' values are much worse than those of the RBFs that calculate the attributes' values. The diagonal line patterns seen in many of the plots occur when the variables' actual value is either 0 or 1, such that all the residuals fall along a line with slope equal to 1. Comparing the NSGA-II/NDLC residuals with the NSGA-II/PSSOM residuals shows that the PSSOM coordinates are much better able to predict the attributes' values, but only marginally better at predicting the design variables' values. This again seems to indicate that the error in the design variables' values is inherent in the NSGA-II sampling and not an artifact of the coordinate system.

A second method of validating the radial basis functions is to evaluate the RBFs over

some grid of coordinates in order to predict the values of both the design variables and the attributes, and to then compare the RBF-predicted values of the attributes, to the attribute values calculated by the original design analysis evaluated over the RBF-predicted values of the design variables. For this validation study, a grid of weights with 41 levels was used. This value was chosen to be different than the grid size used to sample the frontier (50 levels) to give an representation of the quality of the coordinate for interpolating between the original sampled points. Each of the eight RBFs was evaluated over this grid to yield the “predicted” values of the three attributes, $y = g_y(\psi)$, and the five design variables, $x = g_x(\psi)$. The original design analysis f , described in Appendix E, was then used to calculate the “actual” values of the attributes as functions of the predicted values of the design variables, $\hat{y} = f(g_x(\psi))$.

Figure 90 shows histograms of the percentage error of the attributes, calculated as $(\hat{y} - y)/\hat{y} \times 100\%$. The histograms show that the errors between the original design analysis and the RBFs are negligible, being less than 0.25% almost everywhere for all the coordinate systems. The implication of this finding is that the user can have high confidence that the attribute values predicted by the RBFs are accurate.

8.3.3 Continuous Visualization with Prediction Profilers

A prediction profiler comparing the four coordinate systems is shown in Figure 91. The top three rows of plots show the three attributes as functions of the coordinates, and the bottom five rows show the five design variables as functions of the coordinates. The curves in the top three rows of plots show that the PSSOM and mod-NBI weight coordinate systems perform similarly to each other, with the NSGA-II / NDLC coordinate system differing somewhat. The NDLCs are most different in the second column, which shows variations due to changes in the $\psi_{\text{Wing Weight}}$ coordinate. This effect could also be seen in the scatter plots in Figure 86a, which shows that the $\psi_{\text{Wing Weight}}$ coordinate contours are compressed near the minimum value of Wing Weight, and widely spaced near the subfrontier of ML/D and Fuel Capacity.

As in the previous examples, the NDLCs also show the noisiest curves in Figure 91,

although the effect is small enough in the $y_i = g(\psi)$ plots that it would likely not be problematic. The noisiness is more pronounced in the $x_i = g(\psi)$ plots, although these plots also show significant noisiness for the NSGA-II / PSSOM coordinate system. This again seems to confirm that primary source of the noise is the poor convergence of the NSGA-II optimizer rather than being a property of the NDLCs themselves.

An interesting effect can be seen in the plot of TR vs. $\psi_{\text{ML/D}}$, in which the coordinates based on the mod-NBI sampling show large spikes at high values of $\psi_{\text{ML/D}}$, while the coordinates based on the NSGA-II sampling predict $\text{TR} = 0$ across the entire range of $\psi_{\text{ML/D}}$. The peaks predicted by the mod-NBI sampling appear to be erroneous. They are based on a few outlying sampled points. In any case, by comparing the effects of all four coordinate systems in the $y_i = g(\psi)$ plots, these TR spikes do not appear to have much affect on the attribute values.

The multi-curve prediction profilers considered thus far have been used to compare the effects of the four coordinate systems. Presumably in real applications, designers will only be concerned with a single coordinate system, so the prediction profiler will show only a single curve per plot. An example of this is shown in Figure 92 for the mod-NBI / PSSOM radial basis functions. Additionally, each plot in Figure 92 has been underlayed with the corresponding scatter plot that shows a projection of the mod-NBI / PSSOM sampled frontier. A variant of this technique has been previously demonstrated by Baukol et al. [12] in which only points near the current design are shown. In Figure 92, however, the entire sampled frontier is shown. Compared to Figure 91, the current design in Figure 92 has also been changed slightly to correspond to the coordinates $(\frac{1}{5}, \frac{1}{5}, \frac{3}{5})$.

The interpretation of the “augmented prediction profiler” in Figure 92 is that the red curves show the local variations around the “current design” caused by changing one coordinate while the other two remain at a fixed ratio to each other, while the regions spanned by the gray scatter plot projections show the possible locations of the red curves over *all possible* coordinate values. Each sampled frontier point is plotted as a gray circle in every plot; therefore, regions of the plots with no gray circles are not part of the (sampled) Pareto frontier.

Consider for example, the plot of Fuel Capacity vs. $\psi_{\text{Fuel Capacity}}$. The red curves show that fuel capacity increases nearly perfectly linearly with its corresponding coordinate, $\psi_{\text{Fuel Capacity}}$. The region spanned by the gray points reflect this same strong linear trend, suggesting that regardless of the relative values for $\psi_{\text{ML/D}}$ and $\psi_{\text{Wing Weight}}$, the Fuel Capacity vs. $\psi_{\text{Fuel Capacity}}$ will remain strongly linear. However, the gray points also reveal that Fuel capacity is at a local maximum with respect to the balance of $\psi_{\text{ML/D}}$ and $\psi_{\text{Wing Weight}}$. This is indicated by the fact that the current design point sits atop the band of gray points. Changing the relative balance of $\psi_{\text{ML/D}}$ and $\psi_{\text{Wing Weight}}$ while keeping $\psi_{\text{Fuel Capacity}}$ at its current value (0.333) will therefore cause fuel capacity to decrease slightly.

The gray points show a common pattern in every plot in the top three rows of Figure 92, which is reflective of the nature of Pareto frontiers. Toward the left of each plot, the band of gray points is thickest, and it tapers to a point toward the right of each plot. This occurs simply because the right side of each plot corresponds to a single point – the individual optimum of the attribute corresponding to that column’s coordinate – while the left side of each plot corresponds to the opposing two-attribute subfrontier. Consequently, points near the right side of the plots show less variability as the frontier begins to taper toward the individual optima. The bottom five rows of plots, which show the corresponding variability in the design variables, also taper toward the right side of the plots, but do not necessarily widen at the left. For example, the plot of \mathcal{R} vs. $\psi_{\text{ML/D}}$ shows that for $\psi_{\text{ML/D}} = 0$, i.e. for all points on the subfrontier of wing weight and fuel capacity, $\mathcal{R} = 0.1$.

The interactive nature of the prediction profiler can be inferred somewhat by comparing the red curves in Figure 91 to those in Figure 92. The difference between the two figures is that the “current design” has been changed from $\psi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ in Figure 91 to $\psi = (\frac{1}{5}, \frac{1}{5}, \frac{3}{5})$ in Figure 92. When used interactively on a computer, this would correspond to the user’s dragging the vertical dashed line in the third column of plots to the right to increase $\psi_{\text{Fuel Capacity}}$, which would in turn automatically lower $\psi_{\text{ML/D}}$ and $\psi_{\text{Wing Weight}}$ to keep the sum of the coordinates equal to 1.

The red curves in the third column of plots are identical in Figures 91 and 92. By design, the curves within the column in which the user drags the vertical dashed line do not

change as the line is dragged, in other words, each curve is drawn to reflect the fact that as this coordinate increases, the others must decrease correspondingly. Necessarily then, the curves in the remaining two columns must change. This can be seen by comparing the red curves in the first two columns of Figure 91 with those in Figure 92. In this particular case, the differences are slight. Most of the curves have been shifted up or down slightly to reflect the new current design's values for each variable, but the shapes of the curves remain similar in most of the plots. Among the more significant changes are shapes of the curves showing \mathcal{R} vs. $\psi_{ML/D}$ and RBMR vs. $\psi_{Wing\ Weight}$.

As an example of how the coordinates can be used to simultaneously understand tradeoffs in the objective space and their corresponding causes in the design space, suppose that a designer believes that, compared to the current design in Figure 92, a design with more fuel capacity may be preferred. In order to explore such designs, it is best to consider the curves in the third column, since these correspond to changes in $\psi_{Fuel\ Capacity}$ and thus show the “most direct” way of impacting the fuel capacity. From the top three plots in the third column, it can be seen that the fuel capacity can be increased from its current value of 9.6×10^4 to 11.7×10^4 by increasing $\psi_{Fuel\ Capacity}$ to 0.8. The tradeoff for making this change is that ML/D will be reduced from its current value of 10.8 to 10.5, but the wing weight will actually improve slightly, from about 15,750 to 15,100.⁴

Supposing that the designer finds these trades acceptable, he or she may then wish to understand how the wing geometry is being changed in order to bring about these changes in the attributes. These changes are shown in the bottom five rows of Figure 92. The values of TR, Λ_{LE} , and RBMR do not change, the value of \mathcal{R} is reduced from its current value of 7.5 to 6.6, and the value of S is increased from its current value of 1725 to 1875.

In order to underscore the usefulness of the inverse-like functions $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ for quickly exploring tradeoffs, a prediction profiler of the original design analysis, $\mathbf{y} = f(\mathbf{x})$ is shown in Figure 93. As before, scatter plots of the projected sampled frontier are shown

⁴The “current design” values listed here are ready by checking where the red curves intersect with the vertical dashed lines. These values are also marked by the red horizontal lines, and may be read from any column, as each column (always) shows the same current design. The values after making the change are read by checking where the red curves in the third column intersect the $\psi_{Fuel\ Capacity} = 0.8$ line.

behind each curve. The prediction profiler in Figure 93 is very useful for understanding how changes in the design variables affect the attributes, but it is almost useless as an aid for exploring changes that maintain Pareto optimality. Consider again the above scenario of a designer who wishes to explore higher fuel capacity designs. From Figure 93, it is clear that this must be accomplished by increasing S – the only curve in Figure 93 that has a positive slope vs. Fuel Capacity. However, Figure 93 also shows that increasing S will decrease ML/D and increase Wing Weight. It is nearly impossible to determine the corresponding changes that should be made in the remaining design variables as S is increased to ensure that the new current design is Pareto optimal, i.e. that it increases Fuel Capacity while minimally degrading ML/D and Wing Weight. In contrast, this is easy to surmise from Figure 92, which guarantees Pareto optimality.

Even with the addition of the projected Pareto frontier scatter plots, it is difficult to determine if the current point is non-dominated by using Figure 93. Even if the current design falls within the gray region of every plot, this does not guarantee that it is on the Pareto frontier. The converse, however is true: if the current design lies outside the gray field in any graph, then the current design is not on the Pareto frontier.

8.4 Effect of Sampling Size on Coordinate System Quality

In many real design problems, the number of design analysis function calls that can be feasibly evaluated by the multi-objective optimizer is limited by the availability of computing resources. The examples presented in the previous sections demonstrate the coordinate systems on well-sampled Pareto frontiers for which the number of design analysis function calls is effectively unconstrained. The intent of this section is to examine the extent to which the quality of the coordinate system degrades when the number of attainable sampled points is limited.

Three newly sampled frontiers of the wing design problem, obtained using the mod-NBI optimizer, are considered in this section. Frontiers sampled by NSGA-II are not considered here since NSGA-II is known to perform poorly with small populations. The number of design analysis function calls used by the mod-NBI optimizer is controlled by reducing

the number of weight levels, which in turn reduces the total number of sampled points, and by increasing the convergence tolerances⁵ of the `fmincon` algorithm used to solve the subproblems to 10^{-5} . The mod-NBI parameters used to sample these frontiers are listed in Table 9 along with the parameters of the 50-level baseline case that was considered previously in this chapter. The numbers of function evaluations listed in Table 9 are the actual counts obtained from sampling the frontiers using the parameters listed in the first three columns.

Table 9: Cases Evaluated in Study of Coordinate System Quality vs. Sample Size

Number of Weight Levels	Tolerance	Sampled Points	Function Evaluations
50 (Baseline)	10^{-8}	1275	74,790
31	10^{-5}	496	23,916
21	10^{-5}	231	11,768
11	10^{-5}	66	4002

As before, radial basis functions were created for each of the three new cases to predict the values of each design variable and attribute. Unlike the radial basis functions that were considered previously in this chapter, these new RBFs were created without withholding any data points for validation; instead, these RBFs are validated against the baseline 50-level case. This approach was used because, especially in the 11-level case, the number of sampled points is too small to withhold enough points to meaningfully validate the models while retaining enough points to meaningfully train the models.

Only the PSSOM coordinate system is considered in this study since it has been seen to provide similar, but marginally better, results than using the mod-NBI weights as coordinates. Two checks of the quality of the coordinate systems are considered. First, the RBFs created from each of the three newly sampled frontiers in Table 9 were used to resample the Pareto frontier at the 861 coordinate tuples in a 41-level coordinate grid. These sampled frontiers are plotted in Figure 94 using two methods. In Figure 94a the attribute values are calculated directly from the RBFs: $y = g_y(\psi)$, and in Figure 94b the design variable values

⁵The meaning of the convergence tolerance is that the subproblems will be considered solved when the change in the objective function (t) compared to the last iteration is less than $\pm 10^{-5}$ and all constraints are satisfied to within 10^{-5} .

are first calculated from the RBFs, and then the attributes are calculated from the original design analysis: $y = f(g_x(\psi))$. In each scatter plot, the corresponding sampled frontier obtained from the baseline (50-level mod-NBI) RBFs is underlaid in gray.

The results in Figure 94 show that the distribution of points across the Pareto frontier varies with the number of sampled points used to create the RBFs (listed above each column of Figure 94). For the RBFs based on the 496-point mod-NBI sampling, the resampled frontier closely resembles the baseline frontier. As the number of sampled points is reduced to 66, the distribution of points in the resampled frontier becomes uneven. This indicates that the degree of nonlinearity of the coordinate mapping has increased. Importantly, however, the resampled frontier still correctly covers the extent of the true Pareto frontier, as indicated by the black points' coverage of the gray points' in all plots in Figure 94.

Slight variations between the top and bottom rows of plots in Figure 94 can be perceived, but generally there seems to be little difference in the final sampled frontier due to calculating the attributes directly from the RBFs or from the independent variables calculated by the RBFs. This difference is considered in the second quality check, in which the error between these two approaches to calculating the attribute values is calculated. These results are plotted in Figure 95, which shows the distribution of the percent error in each case. The percent error is calculated as in Section 8.3.2: $\% \text{ error} = (f(g_x(\psi)) - g_y(\psi))/f(g_x(\psi)) \times 100\%$. The first column of Figure 95, which shows the baseline case, is identical to the last column of Figure 90. The errors for the three sparsely sampled frontiers are generally less than $\pm 2\%$, and are less than $\pm 0.5\%$ almost everywhere. As expected, the error increases as the number of sampled points decreases.

The implication of these finding is that reducing the number of sampled points on which the RBFs are based generally degrades the quality of the coordinate mapping in the sense that the mapping shows increasing local nonlinearities. This is generally undesirable, as it makes it more difficult for the user to infer the effect that changing the coordinate values will have on the attributes. Importantly, however, there is little degradation in the quality of the RBFs as the number of sampled points is reduced. The RBFs remain capable of accurately predicting consistent values for the design variables and attributes, in the sense

that the predicted design variable values do in fact map to the predicted attribute values, even for sparsely sampled frontiers.

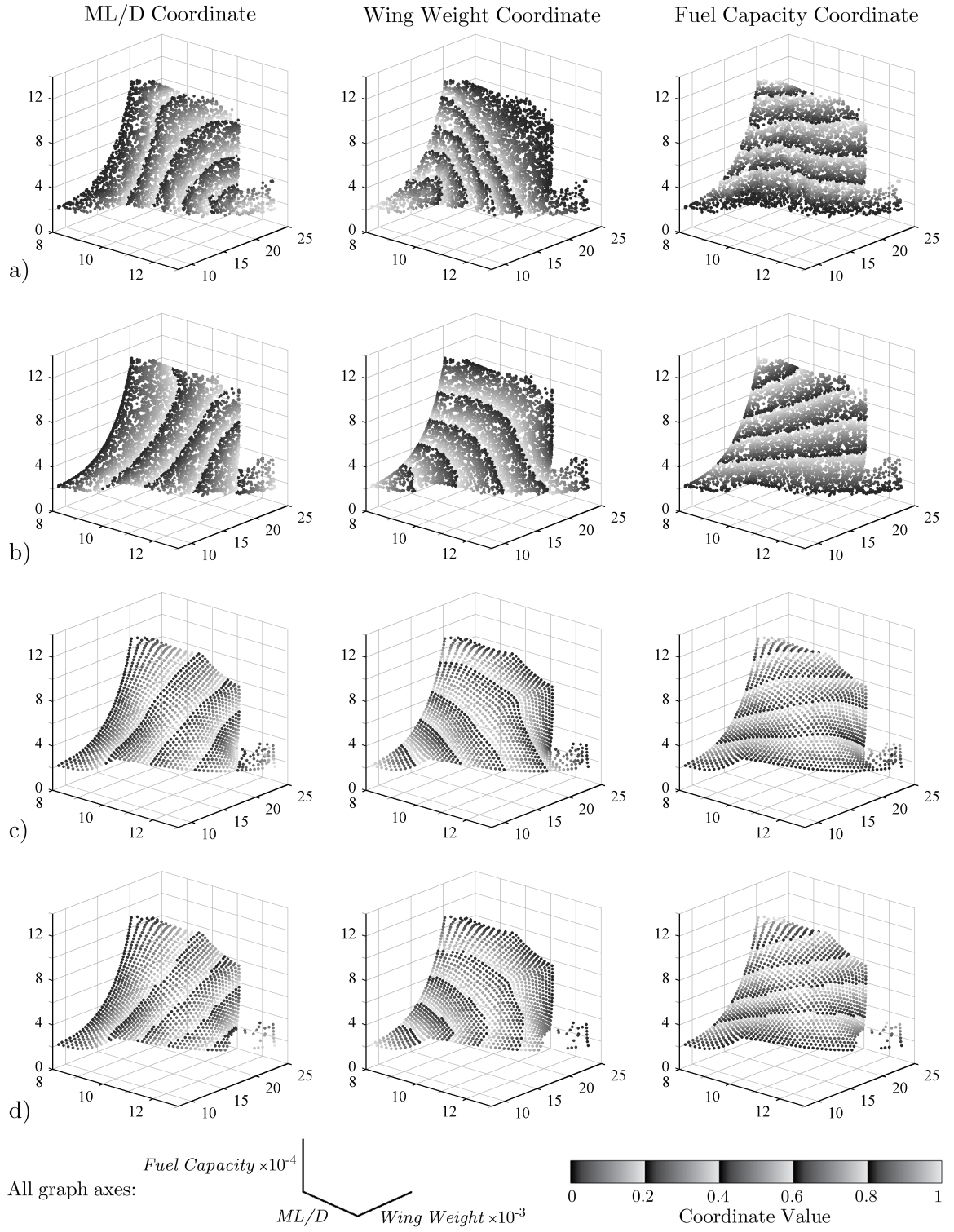


Figure 86: Sampled frontiers for wing design problem, colored by coordinates. a) NSGA-II / NDLC, b) NSGA-II / PSSOM, c) mod-NBI / Weights, d) mod-NBI / SOM

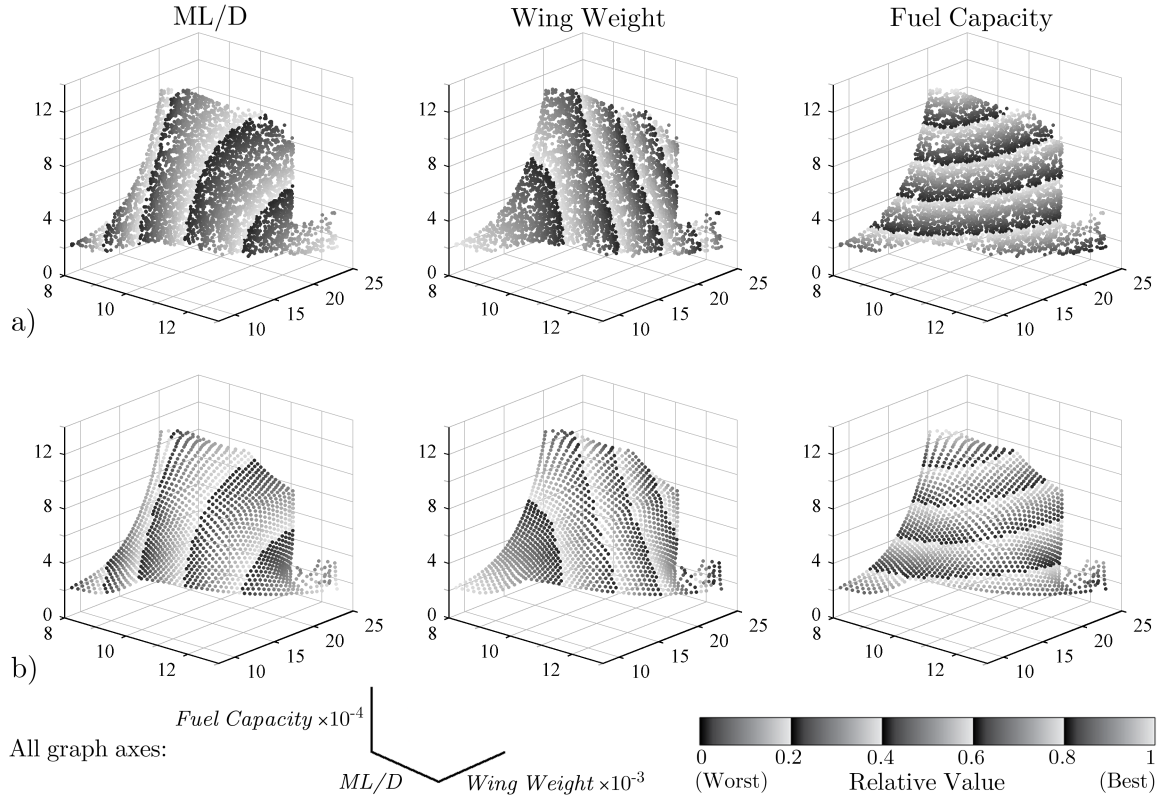


Figure 87: Sampled frontiers for wing design problem, colored by attribute value. a) NSGA-II, b) mod-NBI

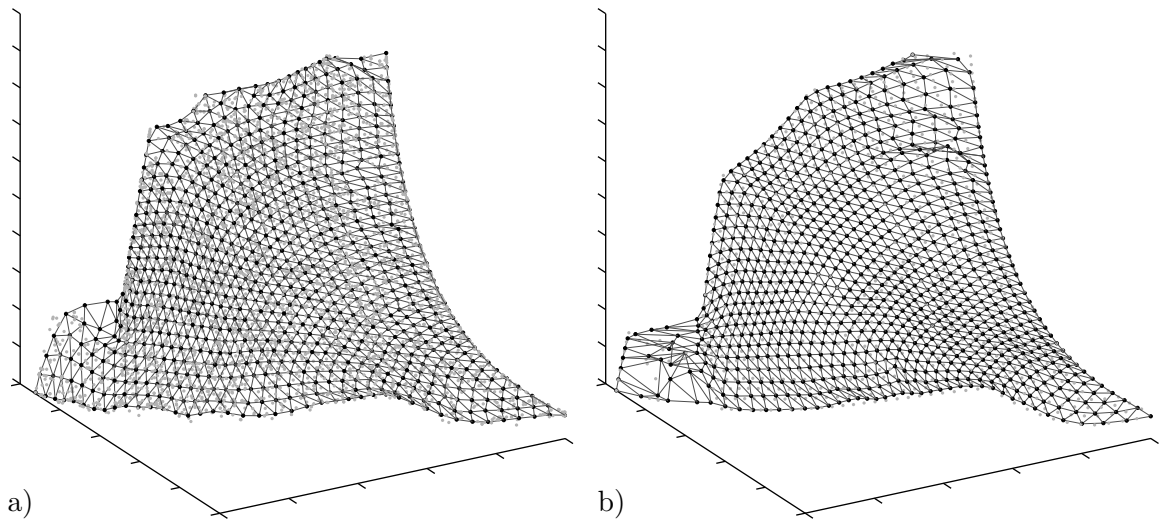


Figure 88: Final trained SOMs for wing design problem. a) NSGA-II sampling b) mod-NBI sampling

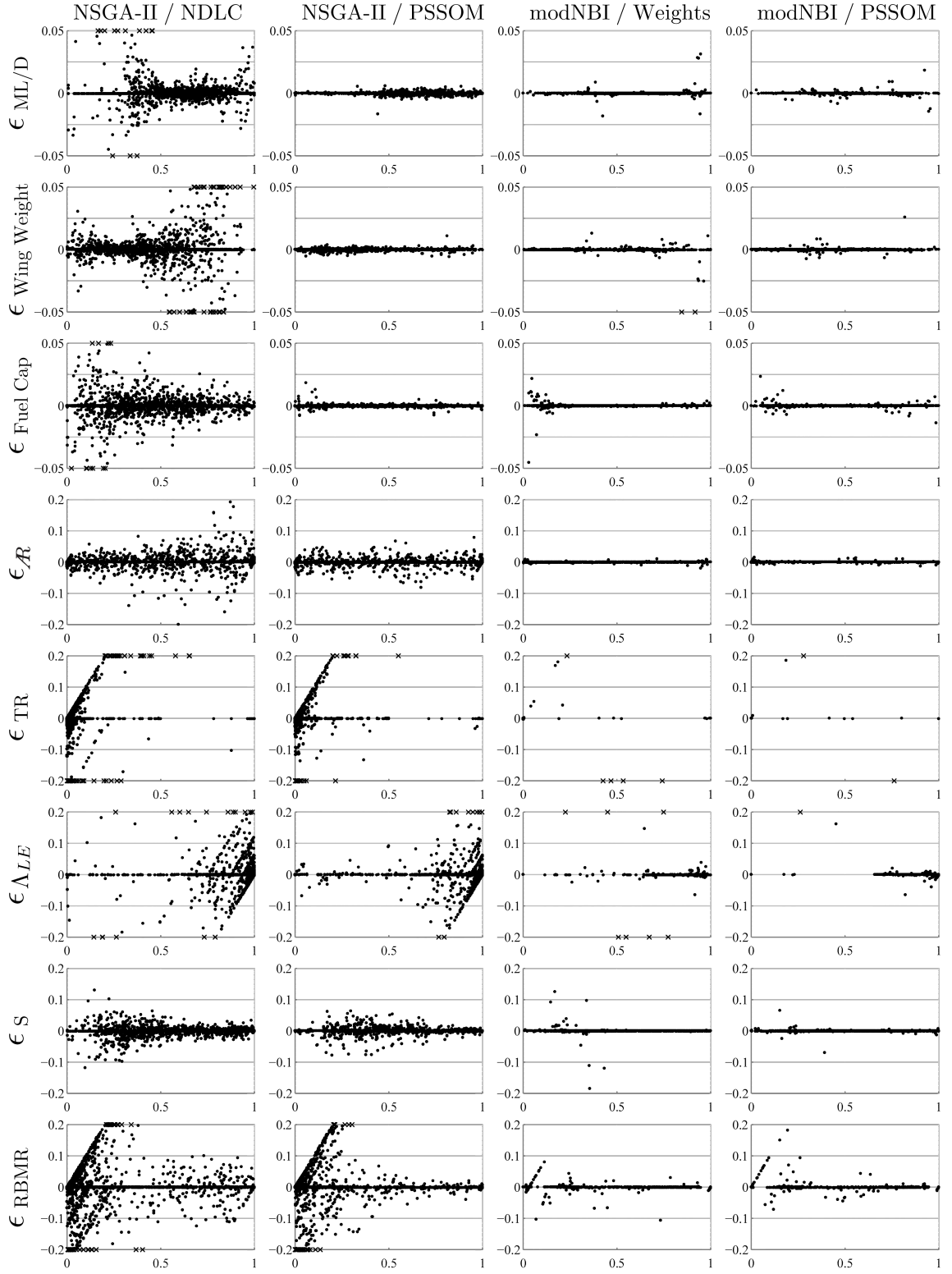


Figure 89: Residual vs. predicted plots for wing design problem

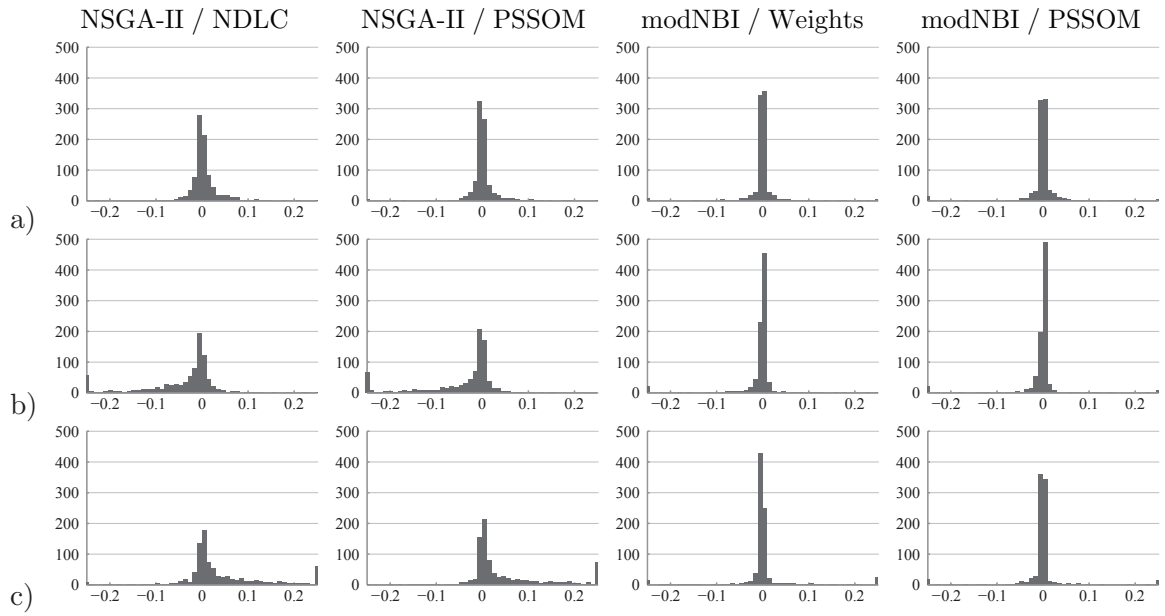


Figure 90: Distribution of percent error for RBF validation study. For all graphs, horizontal axis is percent error and vertical axis is number of points. a) ML/D; b) Wing Weight; c) Fuel Capacity

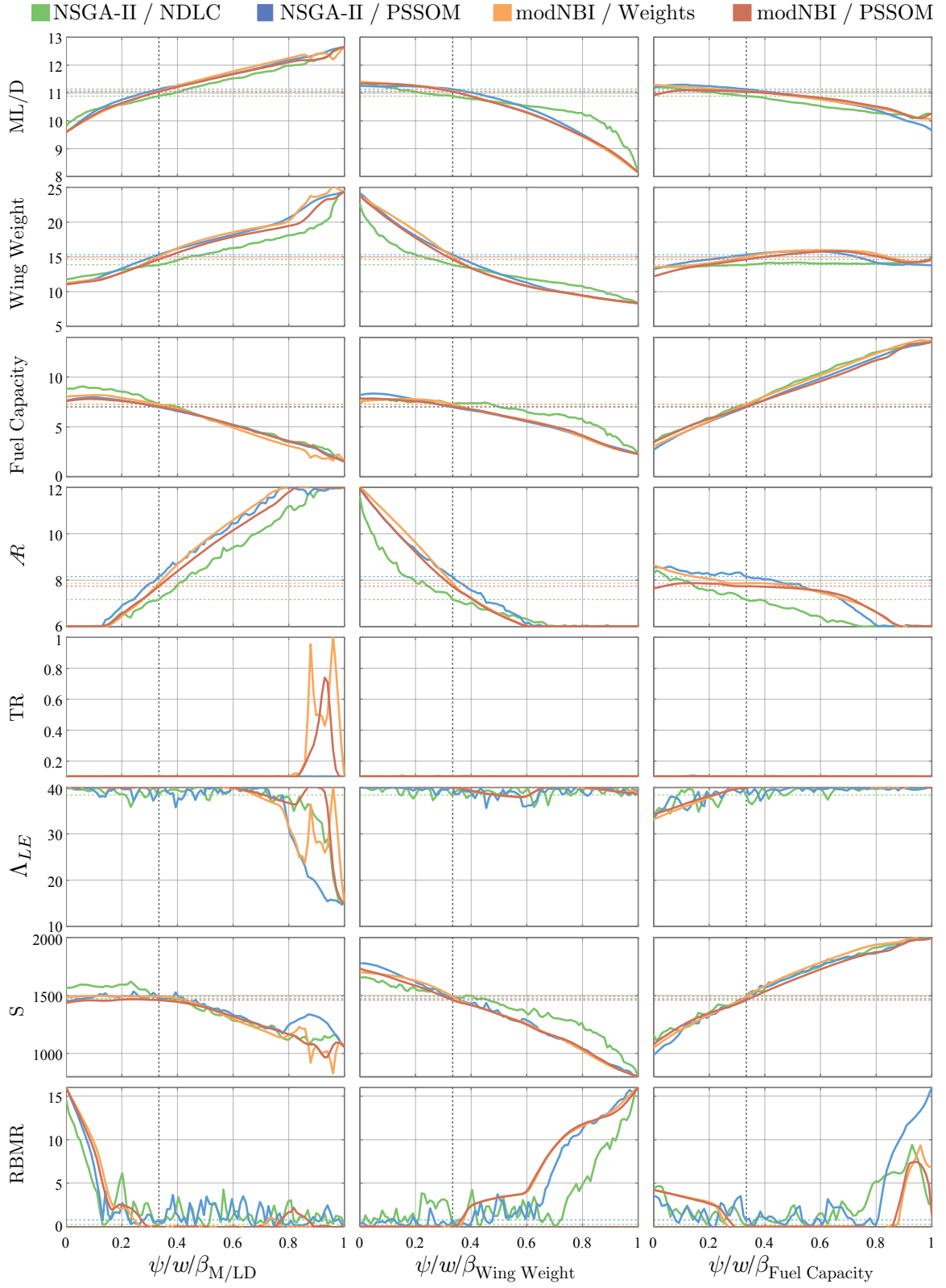


Figure 91: Prediction profiler for wing design problem, showing four coordinate systems, using coordinates as independent variables

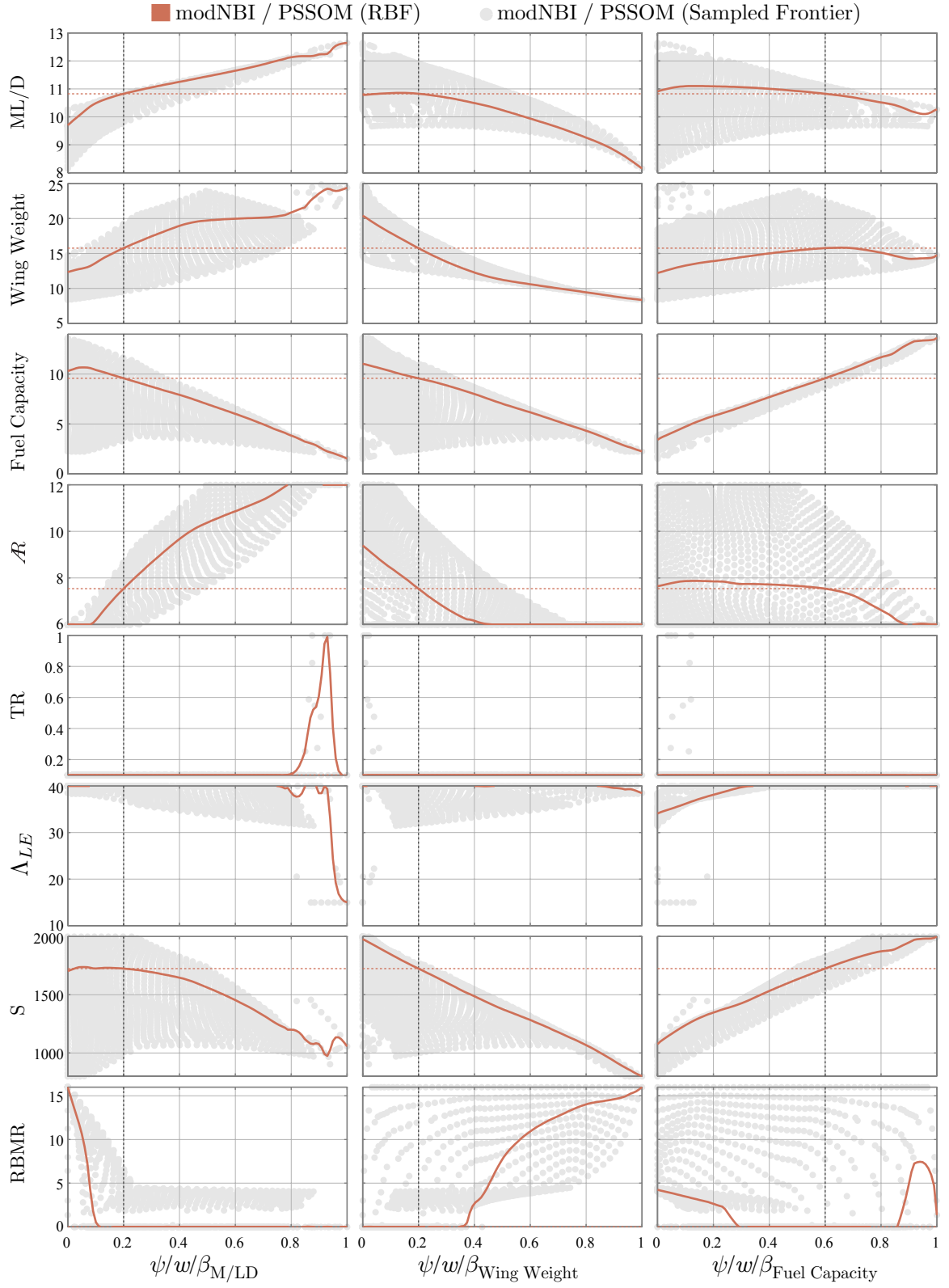


Figure 92: Augmented prediction profiler for wing design problem using coordinates as independent variables

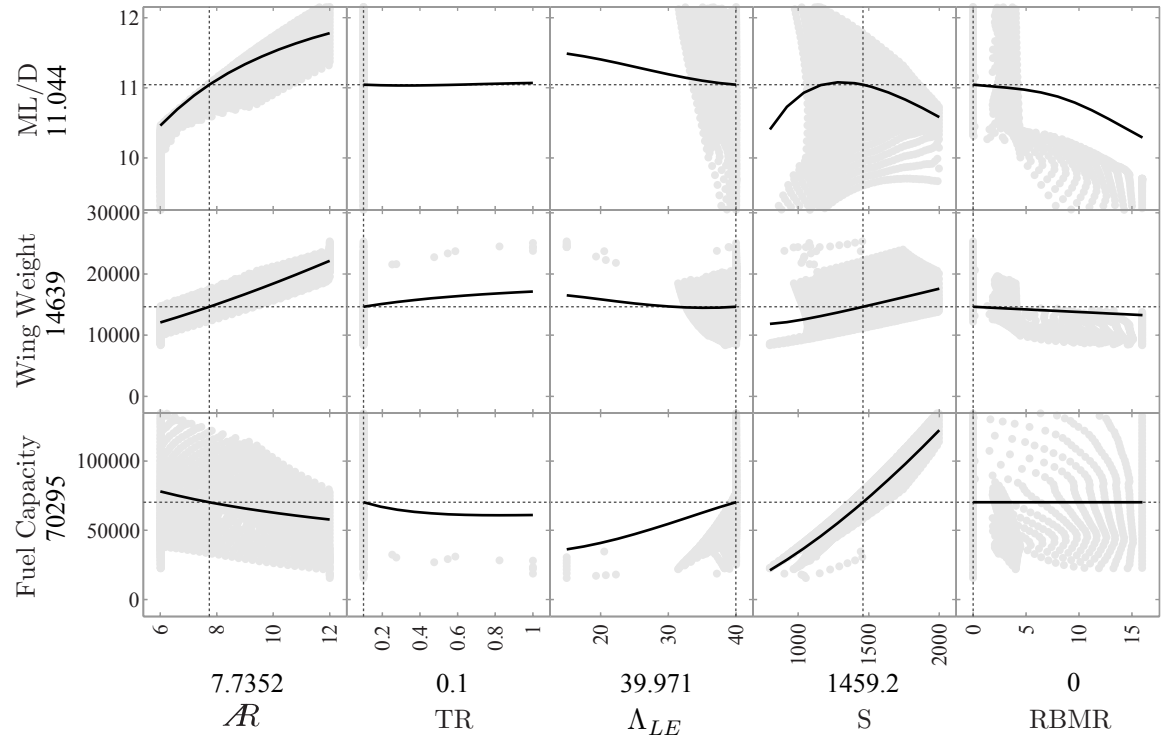


Figure 93: Augmented prediction profiler for wing design problem using design variables as independent variables

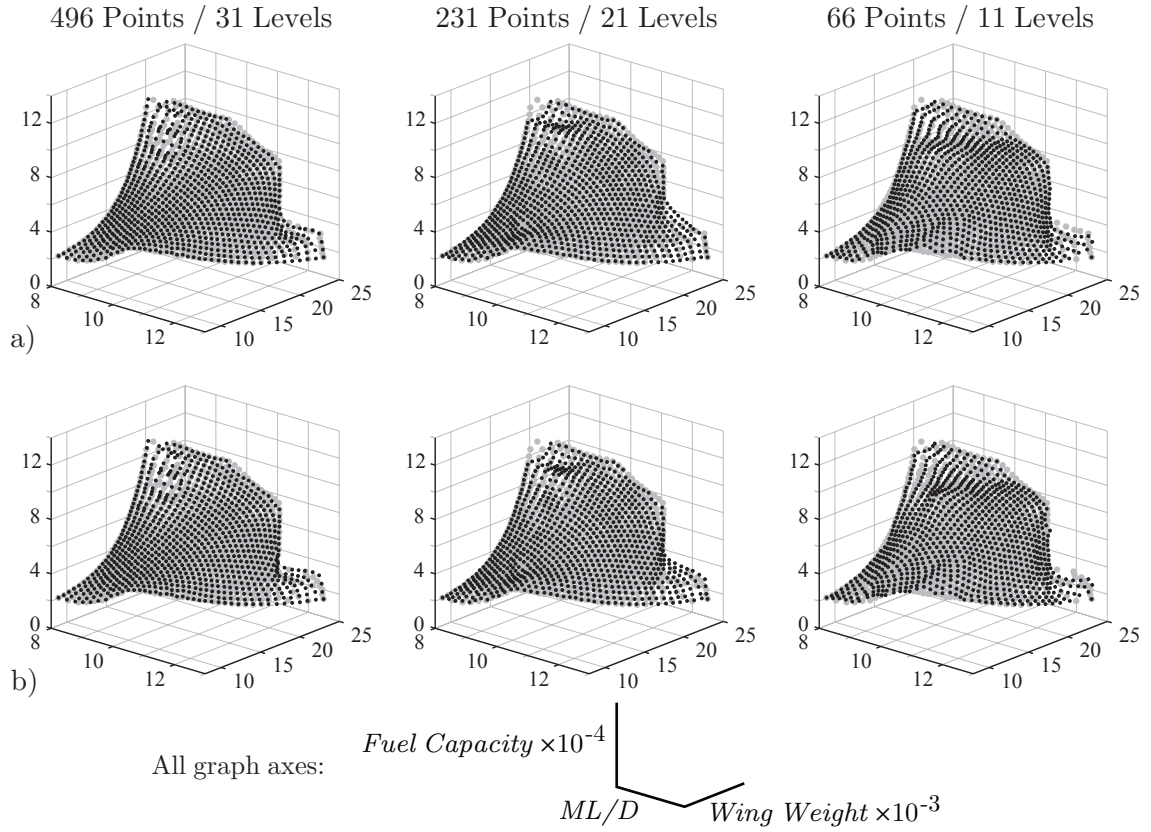


Figure 94: Comparison of resampled Pareto frontiers based on PSSOM coordinates created from sparsely sampled frontiers. a) Attribute values calculated by RBFs; b) Independent values calculated by RBFs, attribute values calculated by original design analysis.

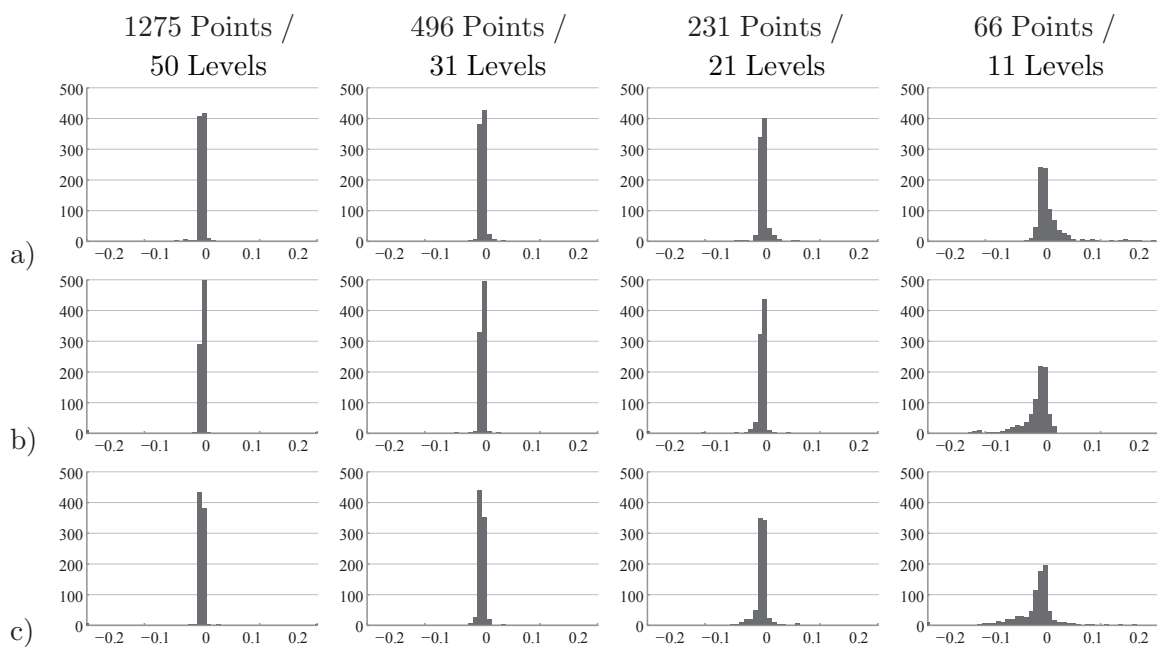


Figure 95: Error histograms for RBF validation based on PSSOM coordinates created from sparsely sampled frontiers. For all graphs, horizontal axis is percent error and vertical axis is number of points. a) ML/D; b) Wing Weight; c) Fuel Capacity

CHAPTER IX

CONCLUSIONS

9.1 *Summary and Discussion of Results*

The primary research objective of this dissertation is to develop and demonstrate methods for constructing coordinate systems that the Pareto frontiers of continuous multi-attribute design problems. The purpose of these coordinates systems is to enable intuitive exploration of the optimal design space as a continuous manifold rather than as a set of sampled points.

The overall hypothesis of this dissertation is:

Hypothesis: If the set of alternatives in a continuous, deterministic decision problem is filtered to the set of non-dominated alternatives, the resulting $(k - 1)$ -dimensional Pareto frontier of these alternatives' image in the objective space can be parameterized using a barycentric coordinate system with k coordinates. By defining such a coordinate system, the design problem may be reformulated from $\mathbf{y} = f(\mathbf{x})$ to $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ where $\boldsymbol{\psi}$ represents the barycentric coordinates. Exploration of the design problem using $\boldsymbol{\psi}$ as the independent variables will have the following advantages over exploration using \mathbf{x} as the independent variables: 1) Every vector $\boldsymbol{\psi}$ corresponds to a Pareto efficient design. 2) The number of $\boldsymbol{\psi}$ -coordinates is equal to the number of attributes regardless of the number of design variables. 3) Each attribute y_i has a corresponding coordinate ψ_i such that the sign of $\partial y_i / \partial \psi_i$ is positive if the objective is to maximize y_i and negative if the objective is to minimize y_i , i.e. each attribute improves monotonically as its corresponding coordinate increases.

The idea of reformulating the design problem to the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ in order to explore the Pareto frontier is motivated by the form of the aggregate objective functions (AOFs) used by many multi-objective optimization methods. The AOFs are parameterized by weighting coefficients on each attribute, such that varying the weights enables the

function to sample various points on the Pareto frontier. However, these functions are only able to sample points on the frontier that maximize the weighted AOF. Often this excludes concave portions of the frontier from being sampled. Even AOFs that are able to sample from the entire frontier are generally unable to provide an even sampling for an even grid of weights. This has been demonstrated in Figures 83 and 84 for the weighted sum and Chebychev norm AOFs.

The goal of this work is thus to create a method for defining a function, $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$, that is conceptually similar to the AOFs described above, but that is tailored to the geometry of a particular multi-attribute design problem's Pareto frontier such that every point on the frontier can be sampled by some vector of inputs $\boldsymbol{\psi}$, and such that an evenly spaced grid of $\boldsymbol{\psi}$ coordinate tuples maps to an evenly spaced sampling of the Pareto frontier.

The preliminary investigations of the geometry of Pareto frontiers described in Chapter 3 revealed that regular, full-dimensional Pareto frontiers have a simplex-like geometry in the sense that there is a one-to-one correspondence between the subfrontiers and the faces of a simplex of matching dimensionality. These two sets correspond in both number and dimensionality. Consequently, there is a one-to-one mapping between points on a particular face of a simplex and points on a corresponding subfrontier of the Pareto frontier. Additionally, any point on a simplex can be represented by a barycentric coordinate tuple; specifically, each point can be represented as a weighted sum of the simplex's vertices, and these weights are called barycentric coordinates. By combining these mappings, any point on a regular Pareto frontier has a one-to-one correspondence with a point in a barycentric coordinate system of matching dimensionality.

These observations establish the *existence* of a function that parameterizes the Pareto frontier with the form $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ in which $\boldsymbol{\psi}$ is a tuple of barycentric coordinates. The bulk of this work is dedicated to developing methods by which a function of this form can actually be created for a given Pareto frontier. The strategy taken here is to represent the frontier as a set of discretely sampled points obtained from a multi-objective optimizer, as this is the form in which Pareto frontiers are known in essentially all real cases. The function $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ is then defined over this sampled set by assigning to each point

in the sampled frontier a corresponding barycentric coordinate tuple. Finally, interpolants are constructed on these discretely sampled points in order to generalize the function g to arbitrary barycentric coordinate tuples, thus allowing the entire Pareto frontier to be explored.

The most critical step in this process is the mapping of barycentric coordinate tuples to the sampled frontier points. Three methods for constructing this mapping are presented here. The first method that was developed, described in Chapter 5, is named “non-domination level coordinates.” This mapping requires only a Pareto-dominance filter, which is applied repeatedly to subsets of the sampled frontier that have been projected into $k - 1$ dimensional subsets of the attributes. Of the three methods, the NDLCs generally yield the lowest quality mapping from coordinates to the sampled frontier. The example problems considered in Chapter 8 show that the NDLCs are locally noisy compared to the other coordinate mappings. However, NDLCs are the simplest of the three methods to implement, and are able to capture larger-scale trends as well as the other coordinate mappings. In the process of conducting this research, the NDLCs also served as a proof of concept and offered insight into the role of $(k - 1)$ -attribute non-domination levels in describing optimality relative to the excluded attribute.

The second coordinate system to be developed was the “Pareto-simplex self-organizing map,” described in Chapter 7. (It is presented third in this document in order to allow its use on the mod-NBI sampled frontier to be demonstrated in Chapter 7.) This coordinate system was motivated by the local noisiness of the NDLCs; the intent here is to use an iterative learning algorithm to enable the coordinate system to adapt to the sampled frontier in a way that would smooth out the local noise seen in the NDLCs. Self-organizing maps are perfectly tailored to this purpose, as the idea of a node topology corresponds well with the idea of a grid of coordinates. The results of testing the various coordinate systems on the example problems reveal that the Pareto-simplex self-organizing map coordinates are generally able to provide the most evenly spaced coordinate grid, especially when paired with the mod-NBI algorithm for sampling the frontier.

The third coordinate system to be developed was the “mod-NBI” coordinates described

in Chapter 6. The motivation here was to take the reverse approach of the Pareto-simplex self-organizing map: rather than train a grid of coordinates to conform to a sampled Pareto frontier, to evenly sample a Pareto frontier such that it naturally corresponds with an evenly spaced coordinate grid. It was discovered that a variant of normal boundary intersection by Mueller-Gritschneider et al. [113] very nearly provides this capability. The final mod-NBI algorithm uses the same linear programming approach of Mueller-Gritschneider et al. with two changes that result in a significantly more even distribution of the sampled points. The weights that parameterize the mod-NBI subproblems can serve as coordinates, or for a slightly more even coordinate mapping, a Pareto-simplex self-organizing map can be trained to the mod-NBI sampled frontier.

As described in Chapter 4, a design space exploration process that leverages one of these coordinate systems generally requires five steps, with the fifth step simply being to make a decision, thus ending the process. The first step is to sample the Pareto frontier, with the goal of producing a fairly evenly spaced sampling that covers the entire frontier. For continuous design analysis functions, the mod-NBI algorithm is very well suited for this step. As indicated by the source code in Appendix C, mod-NBI is easily implemented as a “wrapper” for an existing single-objective optimization algorithm. For design analysis functions that are better suited to an evolutionary approach, an algorithm such as NSGA-II may be used. As indicated by the results in Chapter 8, however, NSGA-II has difficulty precisely converging to the true Pareto frontier. This shortcoming leads coordinate systems based on NSGA-II to be inherently noisy, especially in their parameterizations of the preimage of the Pareto frontier in the design space. It is expected that other evolutionary algorithms would similarly exhibit this fault.

An interesting question not fully investigated in this work is how the density of the sampled frontier affects the function $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$. Since this function is created as an interpolant on the sampled frontier, the error in its output values will strongly depend on the distance between the sampled points over which it is interpolating. Many frontiers, however, are locally “well behaved” in the objective space, and a relatively sparse sampling may be suitable. There is little doubt that the samplings used to demonstrate the coordinate

systems in Chapters 5–8 are larger than they need to be. These were chosen primarily so that the scatter plot visualizations would be dense enough to visualize the frontier as being nearly a continuum of points. The most likely source of trouble when using smaller samplings is the prediction of the design variable values, i.e. $\mathbf{x} = g(\boldsymbol{\psi})$. As demonstrated in Figure 23, it is common for the active constraint set to change many times across the Pareto frontier. Each such change causes a kink in the typically smooth changes of the design variable values across the frontier. Although not demonstrated here, it is also possible for a variable to jump from its lower bound constraint to its upper bound constraint where two local Pareto frontiers intersect to form the global frontier. These discontinuities are difficult to model accurately with any interpolating function, and a sparser sampling will certainly exacerbate the modeling errors.

The second step in the Pareto Simplex Exploration process is the construction of the coordinate mappings, the relative merits of which are discussed above. The third step is to form the interpolating functions. Chapter 8 demonstrated the use of radial basis function networks (RBFs) for this purpose. These functions were found to work well in all cases, and they have the added benefit of being easily created by solving a linear system. By design, the coefficient matrix in this linear system is always square, which ensures that the resulting function is able to exactly reproduce the sampled data from which it was created.

While creating the RBFs, a subset of the sampled frontier was withheld from the fitting process so that it could later be used to independently assess the RBFs’ predictive capabilities. This is done by simply presenting the coordinates of the withheld points to the RBFs and comparing the RBFs’ calculated values for the attributes and design variables with the withheld points’ actual values. The residual vs. predicted value plots in Chapter 8 show that the RBFs are generally extremely accurate for predicting the values of the attributes, but less accurate at predicting the values of the design variables. The lack of accuracy in predicting the design variables is due in large part to the worse convergence of the design variables’ values compared to the attributes’ values. This is especially true for the samplings obtained from NSGA-II, which has no mechanism that seeks to converge the design variables to satisfy the multi-objective KKT conditions. A second significant source of errors

is, as described in the previous paragraph, the kinks in the design variables' trends as the upper and lower bound constraints become active or inactive. These errors appear in the residual vs. predicted plots in Figures 70 and 89 as diagonal bands of points near the upper and lower bounds of the design variables. Generally, the diagonal bands that emanate from exactly the upper or lower limits indicate points whose actual value was unconstrained, but whose predicted value is constrained, while the diagonal bands emanating from predicted values slightly away from the limits are points whose actual value was constrained, but whose predicted value is constrained.

The use of other types of interpolating functions is not considered here, although other appropriate options certainly exist. The radial basis functions may be modified to use a different basis function, which effectively transforms the coefficient matrix that appears in the linear system. Alternatively, a simple linear interpolation could be used based on the triangulation of the barycentric coordinate grid. This method should work well for the mod-NBI weight coordinates, which always form a regular grid that can easily be interpolated. The NDLCs and PSSOM coordinates, however, are irregularly distributed to match the distribution of the sampled frontier. As a result, these coordinates may be difficult to triangulate, especially along the edges of the coordinate space.

The fourth step in the Pareto Simplex Exploration process is to use the reformulated design analysis, $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$, to explore the Pareto frontier. Chapter 8 has demonstrated several visualization techniques that leverage this functional form, although more options than those demonstrated here certainly exist. A particularly powerful interactive visualization is the augmented prediction profiler, an example of which is shown in Figure 92. This single visualization simultaneously shows local trades around a “current design” and the entire sampled frontier. Importantly, the prediction profiler easily scales to any dimensionality, simply by adding more rows and columns of plots. When the prediction profiler is used interactively, it is very easy to converge on potentially high-value designs. These designs may then be investigated by using other visualization techniques. The primary difficulty of implementing interactive prediction profilers in a design process is that they are quite complicated and difficult to construct from scratch. Their use is gaining in popularity,

however. Several commercially available design support software tools now implement prediction profilers, as does the author’s *Rave* toolbox for MATLAB, which was used to create the examples shown in Chapter 8.

In addition to visualization, the reformulated design analysis can be used in almost any situation in which the original design analysis, $\mathbf{y} = f(\mathbf{x})$, would be used. A potentially interesting application would be to use the reformulated function as the design analysis in a single-objective optimization algorithm whose objective function is a proposed value function. Because the reformulated function is, by design, able to produce every potentially optimal design, this approach should give identical results to solving the same optimization problem using the original design analysis. The motivation for using the reformulated function here is that the radial basis functions are extremely fast and have very well-behaved derivatives, e.g. $\partial y_i / \partial \psi_j$. The single-objective optimization runs can therefore be completely almost instantaneously. The implementation strategy here would be to invest in an initial multi-objective optimization run to sample the entire frontier. This requires computing resources but little human involvement. A coordinate system and interpolating functions can then be constructed from this sampling in accordance with the methods described in this dissertation; again, this requires little human effort. When human involvement is finally needed for conducting and interpreting trade studies and assessing value, these studies may be conducted with greatly reduced cycle times.

In many ways, this proposed process is similar to a tradition surrogate-enabled design approach. The important difference here is that the models $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ are constructed for *only* the potentially optimal portion of the design space. Thus resources are not wasted sampling from the suboptimal interior of the objective space, and the predictive quality of the models is not compromised by a need to also accurately model those suboptimal regions. The particular properties of the derivatives of $(\mathbf{y}, \mathbf{x}) = g(\boldsymbol{\psi})$ are also, by design, well suited for exploration. In particular, the fact that each attribute increases monotonically, and fairly linearly, with a corresponding coordinate makes these functions well suited for exploration either by humans or optimization algorithms.

9.2 *Directions for Future Work*

As suggested in the previous section, there are many avenues for future research that involve implementing the coordinate-enabled reformulated design analysis in existing design processes. The extent to which particular common design tasks are better suited to the use of either the original design analysis or the reformulated function is largely yet to be determined. Because the largest hurdle to implementing a coordinate-enabled exploration approach is the cost of generating the initial Pareto frontier sampling, studies on the trade-offs between the size of this sampling and the quality of the resulting reformulated design analysis are potentially interesting. Another large area of future research is the development of new interactive visualizations that leverage the reformulated function or the coordinates directly, as well as studies of how designers actually use these visualizations to make decisions.

An interesting case not examined in this dissertation is the use of coordinate systems on a discontinuous Pareto frontier. Since the intent of the coordinates is to help model the frontier as a continuous manifold, their usefulness on discontinuous frontiers will certainly be diminished. However, for some minor discontinuities, e.g. for frontiers that comprise a few continuous segments, the coordinates may still prove quite useful. In any case, the coordinates should perform well for local exploration away from the discontinuities. A successful solution for parameterizing a discontinuous frontier will likely hinge on the method's ability to inform the designer about the presence of discontinuities. The closest example considered here is the bumpy sphere problem, for which all coordinate systems except the NDLCs essentially filled in the holes in the frontier – a result that is certainly undesirable. Applications to discontinuous frontiers may prove to be an interesting application of the NDLCs, which for continuous frontiers were generally inferior to the other coordinate systems.

Perhaps the greatest single limitation of the approach described in this dissertation is that it requires the Pareto frontier to have the same dimensionality everywhere. This limitation follows directly from the fact that the coordinate systems have the same dimensionality everywhere, and that there does not appear to be an intuitive way to stitch

together multiple coordinate systems of different dimensionality that could together parameterize an entire variable-dimensional Pareto frontier. Nevertheless, the *apparent* lack of an intuitive solution does not preclude the existence of one. Progress in this area will likely first require improved understanding of how variable dimensional Pareto frontiers arise, and whether their topologies obey certain rules. Particularly interesting is the question of whether variable-dimensional simplices are always (or under what circumstances are) topologically equivalent to *simplicial complexes*, which are particular arrangements of simplices that follow the rule that any intersection of simplices in the complex is a shared face of the intersecting simplices. The variable dimensional Pareto frontiers in Figures 26 and 28 are both topologically equivalent to simplicial complexes. If this property could be established in general, or conditionally on some property of the design problem, it may simplify the development of a method to parameterize variable-dimensional Pareto frontiers.

APPENDIX A

SOURCES FOR FIGURE 22

The images in Figure 22 were taken from the following publications. Across rows from top left:

1. Agrawal, G.; Bloebaum, C. & Lewis, K. Intuitive design selection using visualized n-dimensional Pareto Frontier. Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 2005, 1, 145 - 158
2. Agrawal, G.; Lewis, K. & Bloebaum, C. Intuitive visualization of hyperspace pareto frontier. Collection of Technical Papers - 44th AIAA Aerospace Sciences Meeting, Collection of Technical Papers - 44th AIAA Aerospace Sciences Meeting, 2006, 12, 8783 - 8796
3. Blasco, X.; Herrero, J.; Sanchis, J. & Martinez, M. A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization. Information Sciences, 2008, 178, 3908 - 3924
4. Ciprian, M.; Pediroda, V. & Poloni, C. Multi criteria decision aiding techniques to select designs after robust design optimization. Lecture Notes in Computer Science, 2007, 4403 NCS, 619 - 632
5. Deb, K.; Pratap, A.; Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6, 182 - 197
6. Efani, T. & Utyuzhnikov, S. V. Handling uncertainty and finding robust Pareto frontier in multiobjective optimization using fuzzy set theory. 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2010
7. Ferreira, J.; Fonseca, C. & Gaspar-Cunha, A. Methodology to select solutions from the pareto-optimal set: a comparative study. GECCO 2007 - Genetic and Evolutionary Computation Conference, 2007
8. Grierson, D. E. Resolving the Pareto optimization conundrum. 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008
9. Haddock, N. D.; Mattson, Christopher, A. & Knight, D. C. Exploring direct generation of smart Pareto sets 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008
10. Hajela, P. & Shih, C.-J. Multiobjective optimum design in mixed integer and discrete design variable problems. AIAA journal, 1990, 28, 670 - 675

11. Laumanns, M.; Rudolph, G. & Schwefel, H.-P. Spatial predator-prey approach to multi-objective optimization: A preliminary study. *Lecture Notes in Computer Science*, 1998, 1498, 241 - 241
12. Lewis, P. K.; Murray, V. R. & Mattson, C. A. A design optimization strategy for creating devices that traverse the Pareto frontier over time. *Structural and Multidisciplinary Optimization*, 2011, 43, 191-204
13. Li, Y.; Fadel, G.; Wiecek, M. & Blouin, V. Minimum effort approximation of the Pareto space of convex bi-criteria problems. *Optimization and Engineering*, 2003, 4, 231-261
14. Logist, F.; Houska, B.; Diehl, M. & Impe, J. V. Fast Pareto set generation for nonlinear optimal control problems with multiple objectives. *Structural and Multidisciplinary Optimization*, 2010
15. *Ibid.*
16. Lotov, A. V. & Miettinen, K. Visualizing the Pareto Frontier. In *Multiobjective optimization*, Springer Berlin/ Heidelberg, 2008, 213-243
17. Maginot, J.; Guenov, M.; Fantini, P. & Padulo, M. A method for assisting the study of Pareto solutions in multi-objective optimization. *Collection of Technical Papers - 7th AIAA Aviation Technology, Integration, and Operations Conference*, 2007, 2, 993 - 1010
18. Marler, R. T. & Arora, J. S. The weighted sum method for multi-objective optimization: New insights. *Structural and Multidisciplinary Optimization*, 2010, 41, 853 - 862
19. Mattson, C. A. & Messac, A. Pareto frontier based concept selection under uncertainty, with visualization. *Optimization and Engineering*, 2005, 6, 85 - 115
20. Messac, A.; Sundararaj, G. J.; Tappeta, R. V. & Renaud, J. E. Ability of objective functions to generate points on nonconvex Pareto frontiers. *AIAA Journal*, 2000, 38, 1084 - 1091
21. Miglierina, E.; Molho, E. & Recchioni, M. Box-constrained multi-objective optimization: A gradient-like method without "a priori" scalarization. *European Journal of Operational Research*, 2008, 188, 662 - 682
22. Mourelatos, Z. P. & Liang, J. A methodology for trading-off performance and robustness under uncertainty. *Journal of Mechanical Design, Transactions of the ASME*, 2006, 128, 856 - 863
23. Mullur, A. A. & Messac, A. Higher metamodel accuracy using computationally efficient Pseudo Response Surfaces. *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2005, 5, 3169 - 3185
24. Mullur, A. A.; Messac, A. & Rangavajhala, S. Case studies in Pareto set identification using Pseudo Response Surfaces. *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2008, 2008 - 2147

APPENDIX B

SOURCE CODE FOR NON-DOMINATION LEVEL COORDINATES

The source code for calculating the non-domination level coordinates in Chapter 5 in MATLAB is shown below. Three functions are presented here:

- `calculateNonDominationLevelCoordinates.m` - The main function that is called to create NDLCs. Calculates the robust non-domination levels for each $(k-1)$ subset of the attributes, and normalizes each point's coordinates.
- `calculateRobustNonDominationLevels.m` - The function for calculating the robust non-domination levels. Repeatedly calculates the non-domination levels of a random subset of data points and averages the resulting values.
- `calculateNonDominationLevels.m` - Calculates the non-domination levels of a data set. Based on Deb's " $O(MN^2)$ " algorithm in [38, p. 43]. Includes subfunctions for determining which points dominate or are dominated by a given point and to count how many points dominate a given point.

B.1 calculateNonDominationLevelCoordinates.m

```
function [ndlcs]=calculateNonDominationLevelCoordinates(data,targets,rate,rep)
%calculates nondomination level coords for the data in matrix y.
%data is an n*k matrix with data points in rows, attributes in columns.
%targets is 1xk vector of inf or -inf indicating whether to maximize or ...
    minimize each attribute
%rate is the fractional sampling rate for the robust non-domination levels
%reps is the number of repetitions for the robust non-domination levels
%By Matthew Daskilewicz 2012

n=size(data,2);
robust_ranks=zeros(size(data));
for j=1:size(data,2)
    idx=setdiff(1:n,j); %Leave out the j-th attribute
    [robust_ranks_j]=calculateRobustNonDominationLevels(data(:,idx),targets(idx),rate,rep);
    robust_ranks(:,j)=robust_ranks_j; %Store these as the ranks of the left ...
        out attribute
end

%Normalize so each point's coordinates sum to 1
```

```
ndlcs=bsxfun(@rdivide,robust_ranks,sum(robust_ranks,2));
```

B.2 *calculateRobustNonDominationLevels.m*

```
function ...
    [robust_ranks]=calculateRobustNonDominationLevels(data,targets,rate,rep)
%data is an n*(k-1) matrix with data points in rows, attributes in columns.
%targets is 1x(k-1) vector of inf or -inf indicating whether to maximize or ...
    minimize each attribute
%rate is the fractional sampling rate for the robust non-domination levels
%rep is the number of repetitions for the robust non-domination levels
%By Matthew Daskilewicz 2012

n=size(data,1);
s=ceil(n*rate);
idx=zeros(s,rep);
norm_levels=zeros(s,rep);

for r=1:rep
    %Select a random subset of data points
    thisidx=randperm(n);
    thisidx=thisidx(1:s);
    idx(:,r)=thisidx(:);
    data_subset=data(thisidx,:);

    %Rank just these points
    non_dom_levels=calculateNonDominationLevels(data_subset,targets);
    non_dom_levels=non_dom_levels-1; %Begin counting levels at 0, not 1.
    norm_levels(:,r)=bsxfun(@rdivide,non_dom_levels,max(non_dom_levels)); ...
        %Normalize
end

%Calculate robust rank of each point:
robust_ranks=zeros(n,1);
for g=1:n
    isthispt=idx==g;
    if any(isthispt(:));
        robust_ranks(g)=mean(norm_levels(isthispt));
    end
end
end
```

B.3 *calculateNonDominationLevels.m*

```
function non_dom_levels=calculateNonDominationLevels(data,targets)
%Calculates non-domination levels of data, where objectives are
%inf or -inf as indicated by targets
%By Matthew Daskilewicz 2012

for j=1:size(data,2)
    if targets(j)==-inf
        data(:,j)=-data(:,j); %minimize this attribute
    end
end

%Do Deb's  $O(MN^2)$  scheme on pg 43 of Deb2001
S = false(size(data,1));
count = zeros(size(data,1),1);
```

```

for i=1:size(data,1);
    testpt=data(i,:);
    count(i)=dominancecount(testpt,data);
    S(:,i)=finddominatees(testpt,data);
end

%Assign ranks:
non_dom_levels=nan(size(data,1),1);
isaccounted=false(size(data,1),1);
r=1;

while any(~isaccounted);
    idx=find(count==0 & ~isaccounted);
    isaccounted(idx)=true;
    non_dom_levels(idx)=r;
    nsubs=sum(S(:,idx),2);
    count=count-nsubs;
    r=r+1;
end

function [count,idx]=dominancecount(testpt,data)
%counts how many points in data dominate testpt, assuming all objectives
%are maximize. Also returns the idx of these points.
idx=finddominators(testpt,data);
count=sum(idx);

function idx=finddominatees(testpt,data)
%Returns a logical index into rows of data indicating which of those points
%are dominated by the testpt. Presumes that each objective is to maximize.
normmatrix=bsxfun(@minus,data,testpt);
idx=all(normmatrix<=0,2) & ~all(normmatrix==0,2);

function idx=finddominators(testpt,data)
%Returns a logical index into rows of data indicating which of those points
%dominate the testpt. Presumes that each objective is to maximize.
normmatrix=bsxfun(@minus,data,testpt);
idx=all(normmatrix>=0,2) & ~all(normmatrix==0,2);

```

APPENDIX C

SOURCE CODE FOR MODIFIED NORMAL BOUNDARY INTERSECTION

The source code for implementing the modified normal boundary intersection approach in Chapter 6 in MATLAB is shown below. Two functions are presented here:

- `modNBI.m` - The main function that is called to sample the frontier using the modified NBI approach.
- `trigridd.m` - A function called by `modNBI.m` to create the barycentric coordinate grid.

C.1 modNBI.m

This is a pseudocode function in the sense that it must be modified before it can be used, but it is complete enough that users should easily understand which parts must be modified. In particular, the constrained single-objective optimization function (`optfun`) calls are simplified to show the relevant inputs (objectives, constraints, initial values). In order to implement this code, the user will need to customize the functions calls to suit the particular single-objective optimization algorithm that is being used. Also note that this code uses the `linprog` function from the Optimization Toolbox, and thus requires the Optimization Toolbox or a replacement linear programming function.

A nice property of the modified NBI approach is that its source code can be very easily modified to revert it to either Mueller-Gritschneider et al.'s algorithm or Das and Dennis's original algorithm. (The converse generally would not be true; if one were to code Das and Dennis's algorithm from scratch, a very different approach would likely be taken that would require extensive recoding to implement the improvements suggested in Chapter 6.) The steps needed to make these reversions reveal just how similar the three approaches really are.

In order to revert the code below to Mueller-Gritschneider et al.'s algorithm, make the following changes to the modNBI.m function:

- Remove the “or” option from the definition of checkidx, i.e.

```
checkidx = numobjsubfrontier<numobjsubfrontier(index);
```

- Change the value of p (the p -norm) to $p=1$;

In order to revert the code below to Das and Dennis's algorithm, make the following changes to the modNBI.m function:

- Remove the entire basepoint relocation segment of code (denoted by comments)
- Change the definition of A so that it is always equal to the identity matrix, i.e.

```
A=eye(numobj);
```

```
function sampledfrontier=modNBI
%Modified NBI Pseudocode. Based on the 2009 method of Mueller-Gritschneider et
%al.
%The following variables appear in this pseudocode function without being
%declared first. The user must modify this code to properly declare these
%variables before the code will run. These variables are:
% numpts= <The number of points per edge of the weight grid>
% numobj= <The number of attributes>
% optfun= <Function handle of constrained single-objective optimization ...
%         algorithm, e.g. @fmincon>
% f = <The function that calculates attributes = f(independent variables)>
% f.i = <The function that calculates i-th attribute = f.i(independent ...
%         variables)>
%Note: This pseudo code function does not describe how to enforce
%constraints other than the NBI equality constraint. The user must code any
%additional constraints in the calls of "optfun".
% By Matthew Daskilewicz, 2012

%solve for univariate optima:
for i=1:numobj %for each attribute...
    x_individual_optima{i}=optfun(f.i); %do optimizer
    individualoptima(:,i)=f(x_individual_optima{i}); %store results
end

%form weight grid:
weights=trigriddn(numpts,numobj);
%NOTE: Source code of trigriddn is given below

%"sortedweights" will be used below to manage basepoint relocation
sortedweights=sort(weights,2);
[sortedweights,order]=sortrows(sortedweights);

%Reorder weight grid to be in a "inward spiraling" order.
%The subproblems will be solved in this order.
```

```

weights=weights(order,:);
sortedweights=sortedweights';

%Determine which subfrontier each weight tuple is on:
numobjsubfrontier=sum(logical(sortedweights),1);

%Initialize basepoints as weighted sums of the individual optima:
%(These are the classic NBI basepoints)
basepoints=zeros(size(weights));
for j=size(weights,2):-1:1
    basepoints(:,j)=individualoptima*(weights(:,j));
end

%Make somewhere to store optimization results:
finalobjectives=zeros(size(weights));
finalindependents=cell(1,size(weights,2));

%store results of individual minima (because weights has been sorted
%these will be the first numobj columns of weights).
for i=1:numobj
    finalindependents{i}=x-individualoptima{i};
    finalobjectives(:,i)=individualoptima(:,i);
end

%Calculate utopia point and quasinorm search direction:
utopiapoint=diag(individualoptima);
phi=bsxfun(@minus,individualoptima,utopiapoint(:));
quasinorm=-phi*ones(size(phi,1),1);

%For NBI subproblems, the set of independent variables will be appended
%with "t", the distance along the search direction. This set will be
%called "X".

%Form NBI subproblem objective function. This objective function takes in X
% and simply returns "-t" as the objective value (I.e. we want to maximize
% t, the distance along the search direction);
obj=@(X) -X(end);

%Our design analysis function f must be wrapped with another function
%that removes the "t" from X to avoid a "too many input arguments"
%error when calling f:
F=@(X) f(X(1:end-1));

%SOLVE SUBPROBLEMS:
for index=numobj+1:size(weights,2) %"index" is the index of the basepoint ...
    currently being solved
    %Individual optima have already been solved
    %so start looping at numobj+1

    %What subfrontier is this point on?
    subfrontindex=weights(:,index)~=0;

    %What dimension subfrontier is this point on?
    subfrontdim=numobj-nnz(subfrontindex)+1;

    %BASEPOINT RELOCATION SEGMENT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Which previously solved points should be considered in the LP?
    checkidx=numobjsubfrontier<numobjsubfrontier(index) |...
        (numobjsubfrontier==numobjsubfrontier(index) & ...

```



```

        sortedweights(subfrontdim,:) < sortedweights(subfrontdim,index));

%Make matrices of these points' weights and f(x) values
WB=weights(:,checkidx);
FB=finalobjectives(:,checkidx)';

%The weights of the current point...
wk=weights(:,index);

%Calculate dk vector of distances from wk to points in WB
p=2; %p-norm value
dk=sum(abs(bsxfun(@minus,WB,wk)).^p,1).^(1/p);

%Solve for ak vector: ak = argmin dk'ak s.t. WB*ak = wk and ak>=0
ak=linprog(dk,[],[],WB,wk,zeros(numel(dk),1),inf(numel(dk),1));

%Determine the new basepoint location:
basepoints(:,index)=FB'*ak;
%End basepoint relocation segment%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Form "A" matrix indicating which attributes to ignore:
A=double(diag(subfrontindex));

%Form NBI equality constraint for this subproblem: (note, X(end) is t)
eqconstraint = @(X) A*basepoints(:,index) + X(end)*A*quasinorm - A*F(X); ...
    %see definition of F above

%Determine initial X value: use independent variable values of the
%nearest point that has already been solved (similar to dk above but
%consider ALL previously solved points:
dists=sum(abs(bsxfun(@minus,weights(:,1:index-1),weights(:,index))).^2,1).^(1/2);
idx=find(dists==min(dists),1);
initialX=[finalindependents{idx} 0]; %always reset t to 0

%RUN OPTIMIZER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
finalindependents{index}=optfun(OBJ,eqconstraint,initialX);
finalobjectives(:,index)=f(finalindependents{index});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

```

C.2 *trigriddn.m*

This function creates a barycentric coordinate grid, weights, with numpts values per coordinate for a problem with numobj attributes. The code is not particularly easy to interpret, but it is efficient and works as-is.

```

function weights=trigriddn(numpts,numobj)
% By Matthew Daskilewicz, 2012
x=0:(numpts-1);
outs=cell(numobj,1);
[outs{:}]=ndgrid(x);
outs=permute(outs,[2,3,1]);
outs=cell2mat(outs);
outs=reshape(outs,[numpts^numobj,numobj]);
keep=sum(outs,2)==numpts-1;
weights=outs(keep,:);
weights=weights./(numpts-1);

```

APPENDIX D

SOURCE CODE FOR PARETO SIMPLEX SELF ORGANIZING MAPS

The source code for implementing the Pareto simplex SOMs in Chapter 7 in MATLAB is shown below. The MATLAB Neural Network Toolbox and Optimization Toolbox are required in order to use this code. Three functions are presented here:

- `makePSSOMCoordinates.m` - The main function that is called to create the Pareto simplex SOM and the corresponding coordinate system.
- `simplextop.m` - Creates the simplex node topology
- `learnPSSOM.m` - The learning function for creating a PSSOM.

The functions `simplextop.m` and `learnPSSOM.m` are modified versions of the Neural Network Toolbox functions `hextop.m` and `learnsomb.m`. For this reason, the code below does not show the entirety of these functions, but only the parts that have been changed and a description of how these changes can be implemented.

D.1 makePSSOMCoordinates.m

This function is called by the user to create a Pareto simplex SOM from a data set. The inputs are described in the comments of the function. In order to run this function in MATLAB, the Neural Network Toolbox and the Optimization Toolbox must be installed.

```
function finalcoords=makePSSOMCoordinates(infile,idx,targ,m)
%Make Pareto Simplex Self Organizing Map and associated Coordinate System
%Inputs:
% infile: a string indicating the path to a .txt flat file of a sampled
%   Pareto frontier. Columns of this file are variables, rows of this file
%   are sampled points on the frontier.
% idx: a vector indicating which columns of infile to use to create the
%   SOM. I.e. these columns contain the attribute values
% targ: a vector, the same size as idx, whose values are inf or -inf
%   indicating whether to maximize or minimize each attribute
% m: an odd scalar indicating how many nodes to place on each edge of the
%   SOM topology. (Must be odd due to limitations of NNet Toolbox)
%Outputs:
```

```

% finalcoords: A matrix with one row for each row of infile, and as many
%   columns as idx. Each value is the corresponding point/objective's
%   barycentric coordinate.
% By Matthew Daskilewicz, 2012

%% load pareto front data:
data=importdata(infile);
x=data.data(:,idx);

%normalize data so variables range 0-1:
x=bsxfun(@minus,x,min(x));
x=bsxfun(@rdivide,x,max(x));

%reformulate so "max is best" for all variables:
for j=1:size(x,2)
    if targ(j)==-inf
        x(:,j)=-x(:,j)+1; %min is best
    elseif targ(j)==inf
        x(:,j)=x(:,j); %max is best
    end
end

%nntoolbox likes everything transposed:
x=x';

%how many attributes?
n=numel(idx);

%What subfrontiers need to be considered?
sfts=nchoosek([1:n zeros(1,n-1)],n);
sfts=unique(sfts,'rows');
sfts=sort(sfts,2);
sfts=sortrows(sfts);

%which subfrontiers are also subfrontiers of the subfrontiers?
subsfts=false(size(sfts,1));
for s=1:size(subsfts,1);
    theseobj=[0 sfts(s,:)];
    subsfts(:,s)=all(ismember(sfts,theseobj),2);
end

%Points that are on subfrontiers of equal dimensionality are trained
%simultaneously, so we only need to divide points according to the
%lowest-dimensional subfrontier that they are on:
isthisdim=false(size(x,2),n);
sdim=sum(sfts>0,2);
onthisfront=false(size(x,2),size(sfts,1));

%First project the data onto the CHIM+, which helps ensure that
%paretoFront returns only the real subfrontiers. Without this, some
%points that would not be on the continuous frontier's subfrontier are on
%the discrete sample's subfrontier.
X=project3objective(x');
for d=1:n
    isthisd=find(sdim==d);
    for z=1:numel(isthisd);
        i=sfts(isthisd(z),:);
        i=i(i>0);
        onthisfront(:,isthisd(z))=paretoFront(X(:,i));
    end
end

```

```

    end
end

%for each point, determine the lowest dimensional subfrontier that the
%point is on:

%(first just find the first subfrontier in the list that each point is on)
for s=1:size(onthisfront,2)-1
    iidx=any(onthisfront(:,1:s),2);
    onthisfront(iidx,s+1:end)=false;
end

%(now consolidate these into groups of matching dimensionality)
for d=1:n
    isthisdim(:,d)=any(onthisfront(:,sdim==d),2);
end

%Do the same process for the points in the weight grid. Here we can just
%use the number of zero coordinates:

%form weight grid:
pts=ravetrigridn(m,n);
pts(pts<10e-12)=0;

%triangulate weights to get the sets of vertices that form cells:
t=DelaunayTri(pts(:,1:n-1));

%determine which subfrontiers each weight vector is on by locating 0
%values:
weightonthisfront=false(size(pts,1),size(sfts,1));

for s=1:size(sfts,1)
    theseobjs=sfts(s,sfts(s,:)>0); %attributes in this subfrontier
    unobjs=setdiff(1:n,theseobjs); %attributes not in this subfrontier ...
    (these should have coord=0)
    weightonthisfront(:,s)=all(pts(:,unobjs)==0,2);
end

for s=1:size(weightonthisfront,2)-1
    iidx=any(weightonthisfront(:,1:s),2);
    weightonthisfront(iidx,s+1:end)=false;
end

weightonthisorlowerfront=weightonthisfront;
for s=1:size(weightonthisfront,2)
    weightonthisorlowerfront(:,s)=any(weightonthisfront(:,subsfts(:,s)),2);
end

%now determine lowest subfrontier dimensionality of each point by counting
%its zeros:
weightdims=sum(pts>0,2)-1;
for w=1:n
    weightisthisdim(:,w)=weightdims==w-1; %#ok<AGROW>
end

%determine individual maxima:
for d=1:n
    thisopt=find(x(d,:)==max(x(d,:)));
    thisopt=thisopt(1); %in case of duplicate points, choose one.
end

```

```

        utopiapoint(:,d)=x(:,thisopt); %#ok<AGROW>
    end

    %make initial locations of SOM grid in the objective space:
    for j=size(pts,1):-1:1
        vtx(:,j)=utopiapoint*(pts(j,:));
    end

    %(Note that vtx now has the SOM vertices located at the univariate
    %optima, so we don't need to explicitly put them there)

    %% Initialize SOM. This code differs slightly depending no which MATLAB
    % version is being used:
    v=ver('nnet');
    switch v.Release
        case ...
            {'(R2007a)', '(R2007b)', '(R2008a)', '(R2008b)', '(R2009a)', '(R2009b)', '(R2010a)'}
                net=newsom(x, [m sum(1:m)/m], 'simplextop', 'linkdist', 200, 2);
                net.trainFcn='trainbuwb';
            case {'(R2010b)', '(R2011a)', '(R2011b)', '(R2012a)', '(R2012b)'}
                net= selforgmap([m sum(1:m)/m], 0, 2, 'simplextop', 'linkdist');
                net=configure(net,x);
                net.trainFcn='trainbu';
    end

    net.inputWeights{1}.learnFcn='learnPSSOM';

    %Store data that will be used in the training process:
    udata.prefs=pts;
    udata.w=vtx';
    net.IW{1}=vtx';
    udata.trainidx=1:size(x,2);
    udata.somidx=1:45;
    net.userdata=udata;

    %determine which edges link nodes in the same dimensionality subfrontiers:
    %get distances between nodes:
    dist=net.layers{1}.distances;

    %get linked nodes:
    [r1,r2]=find(dist==1); %r1/r2 are lists of the pairs of nodes connected by ...
        each link.
    edges=[r1 r2];
    edges=sort(edges,2);
    edges=unique(edges, 'rows');

    udata.alledges=edges;

    edgeonthisfront=false(size(edges,1),size(sfts,1));
    for s=1:size(sfts,1)
        edgeonthisfront(:,s)= weightonthisorlowerfront(edges(:,1),s) & ...
            weightonthisorlowerfront(edges(:,2),s);
    end

    %now group links according to the dimensionality of their subfrontiers:

    %(first just find the first subfrontier in the list that each point is on)
    for s=1:size(edgeonthisfront,2)-1
        iidx=any(edgeonthisfront(:,1:s),2);

```

```

        edgeonthisfront(iidx,s+1:end)=false;
end

%(now consolidate these into groups of matching dimensionality)
edgeisthisdim=false(size(edges,1),n);
for d=1:n
    edgeisthisdim(:,d)=any(edgeonthisfront(:,sdim==d),2);
end

%By definition, each link joins two nodes that share one coordinate in
%common. For every edge, we need to figure out what coordinate the two
%nodes share, and what is that value of that coordinate. This will allow
%the edges to be grouped into "isocoordinate curves" that contain all links
%that share the same coordinate value.

isocoordinates=nan(size(edges,1),n);

for i=1:size(edges,1)
    coords=pts(edges(i,:),:);
    delta=abs(coords(1,:)-coords(2,:));
    isocoordinates(i,delta<0.0000001)=coords(1,delta<0.0000001);
end

udata.isocoordinates=isocoordinates; %matrix isocoordinates has a line ...
    corresponding to each link, where the line contains one ~isnan value.
%That value is the coordinate value (of the corresponding column) that this
%link has constant between its two nodes.

%% Sequentially train subfrontiers of specified dimensionality:
udata.doneidx=find(weightisthisdim(:,1));
udata.lastmove=inf;

for d=n+1:size(onthisfront,2)
    trainidx=onthisfront(:,d);

    %Store indices of nodes being trained at this iteration
    somidx=weightonthisfront(:,d);
    edgeidx=edgeonthisfront(:,d);
    udata.edgeidx=find(edgeidx);
    udata.trainidx=find(trainidx);
    udata.somidx=find(somidx);
    set(0,'userdata',udata);

    net.inputWeights{1}.learnParam.steps=300; %iterations to use for batch ...
        training
    net.inputWeights{1}.learnParam.init_neighborhood=8; %initial ...
        neighborhood size
    net.trainParam.epochs=340; %this, minus learnParam.steps, is number of ...
        iterations for codebook redistribution

    %do training:
    net = train(net,x(:,udata.trainidx));

    udata.doneidx=union(udata.doneidx,udata.somidx);
end

```

```

%% Now we need to associate points in the sampled frontier with points on ...
    the SOM.
%Rather than use a nearest neighbor approach like a typical SOM would do,
%we need a continuous representation. The approach here is to treat the SOM
%like a faceted surface, and find the point on this surface that is closest
%to each point on the sampled frontier. The sampled frontier points are
%then equated with these nearest points, and the coordinates of these
%nearest points are calculated (since each face is a triangle this is a
%simple calculation).

finalweights=net.IW{1};

vertices=finalweights;
faces=t.Triangulation;

%up front calculations:
%distance from each point on the sampled frontier to each vertex:
yToSomDist=ipdm(x',vertices);

%distance to closest vertex on any face from each sampled point, and which ...
    vertex it
%is:
[yToSomMinDist,yToSomMinIdx]=min(yToSomDist,[],2);

%distance to closest vertex on EVERY face from each sampled point, and which ...
    vertex it
%is:
yToCellMinDist=zeros(size(x,2),size(faces,1),n);
for N=1:n
    theseverts=vertices(faces(:,N),:);
    yToCellMinDist(:, :,N)=ipdm(x',theseverts);
end
[yToCellMinDist,yToCellMinIdx]=min(yToCellMinDist,[],3);

%distance from SOM vertices to each other:
SomToSomDist=ipdm(vertices);

%longest dimension of each SOM face:
pairs=nchoosek(1:n,2);
maxdists=zeros(size(faces,1),1);
for p=1:size(pairs,1)
    theseverts=faces(:,pairs(p,:));
    idx=sub2ind(size(SomToSomDist),theseverts(:,1),theseverts(:,2));
    thesedists=SomToSomDist(idx);
    maxdists=max(thesedists,maxdists);
end

%limiting distances:
duoverall=yToSomMinDist;
dl=bsxfun(@minus,yToCellMinDist,maxdists');

possibleClosestCell=bsxfun(@lt,dl,duoverall);

%finally, do the actual checking of which point on each cell is closest to
%the sampled frontier points:

%things used by quadprog:
A=eye(3); %(inequality constraints)
b=[0;0;0];

```

```

E=[1 1 1]; %(equality constraint)
d=1;
options = optimset('Algorithm','active-set','Display','off','LargeScale','off');

alpha=zeros(size(x,2),size(faces,1),3);
closestpoint=zeros(size(x,2),size(faces,1),3);
dactual=zeros(size(x,2),size(faces,1));

for i=1:size(x,2)
    thispt=x(:,i);
    for j=1:size(faces,1)
        if ~possibleClosestCell(i,j);continue;end
        %form matrix of face's vertices
        thisface=vertices(faces(j,:),:);

        %form QP: a'Ha + fa + c (dont need c for optimization since it's a
        %constant, but c = sum(thispt.^2)

        H = 2*(thisface'*thisface);
        f = -2*thisface'*thispt;
        alpha(i,j,:)=quadprog(H,f,-A,b,E,d,[],[],[],options);

        %from alphas, calculate the actual closest points on each face:
        cp=thisface*squeeze(alpha(i,j,:));
        closestpoint(i,j,:)=cp;

        %distance to closest point:
        dactual(i,j)=norm(thispt-squeeze(closestpoint(i,j,:)));
    end
    disp(i);
end

dactual(~possibleClosestCell)=inf;

%find closest cell to each face:
[closestDist,closestCell]=min(dactual,[],2);

%interpolate each points actual barycentric coordinates from its nearest
%face's vertices and the alpha solution of its projection:
finalcoords=zeros(n,size(x,2));

for i=1:size(x,2)
    thisalpha=squeeze(alpha(i,closestCell(i),:)); %the weights of this point ...
    within its cell
    thisvtx=faces(closestCell(i),:); %the n vertices that make up the closest ...
    cell to this point.
    theseweights=pts(thisvtx,:);
    finalcoords(:,i)=theseweights'*thisalpha;
end

finalcoords=finalcoords';

```


D.2 *simplextop.m*

The source code shown below is used to modify the Neural Network Toolbox function `hextop.m`. In order to use this code, you must first make a copy of the `hextop.m` file named `simplextop.m`. This copy may be placed in any directory on your MATLAB search path. After creating the copy, use the MATLAB function `rehash toolboxcache` to ensure MATLAB recognizes the new file.

In order to implement this code, two sections of `hextop.m` must be modified. First, locate the line of `simplextop.m` that reads `dim = [varargin{:}];`. (The line number will differ depending on your MATLAB version.) Add a new line after this line that contains the following code:

```
dim(2:end)=dim(1)+1;
```

This line of code ensures that all sides of the simplex have the same number of nodes.

Next, go to the end of the `simplextop.m` file. In MATLAB R2010a and earlier versions, the function ends with a single `end`, while in newer versions the function ends with two `ends` on consecutive lines. For R2010a and earlier versions, place the following code after the final `end`, while in newer versions this code must be placed between the two final `ends`.

This code removes cells from the hexagonal topology to make it into a simplex.

```
ipos=pos;
ipos(2,:)=ipos(2,:)/sqrt(3);
idx=ipos(2,:)>ipos(1,:)+eps(max(dim)) | ...
    ipos(2,:)>(max(dim)-1-ipos(1,:)-eps(max(dim)));
pos=pos(:,~idx);
```

D.3 *learnPSSOM.m*

The source code shown below is used to modify the Neural Network Toolbox function `learnsomb.m`. In order to use this code, you must first make a copy of the `learnsomb.m` file named `learnPSSOM.m`. This copy may be placed in any directory on your MATLAB search path. After creating the copy, use the MATLAB function `rehash toolboxcache` to ensure MATLAB recognizes the new file.

In order to implement this code, locate the line of learnPSSOM.m that reads `% Neighborhood ... distance`. (The line number will differ depending on your MATLAB version.) Replace the section of code from that line to the end of the file with the code below.

```
% Neighborhood distance
nd = 1 + (lp.init_neighborhood-1) * (1-ls.step/lp.steps);
neighborhood = (d <= nd);

% Activations
a2 = neighborhood * a + a;
suma2 = sum(a2,2);
loserIndex = (suma2 == 0);
suma2(loserIndex) = 1;
a3 = a2 ./ suma2(:,ones(1,Q));
neww = a3*p';

%get data about which points are being trained:
udata=get(0,'userdata');

if nd<=1;
    %calculate all edge distances:
    dists=ipdm(w);

    %only consider points in the current som training idx:
    ptidx=union(udata.somidx,udata.doneidx);
    edges=udata.alledges;

    %now reduce to only those edges in this subfrontier dimension:
    edges=edges(udata.edgeidx,:);
    idx=sub2ind(size(dists),edges(:,1),edges(:,2));
    dists=dists(idx);

    oldw=w;
    ptidx=ptidx(:)'; %ensure this is a row vector
    allmoves=zeros(numel(ptidx),3);

    for i=ptidx
        %what points does this point link to?
        id=setdiff(unique(edges(any(edges==i,2),:)),i);

        j=i*ones(size(id));
        pairs=sort([j,id],2);

        if isempty(pairs);continue;end

        %how long are these edges?
        [membs,unused]=ismember(edges,pairs,'rows');
        lengths=dists(membs); %i think because id is already sorted, the ...
            ordering of dists matches the ordering of id so that junk ...
            indexes correctly into dists and there is no need for another ...
            sort here.

        meandist=mean(lengths);
        lengthdiff=lengths-meandist;

        %calculate direction vectors from this point to its linked points:
        startpt=w(j,:);
```

```

endpt=w(id,:);
direction=endpt-startpt;

%turn direction into unit vectors:
direction=bsxfun(@rdivide,direction,sqrt(sum(abs(direction),2).^2));

%move a distance lengthdiff/10 in the indicated directions:
moves=bsxfun(@times,lengthdiff/10,direction);

%apply all moves:
moves(isnan(moves))=0; %avoid divide by zero error for overlapping nodes
moves(isinf(moves))=0;
moves=sum(moves,1);
w(i,:)=w(i,:)+moves;
allmoves(i,:)=moves;
end

thismove=mean(abs(allmoves(:)));
if thismove>udata.lastmove
    neww=oldw; %do nothing, we have stopped converging
else
    neww=w; %make the moves
    udata.lastmove=thismove;
end
set(0,'userdata',udata);
w=oldw;
end

dw = neww - w;
dw(loserIndex,:) = 0;

%For codebook vectors not being trained at this iteration, make dw=0
index=1:size(dw,1);
index=setdiff(index,udata.somidx);
dw(index,:)=0;

% Next learning state
ls.step = ls.step + 1;

```

APPENDIX E

WING DESIGN ANALYSIS FUNCTION

The design analysis functions used for the Wing Design problem are based on the work by German and Takahashi [52]. The problem being considered is the preliminary sizing of the wing for a modern 150-passenger class narrowbody airliner. Three attributes of the wing are considered as design objectives:

- To maximize the product of the cruise Mach number and the lift-to-drag ratio (ML/D), a measure of aerodynamic efficiency in cruise
- To minimize the wing weight
- To maximize the wing fuel capacity.

The analysis functions that calculate these three attributes are described in this appendix.

The size class and design flight condition are indicated by the values of the parameters in Table 10, which are held constant in this analysis.

Table 10: Constant Parameters for Wing Design Problem

Parameter	Symbol	Value
Ultimate load factor	n_u	3.75
Average cruise weight	W	150,000
Wetted area, excluding wing (ft ²)	S_{wet}	5000
Cruise Mach number	M_{cruise}	0.78
Altitude	h	32,000
Kinematic viscosity (ft ² /s)	ν	3.708×10^{-4}

The independent variables for this analysis describe the wing geometry. There are five independent variables, listed in Table 11, each of which has an associated feasible range. The feasibility ranges are enforced in this study by pairs of inequality constraints on each independent variable.

Table 11: Independent Variables for Wing Design Problem

Variable	Symbol	Lower Limit	Upper Limit
Aspect Ratio	\mathcal{R}	6	12
Taper Ratio	TR	0.1	1
Leading Edge Sweep	Λ_{LE}	15°	40°
Planform Area	S	800ft^2	2000ft^2
Root Bending Moment Relief	$RBMR$	0%	16%

Several intermediate wing geometry variables must be calculated before the three attributes can be calculated. These are:

The wing span, calculated directly from the aspect ratio and wing area,

$$b = \sqrt{\mathcal{R}S}$$

The quarter-chord sweep, calculated directly from the leading-edge sweep and the taper ratio,

$$\Lambda_{c/4} = \tan^{-1} \left(\tan \left(\frac{\pi}{180} \Lambda_{LE} \right) - \frac{1 - TR}{\mathcal{R}(1 + TR)} \right)$$

The thickness-to-chord ratio is calculated to give the maximum thickness that does not violate a constraint on the drag divergence Mach number. First, the 3D and 2D lift coefficients are calculated as

$$C_L = \frac{W}{0.5\rho V^2 \cos^2(\Lambda_{c/4})}$$

$$C_{L,2D} = \frac{C_L}{\cos^2(\Lambda_{c/4})}$$

The minimum acceptable 2D drag divergence Mach number is calculated from the cruise Mach number of wing sweep,

$$M_{dd,2D} = M_{cruise} \cos(\Lambda_{c/4})$$

Finally, the thickness-to-chord ratio is calculated from the Korn equation [93]. The value 0.9 in this equation represents an aerodynamic technology level.

$$t/c = (0.9 - 0.1C_{L,2D} - M_{dd,2D}) \cos(\Lambda_{c/4})$$

In addition to the lift coefficients computed above, several coefficients related to the wing drag are calculated. The Oswald efficiency factor is calculated as a function of $RBMR$

based on a spline interpolation of five basis data points. These points and the interpolation curve are plotted in Figure 96. These basis points were extracted from data presented by Takahashi [125] in which the Oswald efficiency factor was calculated by comparing the wing's induced drag as calculated using an inviscid vortex lattice code to the induced drag computed by linear theory: $e = C_L^2 / (C_{Di} / (\pi \mathcal{R}))$ [125]. The root bending moment relief, $RBMR$, is a measure of the ellipticity of the spanwise lift distribution, such that $RBMR=0$ is an elliptical lift distribution, and $RBMR=16$ is an approximately triangular lift distribution.

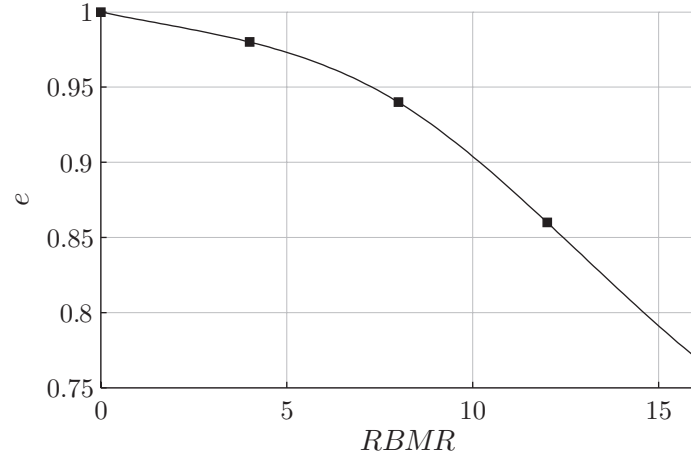


Figure 96: Spline interpolation curve for $e = f(RBMR)$. Basis data points marked are marked with ■.

The induced drag coefficient is then calculated from the aspect ratio, Oswald efficiency, and the 3D lift coefficient.

$$C_{D,i} = \frac{C_L^2}{\pi \mathcal{R} e}$$

The flat-plate wing skin-friction coefficient is calculated from the Reynolds number based on wing chord, $Re_c = Vb / (\mathcal{R}\nu)$:

$$C_{f,wing} = 0.074 Re_c^{-0.2}$$

The wing drag form factor, which relates the wing's actual zero-lift drag to the equivalent flat-plate drag, is calculated from a polynomial regression by Takahashi et al. [126] of the form $FF_{wing} = \sum Coefficient(i) Term(i)$, where the terms and their coefficients are listed in Table 12. The variable \mathcal{R}' appearing in Table 12 is a transformed aspect ratio, given by $\mathcal{R}' = \frac{\mathcal{R}}{\mathcal{R}+2}$.

Table 12: Coefficients for Form Factor Regression [126]

Term	Coefficient
<i>intercept</i>	1.09372
TR	-0.09225
t/c	1.32556
\mathcal{R}'	0.46411
$\cos(\Lambda_{LE})$	-0.83752
$TR \times t/c$	0.01553
$TR \times \mathcal{R}'$	0.16884
$TR \times \cos(\Lambda_{LE})$	0.15554
$t/c \times \mathcal{R}'$	-1.78902
$t/c \times \cos(\Lambda_{LE})$	-2.62139
$\mathcal{R}' \times \cos(\Lambda_{LE})$	0.03318
$TR \times t/c \times \mathcal{R}'$	-0.88546
$TR \times t/c \times \cos(\Lambda_{LE})$	0.49463
$TR \times \mathcal{R}' \times \cos(\Lambda_{LE})$	-0.18727
$t/c \times \mathcal{R}' \times \cos(\Lambda_{LE})$	8.00303
TR^2	-0.08280
TR^3	0.04733
t/c^2	0.02271
t/c^3	7.98567
\mathcal{R}'^2	-0.71383
\mathcal{R}'^3	0.29892
$\cos(\Lambda_{LE})^2$	1.24155
$\cos(\Lambda_{LE})^3$	-0.63223
t/c^4	-24.97043

Finally, the total zero-lift drag is calculated as

$$C_{D,0} = 1.28 \left(2FF_{wing}C_{f,wing} + (S_{wet,other}/S_{ref})FF_{other}C_{f,other} \right)$$

The multiplier 1.28 accounts for various “crud drag” sources that are otherwise not explicitly included in the terms of the above equation.

The three attributes are then calculated as follows [52].

The fuel capacity is calculated from the method in NASA’s Flight Optimization System (FLOPS) [99],

$$\text{Fuel Capacity}(lbs) = 23 \frac{(t/c)S^2}{b} \left[1 - \frac{TR}{(1 + TR)^2} \right]$$

The wing weight is calculated based on a semi-empirical equation from Kroo [80], which has been modified to account for a reduction in wing weight due to *RBMR*:

$$\text{Wing Weight}(lbs) = 4.22S + (1.642 \times 10^{-6}) (1 - RBMR) \frac{n_u \mathcal{RW} \sqrt{AS}}{(t/c) \cos^2 \Lambda_{LE}} \frac{(1 + 2TR)}{1 + TR}$$

Finally, ML/D is calculated from the given cruise Mach number and the calculated lift and drag coefficients:

$$ML/D = M_{cruise} \times CL / (C_{D,0} + C_{D,i})$$

REFERENCES

- [1] AGRAWAL, G., BLOEBAUM, C., and LEWIS, K., “Intuitive design selection using visualized n-dimensional pareto frontier,” in *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, vol. 1, (Reston, VA 20191-4344, United States), pp. 145 – 158, 2005.
- [2] AGRAWAL, G., LEWIS, K., CHUGH, K., HUANG, C.-H., PARASHAR, S., and BLOEBAUM, C., “Intuitive visualization of pareto frontier for multi-objective optimization in n-dimensional performance space,” in *Collection of Technical Papers - 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, vol. 3, (Reston, VA 20191, United States), pp. 1523 – 1533, 2004.
- [3] AGRAWAL, G., LEWIS, K., and BLOEBAUM, C., “Intuitive visualization of hyperspace pareto frontier,” in *Collection of Technical Papers - 44th AIAA Aerospace Sciences Meeting*, vol. 12, (Reston, VA 20191-4344, United States), pp. 8783 – 8796, 2006.
- [4] AGRAWAL, G., PARASHAR, S., and BLOEBAUM, C., “Estimation of multi-objective pareto frontier using hyperspace diagonal counting,” in *Collection of Technical Papers - 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, vol. 2, (Reston, VA 20191, United States), pp. 689 – 702, 2006.
- [5] AGRAWAL, G., PARASHAR, S., and BLOEBAUM, C., “Intuitive visualization of hyperspace pareto frontier for robustness in multi-attribute decision-making,” in *Collection of Technical Papers - 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, vol. 2, (Reston, VA 20191, United States), pp. 729 – 742, 2006.
- [6] ALPERN, B. and CARTER, L., “The hyperbox,” in *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pp. 133–139, 418, Oct 1991.
- [7] AMAR, R. and STASKO, J., “Knowledge precepts for design and evaluation of information visualizations,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 11, pp. 432–442, July-Aug. 2005.
- [8] ANDERSON, E., “A semigraphical method for the analysis of complex problems,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 43, pp. 923–927, 1957.
- [9] ANDREWS, D. F., “Plots of high-dimensional data,” *Biometrics*, vol. 28, no. 1, pp. 125–136, 1972.
- [10] ASIMOV, D., “The grand tour: a tool for viewing multidimensional data,” *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 1, pp. 128 – 43, 1985.
- [11] BAKER, A. P., *The role of mission requirements, vehicle attributes, technologies and uncertainty in rotorcraft system design*. PhD thesis, Georgia Institute of Technology, 2002.

- [12] BAUKOL, C. R., McDONALD, R. A., and DELMAS, N., “A user friendly interface for gaussian process metamodeling,” in *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, 2009.
- [13] BECKER, R. A., CLEVELAND, W. S., and SHYU, M.-J., “The visual design and control of trellis display,” *Journal of Computational and Graphical Statistics*, vol. 5, pp. 123–155, 1996.
- [14] BELL, T. A., JARRETT, J. P., and CLARKSON, P. J., “Multidisciplinary constraint visualization and trade-off exploration in aero engine preliminary design,” in *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, vol. 6, (Reston, VA 20191-4344, United States), pp. 3857 – 3867, 2006.
- [15] BELTON, V. and STEWART, T. J., *Multiple Criteria Decision Analysis: An Integrated Approach*. Kluwer Academic Publishers, 2002.
- [16] BILTGEN, P. T. and MAVRIS, D. N., “Technique for concept selection using interactive probabilistic multiple attribute decision making,” *Journal of Aerospace Computing, Information and Communication*, vol. 6, no. 1, pp. 51 – 67, 2009.
- [17] BISHOP, C. M., SVENSÉN, M., and WILLIAMS, C. K. I., “Gtm: The generative topographic mapping,” *Neural Computation*, vol. 10, pp. 215–234, 1998.
- [18] BRANSCOME, E. C., *A Multidisciplinary Approach to the Identification and Evaluation of Novel Concepts for Deeply Buried Hardened Target Defeat*. PhD thesis, Georgia Institute of Technology, 2006.
- [19] CARR, D. B., LITTLEFIELD, R. J., NICHOLSON, W. L., and LITTLEFIELD, J. S., “Scatterplot matrix techniques for large n,” *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 424–436, 1987.
- [20] CHERNOFF, H., “The use of faces to represent points in k-dimensional space graphically,” *Journal of the American Statistical Association*, vol. 68, no. 342, pp. 361–368, 1973.
- [21] CHI, E. H., “A taxonomy of visualization techniques using the data state reference model,” in *Information Visualization, IEEE Symposium on*, (Los Alamitos, CA, USA), IEEE Computer Society, 2000.
- [22] CHIBA, K., MAKINO, Y., and TAKATOYA, T., “Evolutionary-based multidisciplinary design exploration for the silent supersonic technology demonstrator wing,” *Journal of Aircraft*, vol. 45, no. 5, pp. 1481 – 1494, 2008.
- [23] CHIU, P.-W. and BLOEBAUM, C., “Hyper-radial visualization (hrv) for decision-making in multi-objective optimization,” in *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008.
- [24] CHIU, P.-W. and BLOEBAUM, C., “Hyper-radial visualization (hrv) with weighted preferences for multi-objective decision making,” in *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2008.

- [25] CHIU, P.-W. and BLOEBAUM, C., “Variable range-based preference incorporation in multi-objective decision making,” in *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2009.
- [26] CHIU, P.-W. and BLOEBAUM, C., “Visual steering for design generation in multi-objective optimization problems,” in *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2009.
- [27] CLARICH, A., P., G., PARASHAR, S., and RUSSO, R., “Use of multivariate-data-analysis techniques in modefrontier for efficient optimization and decision making,” in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.
- [28] COOK, K. and THOMAS, J., *Illuminating the Path: the research and development agenda for visual analytics*. IEEE Computer Society, Los Alamitos, CA, 2005.
- [29] COXETER, H. S. M., *Introduction to Geometry*. John Wiley and Sons, Inc., 1961.
- [30] CRAWFORD, S. L. and FALL, T. C., “Projection pursuit techniques for visualizing high-dimensional data sets,” in *Visualization in Scientific Computing* (GREGORY M. NIELSON, BRUCE D. SHRIVER ; ASSOCIATE EDITOR OF A SECTION, L. R., ed.), pp. 94–107, IEEE Computer Society Press, Los Alamitos, CA, 1990.
- [31] CUI, Q., WARD, M., and RUNDENSTEINER, E., “Enhancing scatterplot matrices for data with ordering or spatial attributes,” in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 6060, (USA), pp. 60600 – 1, 2006.
- [32] DAS, I. and DENNIS, J., “A closer look at the drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems,” *Structural Optimization*, vol. 14, pp. 63–69, 1997.
- [33] DAS, I. and DENNIS, J., “Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems,” *SIAM Journal of Optimization*, vol. 8, pp. 631–657, 1998.
- [34] DASKILEWICZ, M. J. and GERMAN, B. J., “Rave: A graphically driven framework for agile design-decision support,” in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.
- [35] DASKILEWICZ, M. J. and GERMAN, B. J., “Observations on the topology of pareto frontiers with implications for design decision making,” in *AIAA Aerospace Sciences Meeting*, 2012.
- [36] DASKILEWICZ, M. J. and GERMAN, B. J., “Rave: A computational framework to facilitate research in design decision support,” *Journal of Computing and Information Science in Engineering*, vol. 12, p. 021005, 2012.
- [37] DASKILEWICZ, M. J. and GERMAN, B. J., “A non-domination level coordinate system for parameterizing and exploring pareto frontiers,” *AIAA Journal*, vol. In Press, 2013.
- [38] DEB, K., *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

- [39] DEB, K., PRATAP, A., AGARWAL, S., and MEYARIVAN, T., “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182 – 197, 2002.
- [40] DONOVAN, S., TAKAHASHI, T. T., STOHR, M., and PETERMEIER, J., “Optimum transverse spanloads for transport category aircraft,” in *AIAA Aerospace Sciences Meeting*, 2011.
- [41] DOS SANTOS, S. and BRODLIE, K., “Gaining understanding of multivariate and multidimensional data through visualization,” *Computers and Graphics*, vol. 28, pp. 311–325, 2004.
- [42] DU, Q., FABER, V., and GUNZBURGER, M., “Centroidal voronoi tessellations: Applications and algorithms,” *SIAM Review*, vol. 41, pp. 637–676, 1999.
- [43] DUFRESNE, S., JOHNSON, C., and MAVRIS, D. N., “Variable fidelity conceptual design environment for revolutionary unmanned aerial vehicles,” *Journal of Aircraft*, vol. 45, no. 4, pp. 1405 – 1418, 2008.
- [44] EDDY, J. and LEWIS, K., “Multidimensional design visualization in multiobjective optimization,” in *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, 2002.
- [45] ELMQVIST, N., STASKO, J., and TSIGAS, P., “Datameadow: A visual canvas for analysis of large-scale multivariate data,” in *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on*, pp. 187–194, 30 2007-Nov. 1 2007.
- [46] ELMQVIST, N., DRAGICEVIC, P., and FEKETE, J.-D., “Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1141 – 1148, 2008.
- [47] FEINER, S. and BESHES, C., “Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds,” in *Proceedings of UIST 90 (ACM Symp. on User Interface Software and Technology)*, 1990.
- [48] FIENBERG, S. E., “Graphical methods in statistics,” *The American Statistician*, vol. 33, pp. 165–178, 1979.
- [49] FISHERKELLER, M. A., FRIEDMAN, J. H., and TUKEY, J. W., “Prim-9: An interactive multidimensional data display and analysis system,” tech. rep., Stanford Linear Accelerator Center, 1974.
- [50] FORRESTER, A. I. J., KEANE, A. J., and BRESSLOFF, N. W., “Design and analysis of ”noisy” computer experiments,” *AIAA Journal*, vol. 44, no. 10, pp. 2331 – 2339, 2006. Computer codes;Kriging prediction;Numerical noise;Objective function;
- [51] FUA, Y.-H., WARD, M. O., and RUNDENSTEINER, E. A., “Hierarchical parallel coordinates for exploration of large datasets,” in *Proceedings of the IEEE Conference on Visualization '99*, 1999.
- [52] GERMAN, B. J. and TAKAHASHI, T. T., “Planform as platform: An approach to air vehicle conceptual synthesis,” in *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2011.

- [53] HABER, R. B. and McNABB, D. A., “Visualization idioms: A conceptual model for scientific visualization systems,” in *Visualization in Scientific Computing* (NIELSON, G. M., SHRIVER, B., and ROSENBLUM, L. J., eds.), pp. 74–92, IEEE Computer Society, Los Alamitos, CA, 1990.
- [54] HAIMES, Y. Y., LASDON, L. S., and WISMER, D. A., “On a bicriterion formulation of the problems of integrated system identification and system optimization,” *IEEE Transactions of Systems, Man, and Cybernetics*, vol. 1, pp. 296–297, 1971.
- [55] HARTIGAN, J. A., “Printer graphics for clustering,” *Journal of Statistical Computation and Simulation*, vol. 4, pp. 187–213, 1975.
- [56] HAYES, C. C. and FARNAZ, A., “Design decision making: adapting mathematical paradigms to fit designers’ actual needs,” in *Proceedings of the human factors and ergonomics society 52nd annual meeting*, 2008.
- [57] HILLERMEIER, C., “Generalized homotopy approach to multiobjective optimization,” *Journal of Optimization Theory and Applications*, vol. 110, pp. 557–583, 2001.
- [58] HILLERMEIER, C., *Nonlinear Multiobjective Optimization: A generalized homotopy approach*, vol. 135 of *International Series of Numerical Mathematics*. Birkhäuser Verlag, 2001.
- [59] HOLDEN, C. M. and KEANE, A. J., “Visualization methodologies in aircraft design,” in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization conference*, 2004.
- [60] HOLUB, J., RICHARDSON, T., DRYDEN, M., LA GROTTA, S., and WINER, E., “Contextual self-organizing map visualization to improve optimization solution convergence,” in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSM*, 2012.
- [61] HONGMAN, K., RAGON, S., STUMP, G., and YUKISH, M. A., “Interactive design selection process through visualization and user guided search,” in *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, vol. 1, (Reston, VA 20191, United States), pp. 618 – 629, 2006.
- [62] HORN, J., NAFPLIOTIS, N., and GOLDBERG, D. E., “Niche pareto genetic algorithm for multiobjective optimization,” in *Proceedings of the First IEEE Conference on Evolutionary Computation; IEEE World Congress on Computational Intelligence*, vol. 1, (Orlando, FL, USA), pp. 82 – 87, 1994.
- [63] HOWARD, R. A., “The foundations of decision analysis,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 211–219, 1968.
- [64] HUBER, P. J., “Projection pursuit,” *The Annals of Statistics*, vol. 13, no. 2, pp. 435–475, 1985.
- [65] HUBER, P. J., “Experiences with three-dimensional scatterplots,” *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 448–453, 1987.
- [66] INSELBERG, A. and DIMSDALE, B., “Parallel coordinates: a tool for visualizing multi-dimensional geometry,” *Visualization, 1990. Visualization '90., Proceedings of the First IEEE Conference on*, pp. 361–378, Oct 1990.

- [67] INSELBERG, A., GRINSTEIN, C., MIHALISIN, T., and HINTERBERGER, H., “Visualizing multidimensional (multivariate) data and relations,” in *Visualization, 1994., Visualization '94, Proceedings., IEEE Conference on*, pp. 404–409, Oct 1994.
- [68] JECMEN, D. M., REDING, J. P., and ERICSSON, L. E., “An application of automatic carpet plotting to wind-tunnel data reduction,” *Journal of Spacecraft*, vol. 4, no. 3, pp. 408–410, 1967.
- [69] JEONG, S., CHIBA, K., and OBAYASHI, S., “Data mining for aerodynamic design space,” *Journal of Aerospace Computing, Information and Communication*, vol. 2, no. 11, pp. 452 – 469, 2005.
- [70] JOHNSON, C., MOORHEAD, R., MUNZNER, T., PFISTER, H., RHEINGANS, P., and YOO, T. S., “Nih/nsf visualization research challenges,” tech. rep., NIH/NSF, 2006.
- [71] KANUKOLANU, D., LEWIS, K. E., and WINER, E. H., “A multidimensional visualization interface to aid in trade-off decisions during the solution of coupled subsystems under uncertainty,” *Journal of Computing and Information Science in Engineering*, vol. 6, no. 3, pp. 288 – 299, 2006.
- [72] KEANE, A. J., “Statistical improvement criteria for use in multiobjective design optimization,” *AIAA Journal*, vol. 44, pp. 879–891, 2006.
- [73] KEENEY, R. L. and RAIFFA, H., *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge University Press, 1993.
- [74] KEIM, D., MANSMANN, F., SCHNEIDEWIND, J., and ZIEGLER, H., “Challenges in visual data analysis,” in *Information Visualization, 2006. IV 2006, Tenth International Conference on*, pp. 9–16, July 2006.
- [75] KIM, H., MALONE, B., and SOBIESZCZANSKI-SOBIESKI, J., “A distributed, parallel, and collaborative environment for design of complex systems,” in *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, 2004.
- [76] KIRBY, M. R. and MAVRIS, D. N., “An approach for the intelligent assessment of future technology portfolios,” in *40th AIAA Aerospace Sciences Meeting and Exhibit*, 2002.
- [77] KOHONEN, T., “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, 1990.
- [78] KOHONEN, T., *Self-Organizing Maps, Third Ed.* Springer-Verlag Berlin Heidelberg New York, 2001.
- [79] KOHONEN, T., “Essentials of the self-organizing map,” *Neural Networks*, vol. 37, pp. 52 – 65, 2013. Clustering problems;Data exploration;Data items;Finite set;High dimensional data;Input datas;Least-squares fittings;Linear mixtures;Similar models;Similarity;SOM;Textual database;Unknown inputs;.
- [80] KROO, I., “Component weights.” <http://adg.stanford.edu/aa241/structures/componentweight.html>, Accessed March 26, 2013.

- [81] KUHN, H. W. and TUCKER, A. W., "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (NEYMAN, J., ed.), University of California Press, Berkeley, 1951.
- [82] LEBLANC, J., WARD, M., and WITTELS, N., "Exploring n-dimensional databases," in *Visualization, 1990. Visualization '90., Proceedings of the First IEEE Conference on*, pp. 230–237, Oct 1990.
- [83] LI, X., GASTEIGER, J., and ZUPAN, J., "On the topology distortion in self-organizing feature maps," *Biological Cybernetics*, vol. 70, no. 2, pp. 189 – 198, 1993.
- [84] LIN, J., "Multiple-objective problems: Pareto-optimal solutions by method of proper equality constraints," *IEEE Transactions on Automatic Control*, vol. 21, pp. 641–650, 1976.
- [85] LINDE, Y., BUZO, A., and GRAY, R., "An algorithm for vector quantizer design," *Communications, IEEE Transactions on*, vol. 28, pp. 84 – 95, jan 1980.
- [86] LUCE, R. D. and RAIFFA, H., *Games and Decisions: Introduction and Critical Survey*. Wiley, New York, 1957.
- [87] MACKINLAY, J., "Automating the design of graphical presentations of relational information," *ACM Transactions on Graphics*, vol. 5, pp. 110–141, 1986.
- [88] MALAK, R. J. J. and PAREDIS, C. J. J., "Using parameterized pareto sets to model design concepts," *Journal of Mechanical Design*, vol. 132, p. 041007, 2010.
- [89] MARIA, A., MATTSON, C. A., ISMAIL-YAHAYA, A., and MESSAC, A., "Linear physical programming for production planning optimization," *Engineering Optimization*, vol. 35, pp. 19–37, 2003.
- [90] MARLER, R. T. and ARORA, J. S., "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, 2004.
- [91] MARLER, R. T. and ARORA, J. S., "The weighted sum method for multi-objective optimization: New insights," *Structural and Multidisciplinary Optimization*, vol. 41, no. 6, pp. 853 – 862, 2010. Apriori;Function values;Pareto-optimal sets;Weighted Sum;Weighted sum method;.
- [92] MARSAW, A., GERMAN, B., HOLLINGSWORTH, P., and MAVRIS, D., "An interactive visualization environment for decision making in aircraft engine preliminary design," in *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.
- [93] MASON, W. H., "Analytic models for technology integration in aircraft design," in *AIAA/AHS/ASEE Aircraft Design, Systems, and Operations Conference*, 1990.
- [94] MATTHEWS, P. C., *The application of self organizing maps in conceptual design*. PhD thesis, Fitzwilliam College, University of Cambridge, 2002.
- [95] MAVRIS, D. N., BANDTE, O., and DELAURENTIS, D. A., "Robust design simulation: A probabilistic approach to multidisciplinary design," *Journal of Aircraft*, vol. 36, pp. 298–307, 1999.

- [96] MAVRIS, D. N., TAI, J., and SCHRAGE, D. P., “A multidisciplinary design optimization approach to sizing stopped rotor configurations utilizing reaction drive and circulation control,” in *5th Symposium on Multidisciplinary Analysis and Optimization*, 1994. AIAA-94-4296.
- [97] MCCLAIN, S. T., TIÑO, P., and KREEGER, R. E., “Ice shape characterization using self-organizing maps,” in *1st AIAA Atmospheric and Space Environments Conference*, 2009.
- [98] MCCORMICK, B. H., DEFANTI, T. A., and BROWN, M. D., “Visualization in scientific computing,” *Computer Graphics*, vol. 21, November 1987.
- [99] MCCULLERS, L. A., “Flight optimization system release 8.12 user’s guide,” tech. rep., NASA Langley Research Center, 2009.
- [100] MCGILL, R., TUKEY, J. W., and LARSEN, W. A., “Variations of box plots,” *The American Statistician*, vol. 32, pp. 12–16, 1978.
- [101] MESSAC, A. and ISMAIL-YAHAYA, A., “Required relationship between objective function and pareto frontier orders: Practical implications,” *AIAA Journal*, vol. 39, no. 11, pp. 2168 – 2174, 2001.
- [102] MESSAC, A., ISMAIL-YAHAYA, A., and MATTSON, C., “The normalized normal constraint method for generating the pareto frontier,” *Structural and Multidisciplinary Optimization*, vol. 25, no. 2, pp. 86 – 98, 2003.
- [103] MESSAC, A., “From dubious construction of objective functions to the application of physical programming,” *AIAA Journal*, vol. 38, pp. 155–163, 2000.
- [104] MESSAC, A. and MATTSON, C. A., “Normal constraint method with guarantee of even representation of complete pareto frontier,” *AIAA Journal*, vol. 42, no. 10, pp. 2101 – 2111, 2004.
- [105] MESSAC, A., SUNDARARAJ, G. J., TAPPETA, R. V., and RENAUD, J. E., “Ability of objective functions to generate points on nonconvex pareto frontiers,” *AIAA Journal*, vol. 38, no. 6, pp. 1084 – 1091, 2000.
- [106] MIETTINEN, K. A., *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, 1999.
- [107] MIGLIERINA, E., MOLHO, E., and RECCHIONI, M., “Box-constrained multi-objective optimization: A gradient-like method without ”a priori” scalarization,” *European Journal of Operational Research*, vol. 188, no. 3, pp. 662 – 682, 2008.
- [108] MIHALISIN, T., GAWLINSKI, E., TIMLIN, J., and SCHWEGLER, J., “Visualizing a scalar field on an n-dimensional lattice,” in *Visualization, 1990. Visualization ’90., Proceedings of the First IEEE Conference on*, 1990.
- [109] MIHALISIN, T., TIMLIN, J., and SCHWEGLER, J., “Visualization and analysis of multi-variate data: a technique for all fields,” in *Visualization, 1991. Visualization ’91, Proceedings., IEEE Conference on*, pp. 171–178, 421, Oct 1991.

- [110] MIHALISIN, T. W., TIMLIN, J., GAWLINSKI, E. T., and SCHWEGLER, J. W., “Multi-dimensional graphing in two-dimensional space,” in *U.S Patent No. 5,228,119*, 1993.
- [111] MOORE, K. T. and NAYLOR, B. A., “The development of an open source framework for multidisciplinary analysis and optimization,” in *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2008.
- [112] MOTTA, R. D. S., AFONSO, S. M. B., and LYRA, P. R. M., “A modified nbi and nc method for the solution of n-multiobjective optimization problems,” *Structural and Multidisciplinary Optimization*, vol. 46, pp. 239–259, 2012.
- [113] MUELLER-GRITSCHNEDER, D., GRAEB, H., and SCHLICHTMANN, U., “A successive approach to compute the bounded pareto front of practical multiobjective optimization problems,” *SIAM Journal of Optimization*, vol. 20, pp. 915–934, 2009.
- [114] MULLUR, A. A. and MESSAC, A., “Extended radial basis functions: More flexible and effective metamodeling,” *AIAA Journal*, vol. 43, no. 6, pp. 1306 – 1315, 2005. Metamodeling;Multidisciplinary design optimization (MDO);Nonradial basis functions;Radial basis functions;
- [115] PARASHAR, S., PEDIRODA, V., and POLONI, C., “Self organizing maps (som) for design selection in robust multi-objective design of aerofoil,” in *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008.
- [116] PENG, W., WARD, M., and RUNDENSTEINER, E., “Clutter reduction in multi-dimensional data visualization using dimension reordering,” in *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pp. 89–96, 0-0 2004.
- [117] RAKOWSKA, J., HAFTKA, R. T., and WATSON, L. T., “Tracing the efficient curve for multi-objective control-structure optimization,” *Computing Systems in Engineering*, vol. 2, pp. 461–471, 1991.
- [118] RAO, J. and PAPALAMBROS, P., “Nonlinear programming continuation strategy for one parameter design optimization problems,” in *American Society of Mechanical Engineers, Design Engineering Division (Publication) DE*, vol. 19-2, (Montreal, Que, Can), pp. 77 – 89, 1989.
- [119] RECCHIONI, M. C., “A path following method for box-constrained multiobjective optimization with applications to goal programming problems,” *Mathematical Methods of Operations Research*, vol. 58, no. 1, pp. 69 – 85, 2003.
- [120] ROTH, S. F. and MATTIS, J., “Data characterization for intelligent graphics presentation,” in *Proceedings of CHI '90*, pp. 193–200, ACM, New York:ACM, 1990.
- [121] SIMOV, P. and FERGUSON, S., “Investigating the significance of “one-to-many” mappings in multiobjective optimization,” in *Proceedings of the ASME 2010 International Design Engineering Technical Conferences*, 2010.
- [122] SIMPSON, T. W. and MARTINS, JOAQUIM, R. R. A., “Nsf workshop report: The future of multidisciplinary design optimization (mdo): Advancing the design of complex engineered systems,” tech. rep., National Science Foundation. Fort Worth, Texas, September 16, 2010, 2010.

- [123] STEGEL, J. H., GOLDWYN, R. M., and FRIEDMAN, H. P., "Pattern and process of the evolution of human septic shock," *Surgery*, vol. 70, pp. 232–245, 1971.
- [124] STUMP, G., LEGO, S., and YUKISH, M., "Visual steering commands for trade space exploration: User-guided sampling with example," *Journal of Computing and Information Science in Engineering*, vol. 9, 2009.
- [125] TAKAHASHI, T., "Optimum transverse span loading for subsonic transport category aircraft," *Journal of Aircraft*, vol. 49, pp. 262–274, 2012.
- [126] TAKAHASHI, TIMOTHY, T., GERMAN, B. J., SHAJANIAN, A., DASKILEWICZ, M. J., and DONOVAN, S., "Form factor and critical mach number estimation for finite wings," *Journal of Aircraft*, vol. 49, pp. 173–182, 2012.
- [127] TEAGUE, M. J. and MEYER, H. G., "A computer graphics display and data compression technique," tech. rep., NASA National Space Science Data Center, NSSDC 74-07, 1974.
- [128] THURSTON, D. L., "Utility function fundamentals," in *Decision Making in Engineering Design* (LEWIS, K. E., CHEN, W., and SCHMIDT, L. C., eds.), ch. 3, pp. 15–19, ASME Press, 2006.
- [129] TIWARI, S., DONG, H., WATSON, B. C., and LEIVA, J. P., "Visualldoc: New capabilities for concurrent and integrated simulation design," in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.
- [130] TORY, M. and MÖLLER, T., "A model-based visualization taxonomy," tech. rep., Computing Science Dept., Simon Fraser University, 2002.
- [131] TORY, M. and MÖLLER, T., "Rethinking visualization: A high level taxonomy," in *Proceedings IEEE Symposium on Information Visualization*, 2004.
- [132] TRIANTAPHYLLOU, E., *Multi-Criteria Decision Making Methods: A comparative study*. Kluwer Academic Publishers, 2000. Parlos, Panos M. (ed.).
- [133] TRINKAUS, H. L. and HANNE, T., "Knowcube: A visual and interactive support for multicriteria decision making," *Computers and Operations Research*, vol. 32, no. 5, pp. 1289 – 1309, 2005.
- [134] TUFTE, E. R., *The visual display of quantitative information, 2nd Ed.* Graphics Press, 2001.
- [135] TUKEY, P. A. and TUKEY, J. W., "Graphical display of data sets in three or more dimensions," in *Interpreting multivariate data* (BARNETT, V., ed.), pp. 189–275, Wiley, 1981.
- [136] UPSON, C., FAULHABER, T.A., J., KAMINS, D., LAIDLAW, D., SCHLEGEL, D., VROOM, J., GURWITZ, R., and VAN DAM, A., "The application visualization system: a computational environment for scientific visualization," *Computer Graphics and Applications, IEEE*, vol. 9, pp. 30–42, Jul 1989.
- [137] VALOVA, I., BEATON, D., BUER, A., and MACLEAN, D., "Fractal initialization for high-quality mapping with self-organizing maps," *Neural Computing and Applications*, vol. 19, no. 7, pp. 953 – 966, 2010.

- [138] VAN LIERE, R. and VAN WIJK JARKE J., “Visualization of multi-dimensional scalar functions using hyperslice,” *CWI Quarterly*, vol. 7, pp. 147–158, 1994.
- [139] VAN WIJK, J. and VAN LIERE, R., “Hyperslice: Visualization of scalar functions of many variables,” in *Visualization, 1993. Visualization '93, Proceedings., IEEE Conference on*, pp. 119–125, Oct 1993.
- [140] VENKATESWARAN, S., HUNT, L. R., and PRABHU, R. K., “Computational method to predict thermodynamic, transport, and flow properties for the modified langley 8-foot high-temperature tunnel,” tech. rep., NASA-TM-4374, 1992.
- [141] WARD, M., GRINSTEIN, G., and KEIM, D., *Interactive data visualization: foundations, techniques and applications*. A K Peters, Ltd. Natick, MA, 2010.
- [142] WILKINSON, L., *The Grammar of Graphics*. Springer, 2005.
- [143] WILKINSON, L., ANAND, A., and GROSSMAN, R., “High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1363 – 1372, 2006.
- [144] WINER, E. and BLOEBAUM, C., “Visual design steering for optimization solution improvement,” *Structural and Multidisciplinary Optimization*, vol. 22, no. 3, pp. 219 – 229, 2001.
- [145] WINER, E. and BLOEBAUM, C., “Development of visual design steering as an aid in large-scale multidisciplinary design optimization. part i: Method development,” *Structural and Multidisciplinary Optimization*, vol. 23, no. 6, pp. 412 – 424, 2002.
- [146] WINER, E. and BLOEBAUM, C., “Development of visual design steering as an aid in large-scale multidisciplinary design optimization. part ii: Method validation,” *Structural and Multidisciplinary Optimization*, vol. 23, no. 6, pp. 425 – 435, 2002.
- [147] WONG, P. C., CRABB, A. H., and BERGERON, R. D., “Dual multiresolution HyperSlice for multivariate data visualization,” in *Information Visualization '96, Proceedings IEEE Symposium on*, 1996.
- [148] YOON, K. P. and HWANG, C.-L., *Multiple Attribute Decision Making: An Introduction*, vol. 104 of *Quantitative Applications in the Social Sciences*. Sage Publications, 1995.
- [149] ZITZLER, E. and THIELE, L., “An evolutionary algorithm for multiobjective optimization: The strength pareto approach,” Tech. Rep. Technical Report 43, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 1998.