A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Doctorate of Philosophy, Mechanical and Aerospace Engineering

By

Nathaniel Steven Knerr

May 2020

LOST IN DESIGN SPACE: INTERPRETING RELATIONS/STRUCTURE BETWEEN DECISIONS AND OBJECTIVES IN ENGINEERING DESIGN

Nathaniel Steven Knerr, Ph. D.

Cornell University 2020

In the early phase design of engineering systems, it has become increasingly popular to use a system model and an optimizer to generate a Pareto Frontier of designs that satisfy certain objectives or criteria (e.g. cost, performance, risk). A decision maker then considers this design space and considers which design appears to best satisfy a stakeholder's requirements.

While the generation of alternatives is relatively straightforward, the stakeholder must still make sense of the potentially thousands of alternatives. Additionally, stakeholders often want to consider the possibility of changing priorities, market segmentation via design families and decision sensitivity for further investigation of system priorities. In practice, stakeholders also have many criteria (5 or more), which can be difficult to visualize or represent simultaneously. As such, we aim to create new tools and techniques to extract or aid in finding this information from large multi-criteria decision problems.

We approach this with three techniques. The first, Cityplot, uses a virtual reality representation of the design space by use of dimension reduction to represent both the engineering decisions

that create a design and the criteria that a stakeholder would care about. A human study is performed to show the validity of the approach by having human subjects perform basic tests and evaluations of real and synthetic design problems. We find the Cityplot allows the intuitive visualization of decisions and a large number of objectives. The second, MOMS-CFCs, provides an interpretation of a single linkage tree and matching procedure to find rules of thumb regarding system dependencies and decision sensitivities. While promising, a few key issues prevent the approach from being applicable in most practical applications. The third, MAPSA, models the shape of the Pareto frontier with a mean plane which parameterizes the space and a model of the residuals which characterizes the overall shape of the frontier. This enables characterization of the tradeoffs and summarization of key features of the frontier. We show some simple mathematical results and create a rough model of how such an approach can be useful for decision making.

BIOGRAPHICAL SKETCH

Nathan Knerr is a Mechanical Engineering PhD Student at Cornell University under Daniel Selva. His research interests include machine learning, statistical analysis and visualization in service to engineering design and trade-space evaluation. Nathan graduated with high honors from the University of Texas at Austin with a BS in Aerospace Engineering. He is a member of the ASME and AIAA.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Daniel Selva and my thesis committee, Guy Hoffman, Peter Frasier, and Karthik Sridharan for all of their support. I would also like to thank Marcia Sawyer for her great assistance in administrative support during my PhD at Cornell. Finally, I would like to thank Brian Kirby for being open to talk in difficult times.

Financially, I would like to thank Daniel Selva for finding funding, primarily via startup resources; that is, I should thank the students and donors of Cornell University and the taxpayers of the state Texas.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Overview and Motivation

A key challenge of practical systems engineering, perhaps *the* key challenge of systems engineering, is knowing the conditions under which one is making a good decision. In optimization and academic research on engineering design it is often convenient to treat the problem as fundamentally a simple optimization problem. This presents several issues, however; for one, the solution to an optimization problem is frequently on the boundary of the feasible domain (indeed, for linear programs, it is always on the boundary). This means that if an engineer naively implements the solution found from a numerical solver, the solution will often make critical sacrifices that endanger the overall functionality of the product (for example, taking unnecessarily large risks to cut costs). For another, people are notoriously bad at understanding their own preferences. In an engineering context, this often results in clients changing priorities partway through a project or shortly after the first analysis begins being produced. Combined, these problems present practical challenges to be remedied with new approaches to analyzing engineering design spaces.

A means to sidestep these issues is to present clients with a range of concrete options. In traditional, non-formal engineering practice, this is often implemented by repeatedly optimizing the problem for a range of constraints. These "scenarios" are then presented as a study to allow engineers to consider the range of possible solutions in the engineered system to ensure that an informed choice can be made to weigh concerns against each other. A development of this approach is to use multi-criteria decision making to generate the range of optimal solutions in a single optimization run. This Pareto surface—informally, the set of possible designs for which improved performance cannot be obtained for free—allows a client to understand the overall

tradeoffs in the metrics (e.g. cost, performance and risk) which a client would understand in concrete examples. This allows for a more informed choice; rather than attempting to decide on preferences *a priori*, clients and engineers decide on their preference *a posteriori* and hence are better informed of potential unexpected options that may not have been considered. Additionally, because the decision is between more concrete alternatives, clients are better equipped to do "common sense tests" and verify, for example, that in the pursuit of fuel economy our automobile design hasn't become a motorcycle. It also enables less abstract decision makers to "envision" how a given solution might work in the field and choose a solution that better fits their objectives.

Simultaneously, there are key questions that are asked regarding how the designs relate to one another as decision variables change. In practice, programs change as clients discover new priorities and need to change the project midway; existing systems need to be updated without taking down service entirely; decision makers want to make a decision "on the margin" where performance appears to be locally optimal and there's not too much to gain without dramatically changing the overall system structure (decision variables); engineers want to know which decisions have the largest impacts on performance, cost and risk so they can prioritize those factors; decision makers want to be assured there is flexibility in early design to possibly adapt the system to other market segments. These all require an understanding not only of the specific designs, but of the entire design space—particularly of how designs relate to each other as the decisions are changed.

This highlights a key advantage of multi-criteria decision making: preservation of knowledge. In a traditional optimization-based approach to systems engineering or engineering design, preferences are specified *a priori* via some kind of utility function and then the utility

function is optimized to find a single optimal design (Hazelrigg 1998). One issue is that if the system priorities were to change, then single-objective optimizers must be rerun (and warm-starting may be impractical). Often more important, the optimizer itself will usually give little insight to the behavior of the system itself as parameters are changed. This amplifies the other issues: not only is it not clear how the "optimal" solution will change if the priorities are mis-specified, and a decision maker gains very little knowledge of how sociotechnical factors interact with the priorities and how priorities themselves may need to be re-evaluated. With *a posteriori* decision making, the "subjective" questions of what a client would prefer are ideally largely separated from the optimization process itself, allowing the optimizer to focus on "objective" questions of physics and legalities that remain largely persistent regardless of client specifics. This means that optimizer results can be saved and used between projects or to inform potential later business developments and teach the users and clients of possibilities in a project domain.

Unfortunately, there is a key problem. Such multi-criteria decision making usually generates thousands or more of possible solutions which a client must potentially consider. All the information generated from the vast space of alternatives is often overwhelming and it's not feasible to inspect thousands of possibilities all at once. Traditionally, to represent the thousands of solutions, an engineer would create several scatterplots and parallel axis plots and attempt to judge a decision based on the overall shape seen in the scatterplots. For example, a common strategy is to pick a point as close to the origin (ideal point) as possible; another strategy is to look for "knee points" where the Pareto frontier bends towards the origin and appears as a kind of local optima. However, as will be seen later, these plots can have problems that make them difficult to use and we will look to rectify these issues.

In this thesis we explore and develop novel methods to understand the tradeoff surface, particularly as it relates to how designs change in the space. We will approach the problem both visually and analytically. Visually, we will use 21st century visualization technologies to create virtual landscapes that represent the design space and allow clients to immediately see how designs relate in both the decisions that define the designs and also the important system criteria. Analytically, we will recover a hierarchy of coupled decisions that drive system criteria. Combining analytic and visual approaches, we will then take an approach inspired by the JPEG algorithm to produce reports with quantitative information describing the overall shape and features of a design space.

1.2 Previous Approaches

1.2.1 Systems Engineering and Design Space Exploration

As defined by INCOSE, Systems engineering is the interdisciplinary approach to develop large successful interdisciplinary synthesized systems (Haskins 2006). This places the discipline at an intersection of business interests (the product must be economical and be able to turn a profit) and project constraints (the project must be physically, technologically and legally realizable). In practice, this can be quite difficult: even if the physical elements are simple, the business concerns can be irregular or difficult to characterize—subject to the whims of a fickle market or shifting government priorities. In practice, the physics can often be nonlinear and multiple subsystems and sub-disciplines can conflict and need to mediate conflicts. As an example, the software engineers may want a high-powered computer system, but heat constraints may make such desires impossible. Systems engineering is then a discipline with applications from water management (P M Reed et al. 2013; Kasprzyk et al. 2012) to small and large satellite systems (Selva, Cameron, and Crawley 2014; Selva and Krejci 2013) and others (Haskins 2019).

Given the expenses and stakes of large projects, making good early decisions is key: a study by the Defense Acquisition University once found that 80-90% of a systems development costs became locked following the conceptual development phase (Haskins 2006). This is important, as an example of the costs involved in large engineering projects, the Orion/SLS program cost $30 billion in development costs alone (Strickland 2013). Given the costs at stake, any improvements in the process by which decisions are made have dramatic potential real-world savings and implications.

Approaches to deal with the inherent difficulty of the optimization task have emerged to deal with these issues. Multidisciplinary design optimization studies methods to aggregate conflicting sub-disciplines with separate models for optimization and constraint (Simpson and Martins 2010, 2011). Related, engineering design approaches such as Value Driven Design (Hazelrigg 1998) or Robust Engineering (Beyer et al. 2007) attempt to recognize the existence of optimization as a model for the engineering process. Value-driven design attempts to coalesce all the subsystems "values" into an overall value/cost model which is then optimized to yield the single "optimal" design. Robust engineering attempts to understand how to give sufficient space around the constraints to ensure that even in a case of misspecification, the chosen solution will still satisfy the minimum constraints (e.g., the simplifying assumptions in the aerodynamics model will still permit the airplane to fly after optimization).

A competitor to these single-objective optimization models is Multi-Criteria Decision-Making, to borrow the terminology from operations research (Ho, Xu, and Dey 2010; De Almeida et al. 2015), or Multi-Objective Optimization (Marler and Arora 2004; Eddy and Lewis 2002a; Xing, Chen, and Yang 2009; Simpson, Carlsen, and Congdon 2008; Beltrame and Nicolescu 2011; Woodruff, Reed, and Simpson 2013a; Patrick M Reed and Kollat 2008), to

borrow the terminology from engineering design. Regardless of the naming convention, the basic principles remain the same; rather than attempt to create a single optimization objective which yields a single defined solution, the multiple conflicting criteria are simultaneously optimized against each other to create a set of relatively optimal solutions. This set of solutions is characterized by the property that improvement in one objective requires taking a loss in another—in other words, you can't get more for less. This Pareto Frontier, as the set is called, provides the *tradeoff space* where any rational decision maker will choose a Pareto optimal solution from the frontier and in doing so implicitly applies a utility function to trade between the conflicting objectives. The frontier also provides insight into important engineering features such as flexibility or reliability as it provides understanding of related alternatives (Ross and Hastings 2005). Understanding these alternatives also aids in recognizing that the optimization results and formulations are fundamentally correct and can aid in debugging or convincing a client that he results are fundamentally correct and robust (Stumpf et al. 2009).

For this document, we will largely abstract away the optimization process itself and assume we have a reasonably representative sample of the Pareto Frontier. In doing so, we attempt to look at the *understanding* problem. Given an approximate Pareto Frontier can be found, how one generates an understanding of the Pareto Frontier to make more informed design decisions?

1.2.2 Visualization

1.2.2.1 Traditional Approaches

The most common method of visualizing the values of criteria of a set of designs is to plot all the criteria values with each criterion as an axis in a scatterplot. However, visualizing the

tradeoffs of more than 3-5 criteria quickly becomes intractable. More significantly, it gives no information on what decision changes drive performance changes.

A popular method to visualize more than 3-5 criteria is to instead plot the criteria/objective in a parallel axis plot. This allows plotting an unlimited number of criteria. However, these plots quickly become difficult to read and it is not obvious how to order the axes, which can alter which criteria appear to trade and can unintentionally alter the perception of the objective space. As with scatterplots, there is no specialized indication relating the design space changes to objective space changes. (Daskilewicz and German n.d.; Woodruff, Reed, and Simpson 2013b; Stump et al. 2004; Decision Vis n.d.)

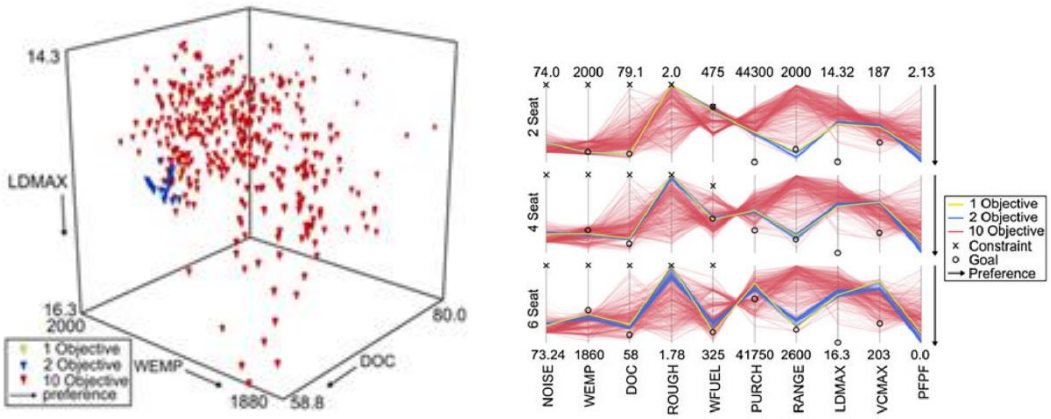


Figure 1: Visualization of a multi-objective general aviation aircraft problem via (left) scatterplot, (right) parallel axis plot (from (Woodruff and Reed, 2013))

Interactive tools have been devised to provide information about both the design and the objective spaces using multiple views. Cloud Visualization (Eddy and Lewis 2002b), the ARL Trade Space Visualizer (Stump et al. 2004), RAVE (Daskilewicz and German n.d.) and commercial software such as DiscoveryDV (Decision Vis n.d.) allow for user interaction between elements of standard plots and allow for linking multiple types of views to aid user understanding of the tradeoffs in the space.

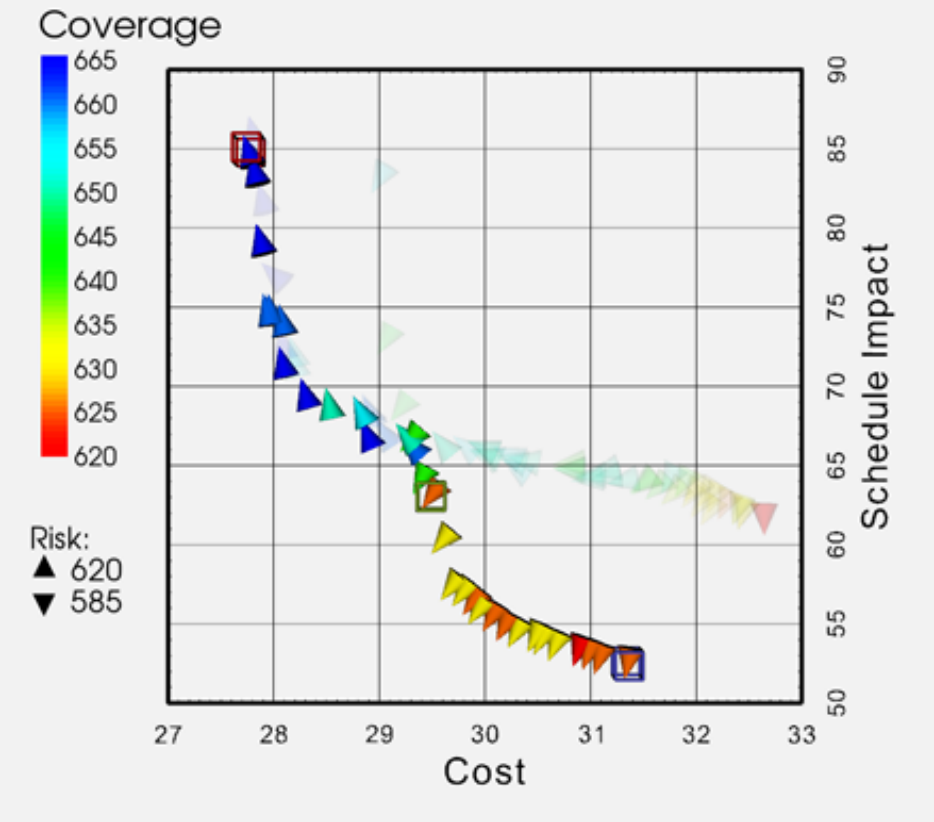


Figure 2: DiscoveryDV (from: DiscoveryDV, 2015)

Kanukolanu et al. use boxes in a scatterplot to represent robust solutions of a simulation (Kanukolanu, Lewis, and Winer 2006). Zhang et al. use boxes of differing size to represent aggregate statistical quantities ((Luke) Zhang et al. 2012). Both (Kanukolanu, Lewis, and Winer 2006; (Luke) Zhang et al. 2012) allow for user interaction in choosing what to plot. These methods place interaction and collating information via multiple standard types of plots as the centerpiece of their approach. This allows for amassing data and aids in user understanding, but requires work to make effective static images representing the decision → criteria map.

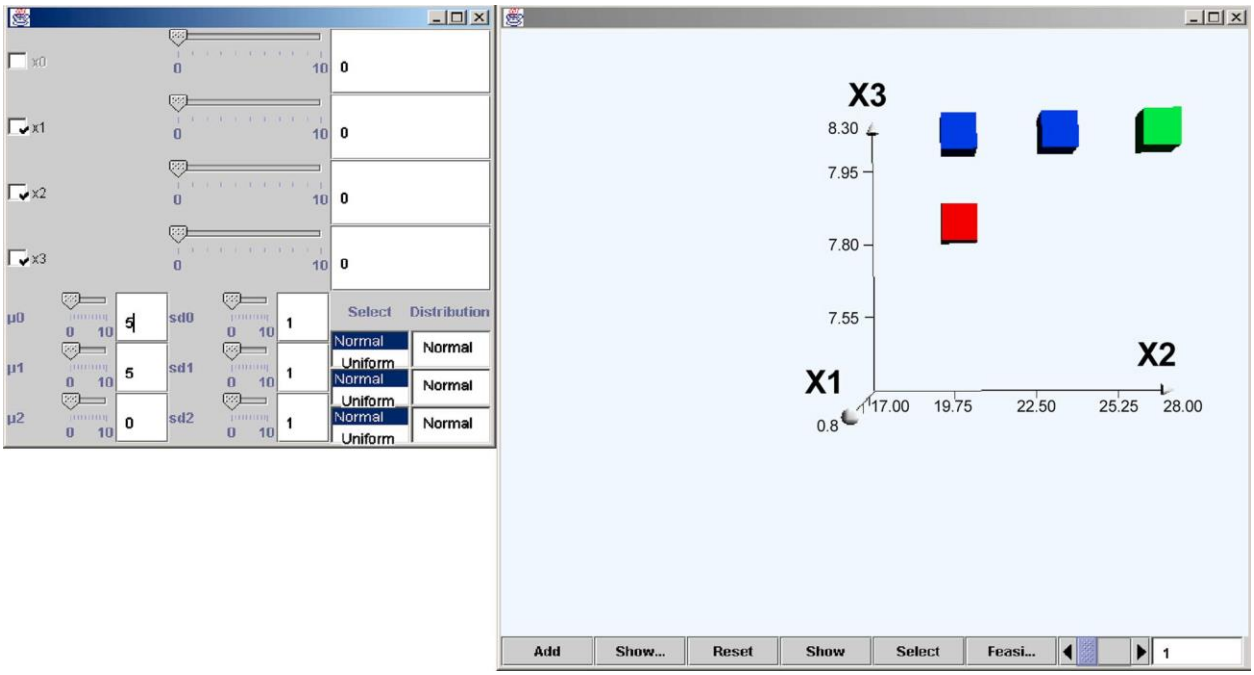


Figure 3: Use of Boxes to Represent Engineering Quantities (from: (Kanukolanu, Lewis, and Winer 2006))

1.2.2.2 Novel Engineering approaches

Visualizations have been developed specifically for tradeoff analysis. Unal et al. developed an index based on crossings of parallel axis plots to visualize tradeoffs between criteria (Unal, Warn, and Simpson 2015). Chiu et al. developed the Hyper-Radial Visualization method, which parameterizes multiple criteria and plots the results on two axis to try and replicate the standard picture of Pareto optimality in 2 dimensions (Chiu et al. 2009). The Hyperspace Pareto Frontier uses a winding path to form a lossless parameterization of every point in the high dimensional space; however, neighborhoods in the original space may not be reflected by neighborhoods in the visualization (Agrawal, Parashar, and Bloebaum 2006). These methods do not take into account the decisions and focus only on finding ways to represent the tradeoffs in 2 dimensions.

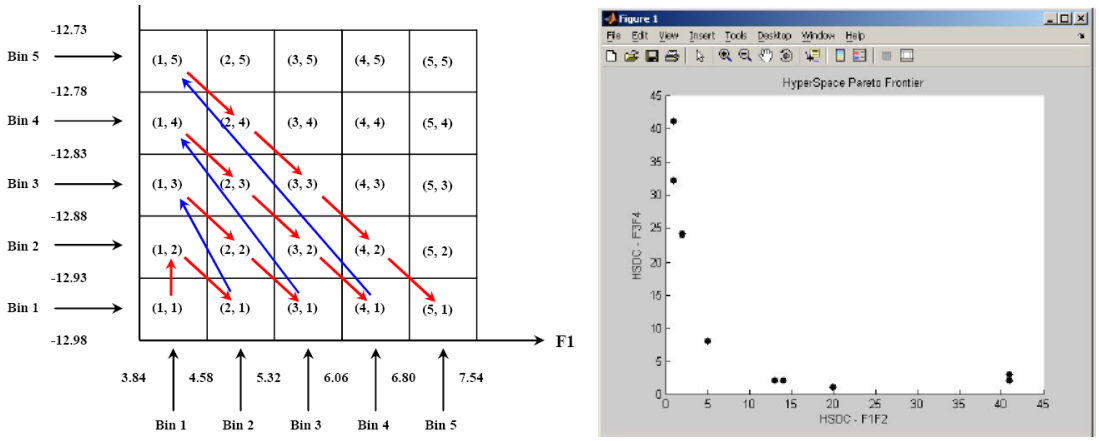


Figure 4: The Hyperspace Pareto Frontier (from: (Agrawal, Parashar, and Bloebaum 2006))

1.2.2.3 Dimension Reduction Approaches

Dimension reduction techniques have been used in visualization to represent high-dimensional objects a low-dimensional space that can be drawn. The Geometric Analysis for Interactive Aid (GAIA) technique in the Preference Ranking Organization METHod for Enrichment Evaluations (PROMETHEE) method considers pairwise preferences between criteria and then classifies the preference of decisions with respect to the criteria with PCA (Mareschal and Brans 1988).

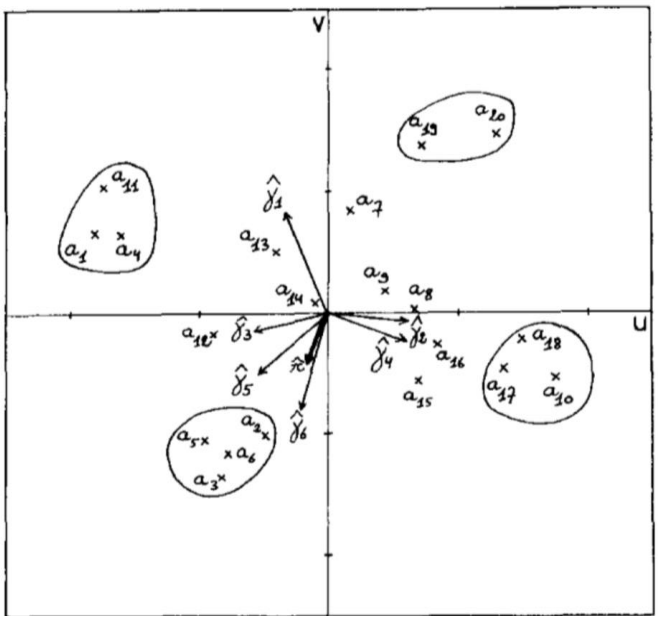


Figure 5: GAIA (from: (Mareschal and Brans 1988))

Richardson and Winer pioneered the use of self-organizing maps to visualize a single objective value (Richardson and Winer 2011). Shimoyaa et al. then used self-organizing maps to present the results of a multi-criteria optimization algorithm by presenting each criterion and corresponding robustness measure as a separate SOM (Shimoyama et al. 2009)

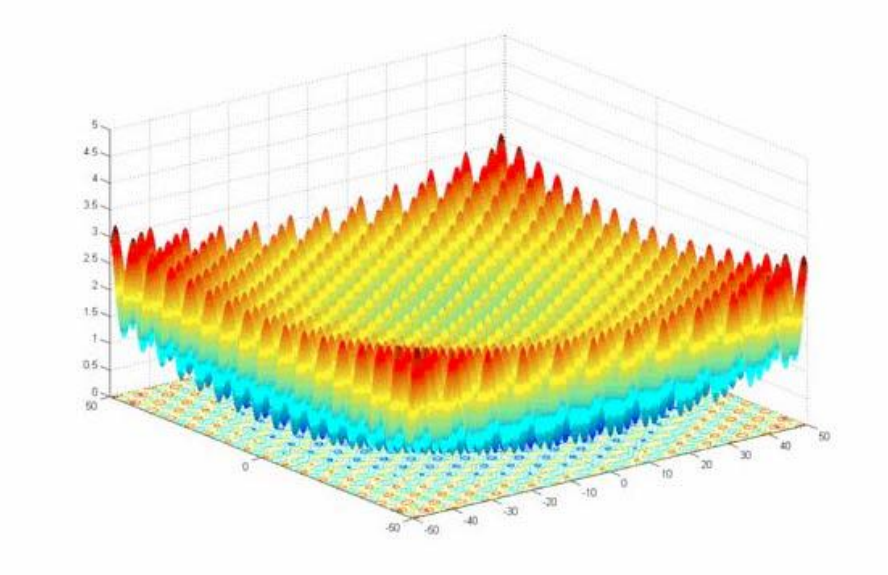


Figure 6: Self-Organizing Maps (from (Richardson and Winer 2011))

1.2.2.4 Virtual Reality for Engineering

The engineering community has primarily considered applying VR for enabling remote engineering or communication for teams (Wendrich et al. 2016) or representing and interacting with CAD models (Pareek, Sharma, and Esfahani 2015). Kuchinski examined how virtual environments could be classified into the information viewed and the mechanisms used to visualize the information, including 3d representations (Kuchinski 2000). More work related to design space analysis is using VR to visualize fluid particle traces in computational mixing simulations (Duncan and Vance 2006). Interactive VR was also used to represent and simulate structural models such as connecting rods (Yeh and Vance 1998). 3d visualization of engineered systems, paired with model based systems engineering is a technique for storytelling-based systems engineering (Madni 2014).

1.2.2.5 Virtual Reality for General Data Analysis

Virtual Reality (VR) has some precedents from the 90s in the Information Retrieval community, primarily for visualizing the relationships between webpages on the early internet. One such technique was the Vineta Virtual Reality system (Young 1996), which placed documents in virtual space and used a radar plot to represent the documents based on document properties. Related methods attempted to reproduce "familiar" spaces such as forests or city-squares (Hoffman and Grinstein 2001; Young 1996) and are based on the idea of Benediktine spaces (Benedikt 1991). These techniques build a 3d world to visualize the space of documents and seek to replicate natural structures. Figures are designed for users to navigate interactively (Young 1996; Hoffman and Grinstein 2001). Nagel, Granum and Musaeus developed a VR system for scatterplots and surface plots for general data visualization (Nagel, Granum, and Musaeus 2001). . One major weakness in these techniques is a need to project high-dimensional objects to low dimensional space. This should be less problematic in a lower-dimensional problem such as systems architecting (~30 decisions) than in larger problems such as classifying documents (~30,000 words).

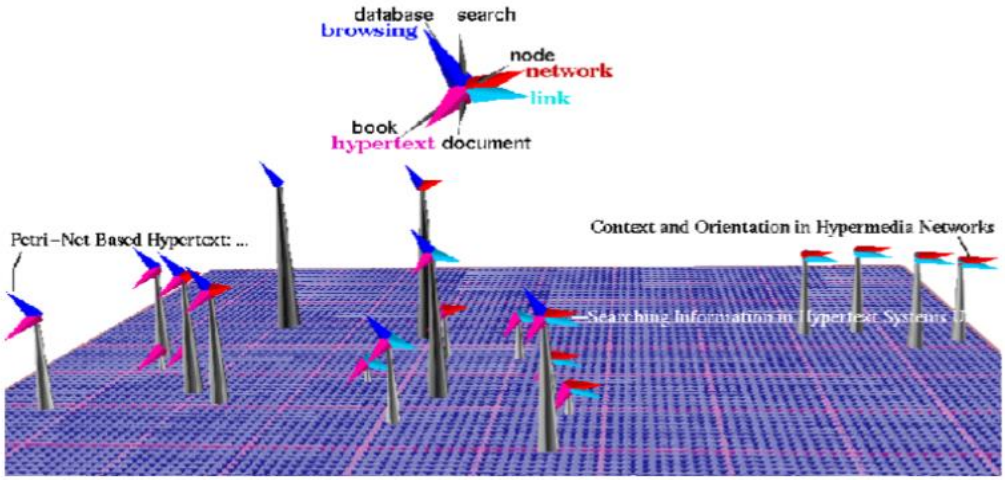


Figure 7: Vineta Virtual Reality System (from: (Young, 1996))

1.2.3 Evaluation of Design Visualization

1.2.3.1 Evaluation Metrics

Evaluations of visualizations for design space exploration are available within the systems engineering and design community. Tusar and Filipic surveyed scatter plots, bubble charts, parallel coordinates radial coordinate visualization, level diagrams, prosecutions (scatterplots, filtered by range) and hyper radial visualization on the DEB4DK dataset to compare the representations of a knee bending towards the Pareto optimal point. On the 4D dataset, only the level sets, prosecutions and hyper radial visualization were able to clearly distinguish the knee point. For evaluation, Tusar and Filipic compared the drawn images and reported what was seen; no other persons were involved in the evaluation of the methods (Tu and Filipi 2016).

Relative evaluation of visualization systems has precedent in the broader data visualization community. Plaisant, Fekete and Grinstein summarize a series of contests for information visualization from 2003-2005 for visualizing relationships and properties of datasets ranging from people to documents to technology companies. Although objective quantitative metrics were attempted, they were not used; human judges reviewed all submissions and scored the submissions on quality of data analysis, appropriateness of the representation and the quality of a written case study (Plaisant, Fekete, and Grinstein 2008). However, we did not find any existing measures that were consistently used for assessing the users' ability to use visualizations to describe and understand design spaces.

1.2.3.2 Experiments in Visualization

Concerning the design of experiments to validate visualization systems, the literature emphasizes the challenges associated with rigorous human research in this area, and in fact

several papers have identified systemic deficiencies in past visualization studies. Seriai, Benomar, Cerat and Sahraoui performed a systematic survey of papers published in IEEE Xplore and Scirus and found only about half the papers used quantitative measures and few used human subjects to evaluate software tools (Seriai et al. 2014). Isenberg, Chen, Sedlmair and Moller surveyed presentations at the IEEE Visualization conference for techniques and methodological weaknesses, specifically they found that the vast majority of papers were quantitative observations of users using tools with small sample sizes and poor experimental controls (Isenberg et al. 2013). They found that most studies were based on qualitative assessment of visualizations and assessment of algorithmic performance (e.g., rendering speed). They also found poor reporting of experimental conditions, and, when multiple subjects were tested, most used 20 or fewer participants. (Isenberg et al. 2013).

1.2.4 Statistical Analysis for Design Space Applications

It has been observed that in many practical cases, when plotted on a scatterplot, the designs fall into distinct clusters due to similarity in the decisions (Kim and Ding 2005) or similarity in objective values (Mareschal and Brans 1988). This is an interesting and useful observation. For example, the clusters can be used to distinguish design families or classes of potential designs. It can also be used to model the degree to which a designer must be forced to commit to key decisions early in the systems engineering design cycle.

From this perspective, our goal of finding important decisions is similar to sensitivity analysis. Factor Prioritization is a common task in sensitivity analysis that seeks a ranking of decisions based on their impact on an objective value. The most common method is Sobol indices, which uses a decomposition of the functional relationship between decisions and criteria to estimate the effect of any single decision or coupling of decisions as the variance contribution

to the overall observed variance of responses. To remain tractable, the number of couplings is fixed in advance and as a results the possible interactions that will be examined becomes fixed (Saltelli et al. 2008; Cariboni et al. 2007; Yin and Chen 2008; Allaire and Willcox 2012; Chen, Jin, and Sudjianto 2004).

One approach in the past to extract information from data is to use classic machine learning techniques. Machine learning techniques to analyze decision effects on objectives have been used in the engineering literature before. For example, Orsborn, Boatwright, and Cagan used Principal Component Analysis (PCA) to extract compound features to characterize classes of vehicles (Orsborn, Boatwright, and Cagan 2008).

Of note, trying to find decisions that give similar or equivalent values in the objective space has been proposed before. De Weck developed the isoperformance methodology for exploring a design problem. In this framework a satisfactory design is found, and then the space is explored to find designs with the same performance but different decisions, cost and risk (de Weck 2001).

One direction from outside the engineering design community that offers promise for making sense of data is the Automatic Statistician (Lloyd et al. 2014; Duvenaud et al. 2013; Grosse et al. 2012), which uses a composition of kernel methods to build a model of a dataset and produce a mixture of text and images to describe it's components. While the Automated Statisician is an inspiration for our analytical approaches, we will focus exclusively on multidimensional Pareto frontiers and engineering design spaces. Additionally, we will attempt to create methods and approaches that are simple and transparent to engineers who may not have a deep understanding of statistical methods.

1.3 Objectives of Research

1.3.1 Formal Notation and Problem Definitions

We will formally define a multi-criteria decision making problem as:

$$\min_{x}(\vec{y} = f(\vec{x}) : g(\vec{x}) \geq 0) \quad [1]$$

The $\vec{y} \in \mathcal{Y} \subset \Re^d$ represent criteria or objectives: these are the qualities of a design that a client of designer would explicitly care about, such as cost, risk metrics, performance metrics (such as ground coverage or weight for satellite systems) or "nice to have" features. $\vec{x} \in X$ represent decisions that define a design (such as the size or placement of a solar panel on a satellite); each unique combination of values for $\vec{x}$ is then a design, though we will often refer to the associated value of $\vec{y}$ as a design. Observe then that designs are unique but corresponding objective values need not be. g represents constraints to satisfy and f represents a black box simulation that produces metrics from a given design.

When optimized, we obtain a Pareto frontier to be notated as

$$P = \{y : \nexists y' \forall i \; y'_i < y_i\} \quad [2]$$

That is, for any design on the Pareto frontier, it is not possible to find another design, also on the Pareto frontier, for which all objectives are improved. For all techniques in this paper, we will assume a representative sample of the Pareto frontier can be found.

All the above notation will be universal, however as different chapters of this document will have different techniques, there will be slightly different assumptions. To keep the assumptions clear we have Table 1. Observe that most assumptions are features of the representation of the design problem. These are often not as strict as may initially appear as almost all engineering systems, particularly in metrics, are represented numerically in modelling and many times a "good enough" function is sufficient for the purposes of the approach.

The assumptions are defined as follows:

1. Minimize All Objectives: all criteria are assumed to be costs to minimize. This is usually a mathematical formality for simplicity as each criterion can be maximized with the transformation $\vec{y} \rightarrow -\vec{y}$ to flip minimization and maximization. Maximize indicates that the criteria are instead maximized (again, without loss).
2. Representative Sample Exists: A sample of designs (decision-objective pairs) can be generated that will implicitly contain all information that the user would be interested in. A general sample is assumed to be drawn randomly from the subspace. A complete sample assumption may be able to use a random sample, but wants a complete sample for best accuracy or provable results.
3. Uses Pareto Frontier: the sample should be drawn from the Pareto frontier
4. Uses Sample Space: the sample should be drawn from the complete tradespace.
5. Distance Function in Criteria Exists: a distance function can be found for the criteria to specify if two designs are "similar" based on the values of the criteria. The function should obey the normal rules of a mathematical metric.
6. Distance Function in Decisions Exists: a distance function can be found for the criteria to specify if two designs are "similar" based on the values of the decisions. The function should obey the normal rules of a mathematical metric.
7. Comparator in Decisions Exists: it is possible to take two possible designs and extract the changes between the designs and compile changes in a meaningful way. This can be as simple as looking at the designs and accumulating the changes in a list for counting.

8. Pareto Surface in Criteria Exists in Real Subspace: Criteria can be embedded in a real space. Effectively, categorical variables and satisfaction values can be represented accurately as numbers and the resulting embedding will be an reasonably accurate representation of the actual practical application of the criteria.
9. Non-Degenerate Pareto Frontier: There are no criteria which fail to trade or generate only one solution set for that criteria. This is largely done without loss. For fully degenerate Pareto Frontiers (single points) analysis is unnecessary as there is only one optimal design to consider (i.e. the engineering "task" is largely complete). For more general degenerate frontiers, it is frequently possible to create a non-degenerate Pareto Frontier from by removing or combining criteria that fail to trade or fail to constrain the design space.
10. Positive Orthant: all criteria values are positive over the entire sample. This is more a mathematical simplification as the finite sample can always be translated and as a practical matter, the useful range of the true Pareto Frontier is usually bounded anyway. This assumption will, however, simplify the math by acting as a lower bound on the allowable performance.

Table 1: Assumptions Table

Assumption	Chapter 2 Cityplot	Chapter 3 MOMS-CFCs	Chapter 4 MAPSA	Made without Loss?
Minimize All Objectives	Maximize	Yes	Yes	Yes
Representative Sample Exists	Yes	Complete	Yes	No
Uses Pareto Frontier	Yes*	No	Yes	No
Uses Sample of Space	Yes*	Yes	No	No
Distance Function in Criteria Exists	No	Yes	No	No
Distance Function in Decisions Exists	Yes	No	No	No
Comparator in Decisions Exists	No	Yes	No	No
Pareto Surface in Criteria Exists in Real Subspace	No	No	Yes	No
Non-degenerate Pareto Frontier	Yes	Unneeded	Yes	See below
Postive Orthant $\vec{y} > 0$	No	No	Yes	Yes

*** Can be swapped to generate vastly different kinds of information. See chapter 2.**

1.3.2 Aim to Accomplish

As the purpose of this research is to find our way in design space, we identify the following areas that we would like to learn:

1. Is it possible to describe and navigate a very high dimensional design spaces?

   More concretely,

   a. Can we visualize and understand design spaces with more objectives than the normal 2-4 that are given with scatterplots?

   b. Can we describe tradeoff rates on the Pareto front visually or numerically in a compact way that is illuminative? E.g. for understanding the sacrifices being made?

    c. Can we describe the overall shape and structure of the Pareto front in a few key numbers or graphs?

2. Is it possible to answer the following using effective visualization of the decision→objective mapping?
    a. Does the design space has “regions” of performance?
    b. Do decisions relate to criteria in key ways?
    c. Are there natural design families?
    d. Are some decisions more important than others?
3. Are our visualizations easier to read and interpret in confirmation or denial of the previous questions?
4. Can we provide insight to underlying reasons for clustering behavior observed in visualization?
5. Can we formulate nonparametric methods exploiting the comparisons of designs in the trade-space to understand the sensitivity and key decisions based on apparent clustering behavior?
6. Can decomposition of the Pareto frontier provide insights into the behavior in very high, difficult-to-visualize dimensions? Can such a model then be further utilized in a decision making process?

As is normal for research projects, we wish to answer all these questions in the affirmative by demonstration. In doing so we hope to advance the state of the art by developing visualizations that are better able to represent the high-dimensional nature of the tradespace and do so in a way that respects the constraints and requirements of engineering design (e.g. analysis is reliable, multi-dimensional, relatively few input factors or highly geometric, small-sized or

model-generated samples, relatively clean underlying mathematical systems). From the standpoint of engineering design, this represent novel solutions to problems in multi-objective engineering practice for generating understanding of the design challenges. From the standpoint of more general data analysis, this represents a specialization to an economically important field and the development of techniques in a space where failure understanding is often considered a requirement.

One key area which we will attempt to develop is the use of new visualizations for use in high-dimensional tradespace exploration problems. Currently, standard methods are either limited in the number of objectives that can be represented, rapidly become too busy to see the features of the tradespace or attempt to place the design space in a reduced space which is either lossy or otherwise heavily distorts the relationships between designs. We develop the state of the art by opening up the possibility to represent much higher dimensional tradespaces, respect the decision-objective tradeoff and provide means to control for the losses. Cityplot, in particular, will also create a new application to virtual spaces to the multi-objective engineering analysis that allows intuitive use of the third dimension and the representation of the tradespace at different "scales." MAPSA will introduce some visualization designed to summarize the shape of the Pareto surface due to a new approach to understanding the shape of the Pareto frontier and controls loss by providing bounds on the representation errors.

Table 2 compares the advantages of different common visualization techniques (plus ours). Of interest are:

1. Max Criteria: the maximum number of criteria that can be naturally represented by the visualization

2. Lossy: if there is dimension reduction or similar techniques, is information is lost in the compression?
3. Naturally includes Decisions and Objectives: Does the plot provide natural ways to include both decision information and criteria results for understanding the mapping between engineering decisions and the resultant criteria values?
4. Natural User Interaction: Does the plot have natural ways for users to interact with the plot to get more information?
5. Naturally captures different “scales”: Does the plot tend to reveal only large-scale global information or does it allow for users to see both local and global effects easily?

Table 2: Advantages of Techniques Compared, Visualization

	Max Criteria	Lossy	Naturally Includes Decisions and Objectives	Natural User Interaction	Naturally captures different "scales"	Other notes
Scatterplots	2 or 3, +2-4 with unusual glyphs or coloring	No	Only if sacrifice objective as a visualization axis.	Yes	No	Dramatically harder to use with more criteria
Parallel Axis Plots	unlimited	Sensitive to axis order	As objectives only.	Yes	No	Busy, hard to use
Lossless compression methods (hyper- methods)	unlimited	No, but have to box objectives	No.	No	No	Complex space distortion
SOM	unlimited	Highly	No.	No	No	
PROMETHEE	unlimited	Highly	No	No	No	Requires criteria preferences
Cityplot	**unlimited**	**In Decisions**	**Yes**	**Yes**	**Yes**	**Requires distance in decisions**
MAPSA	**Unlimited**	**Complicated**	**No**	**No**	**Complicated**	**Based on representing key numbers**

Another area we will attempt to innovate in is the use of analytics to find and describe useful features of the design space. Currently, the state of the art is mostly aimed at general purpose applications. This leads to linearity in the model, engineers forced to make statistical modelling choices or great complexity that makes the model less useful in applications where understanding is the goal. We aim to demonstrate how simple modelling can be used to build specialized tools that can provide very useful information based on the analytic features of the methods chosen. MOMS-CFCs provides insight to the classic question of "why do my designs look grouped" by posing the problem as one of sensitivity and shows that this can be used to mine decision commitment (the degree to which a decision "locks in" a level of performance by constraining available options) in the conceptual engineering analysis. MAPSA shows how generalized linear models can build nonlinear models that provide summary of the overall shape of high dimensional Pareto frontiers—essentially shifting the understanding task to one of compressing it to a few key parameters.

Table 3 compares the effectiveness of different analysis methods. Specifically, it looks at the areas:

1. Linear: Does the technique assume a linear model
2. Finds Global Sensitivities: Can the method provide natural descriptions of the global sensitivity of criteria values from decisions?
3. Finds Local Sensitivities: Can the method provide natural descriptions of the small scale sensitivity around a key design or neighborhood?
4. Interpretable: Does the method provide natural explanations of its results either by being mathematically simple enough to be understood by the sophisticated layperson or providing plaintext or visual explanations of the results?

5. Specialized for MCDM applications: Is the method intended to represent and understand Pareto-frontiers and find the information used in engineering applications?

Table 3: Advantages of Techniques Compared, Analytics

	Linear	Finds Global Sensitivities	Finds Local Sensitivities	Interpretable	Specialized for MCDM applications	Other Notes
Linear Models (ANOVA, linear regression, etc)	Yes, or Generalized with specialized basis	Yes	No	Yes	No	Traditional. Effective.
Neural Networks	No	No	Sometimes	Sometimes	No	Can require significant data intake
Automated Statistician	No	No	No	Yes	No	Far more general. Not really intended for multidimensional applications
Tradeoff Index and Pareto Shape Index	Tradeoff Index, Yes. Pareto Shape Index, No.	Yes	Yes	Yes	Yes	Like MAPSA, except only goes up to 2^{nd} order
MOMS-CFCS	**No**	**Can**	**Yes**	**Yes**	**Yes**	**Highly nonlinear. Discovers subsystem coupling.**
MAPSA	**Generalized, with naturally chosen basis**	**Yes**	**Approximate**	**Yes**	**Yes**	**Surrogate model for Pareto Applications. Designed to succinctly represent the space**

1.3.3 Roadmap to a Solution and this Document.

In this chapter we have provided motivation, background knowledge and shown where weaknesses lie in the state of the art.

In Chapter 2, we present Cityplot. Cityplot is a novel VR-based visualization technique that is able to represent both the decisions and objectives simultaneously. It is also capable of scaling to large numbers of criteria and naturally represents and investigates design families. We draw Cityplot for a number of different real and simulated engineering problems and discuss how to find appropriate distance functions. Finally, we design and present an experimental study of Cityplot to show that it leads to improved performance and understanding of the design space compared to traditional approaches for design visualization.

In Chapter 3 we present a novel interpretation of the "clustering" behavior engineering design researchers notice when visualizing random samples of engineering design metrics. We then show that this can be related to the decisions in a manner analogous to sensitivity analysis that organically discovers highly coupled decisions that drive significant changes in performance. Finally, we discuss some significant issues to overcome before the technique becomes a practical method to discover decision couplings in real-world problems.

In Chapter 4 we provide a new technique and interpretation for numerically understanding the shape of Pareto Frontiers for engineering design inspired by compression. Following this method, we attempt to generalize Unal's shape index (Unal, Warn and Simpson, 2016) to describe the shape of the Pareto Frontier with a coordinate transform and surrogate modelling.

Finally, in Chapter 5 we conclude by revisiting the contributions of this document and discussing the limitations and contributions to the state of the art. We also describe potential avenues for further development and research.

Note: much of the contents of Chapter 2 have been published as follows:

Nathan Knerr, Daniel Selva. Cityplot: Visualization of High-Dimensional Design Spaces with Multiple Criteria. Journal of Mechanical Design. 138(9). 21 July 2016

Nathan Knerr, Nathaniell Kinzley, Daniel Selva. Evaluating A 3d Visualization For Understanding Design Space Structure. Systems Engineering [in submission]

Much of the contents of Chapter 3 have been published as follows:

Nathan Knerr, Daniel Selva. Mining Multi-Objective Minimal Commitment Decision Significance via Cluster-and-Find-Changes. 58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. Grapevine, Texas. 9-13 Jan. 2017

Much of the contents of Chapter 4 are to be published

Nathan Knerr Place And Project: A Method To Approximate And Summarize Pareto Frontiers. Structural and Multidisciplinary Design Optimization [in preparation]

2 CITYPLOT: A VIRTUAL REALITY VISUALIZATION OF THE DESIGN SPACE

Cityplot is a novel visualization technique that extends virtual reality to the representation of the abstract design space. This aids in interaction and allows the representation of more objectives and information in a single cohesive environment.

The goals of Cityplot are to: (1) assist in seeing how design decisions affect design criteria; (2) allow for understanding the tradeoffs between conflicting criteria; (3) be capable of being applied to both continuous and combinatorial design spaces in high dimension. Notably, when it comes to understanding and seeing the design space, Cityplot will aid in understanding design families, understanding the relationships between large numbers of criteria and understanding how sensitive criteria are to decision changes. Beyond presenting Cityplot, the primary contribution of this chapter is to demonstrate:

1. It is possible to find distance functions in many practical situations that have natural interpretations, such as number of components included
2. One can visualize many criteria and find structure about tradeoffs and relationships to decisions which achieve those tradeoffs
3. Modifications in the distance function or in the set to visualize affect the resulting plot and what the results mean
4. Cityplot can be used to gain insight on real-world problems
5. Humans are able to easily learn and utilize Cityplot for design space exploration tasks.

It should be noted that a technique known as Cityscapes has a similar name to our technique (Cityplot), but refers to a completely different technique, with more resemblance to a mesh plot built out of bar graphs than any of the techniques described here (Walker et al. 1993).

Regardless, it often appears in the 90s visualization literature (Hoffman and Grinstein 2001; Young 1996) and can be a source of confusion.

2.1 On Distance Functions

A key requirement of our method will be that a user-supplied distance function is available. Similarity measures and distance functions are commonly used for applications ranging from machine learning to spell checking. Section 14.3 of (Hastie, Tibshirani, and Friedman 2009) provides a brief overview of some typical ways to describe object similarities in machine learning and the relationship to kernel methods (Hastie, Tibshirani, and Friedman 2009). A discussion of building empirical similarity matrices (sufficient for our method) by asking humans to rank pairs of designs by similarity or doing pairwise comparisons is available in (Borg and Groenen 2005). Of interest for discrete design problems is the edit distance of graphs. Among the early papers to give the idea of a graph edit distance is (Sanfeliu and Fu 1983) which finds edit distance for a large number of operations when the graphs are attributed by a graph grammar.

Distance functions have been considered in the engineering design literature. McAdams and Wood created a similarity metric for design-by-analogy by considering weighted customer needs and relating these to functional flows. The similarity is then the inner product of the related functional flows weighted by customer needs (McAdams and Wood 2002). These ideas are extended and reapplied to find similar functional requirements over multiple products in (McAdams, Stone, and Wood 1999).

2.2 Nomenclature/Setting

We perform a slight modification of the formulation in section 1.3.1:

$$\max_{\vec{x} \in X} \vec{f}(\vec{x}) \qquad [3]$$

That is, we require our criteria to be all maximized. This is a slight improvement to intuition as it will make "better" criteria values correspond to taller buildings (the building metaphor is defined in section 2.4.1). Reversing this convention will make "better" criteria values correspond to shorter buildings. The advantage of maximization is an intuitive interpretation; the advantage of minimization is less visual crowding

As noted in 1.3.1, we assume (cf. section 2.3) there is a *distance function* $m: X \times X \to \Re$ (where × means Cartesian product) that describes how dissimilar two designs are and obeys the usual definition of a metric: a nonnegative, symmetric function that satisfies the triangle inequality and is zero if and only if the designs are identical.

A *Cityplot* is a graphical representation created by using dimension reduction and bar graphs (cf. section 2.4) to approximately represent the relationships between designs $\vec{x}$ with respect to the distance function *m* and the objective function $\vec{f}$ (cf. Figure 8).

We assume we are able to obtain a *sample* of known designs and corresponding criteria values $S = \{(\vec{x_i}, \vec{y_i})_{i=1}^{N}\}$ for an interesting subset of the design-objective space. For the purposes of drawing the Cityplot on the generated sample, it is possible for the objective functions to be fully black box.

2.3 Problem Types and Distance Functions

The Cityplot is designed to handle both continuous and combinatorial design spaces which appear often in the early-phase design of complex engineered systems. This section describes five simple, typical problems in early-phase design and some distance functions that can be used on each (Selva, Cameron, and Crowley 2017). Section 2.7 has actual problem instances for each type.

2.3.1 Down-Selection

In a down-selection problem, the space of feasible designs can be represented as $X = \{0,1\}^n$. That is, there are *n* yes/no decisions represented by 1 or 0 respectively. Examples of down selection problems in systems architecting are choosing a subset of functions that the system must perform, or a set of components that the system will comprise (Selva, Cameron, and Crowley 2017). In such a case, we propose simply using the *Hamming distance* (the number of changed characters in a fixed length string to transform one string to another) as the distance function:

$$m(\overrightarrow{x_1}, \overrightarrow{x_2}) = \sum_{i=1}^{n} |(\overrightarrow{x_1})_i - (\overrightarrow{x_2})_i| \quad [4]$$

This distance has the intuition of the number of decisions that must be changed to make the two designs the same. When using 0 and 1 as elements of $\vec{x}$ this is equivalent to the usual L_1 or L_2 distance.

2.3.2 Assignment

In an assignment problem, a design consists of assigning *n* objects (e.g. workers) to *b* bins (e.g., machines). An object can be assigned to any number of bins and a bin can have zero or multiple objects. The assigning problem appears in system architecture when mapping functions to components, among other applications (Crawley, Cameron, and Selva 2016). A solution to an assigning problem can be represented as a matrix $A \in \{0,1\}^{n \times b}$ indicating if an object is assigned to a corresponding bin (that is, $A_{ij} = 1$ iff object i is assigned to bin j). We thus use the following distance function:

$$m(\vec{x}_1, \vec{x}_2) = \sum_i \sum_j \left| \left(A^{\vec{x}_1}\right)_{ij} - \left(A^{\vec{x}_2}\right)_{ij} \right| \quad [5]$$

where the superscript $A^{\vec{x}_1}$ means "the assignment matrix for design $\vec{x}_1$." This distance is consistent with treating assignment as down-selecting where we down-select on (object, bin) pairs and using the reasoning in section 2.3.1

2.3.3 Partitioning

In a partitioning problem, each design, $\vec{x}$, is a partitioning of *n* objects—a grouping of *n* objects into discrete subsets such that each object is in exactly one subset. Partitioning is distinct from and more constraining than assigning in the sense that the bin numbers are irrelevant and all objects are each assigned to exactly one bin. Partitioning problems appear in system architecture when choosing a system decomposition, among other applications. Since it is more intuitive to imagine moving objects across bins, we propose using the transfer distance, which is the number of objects that must be moved to make one partition identical to another. Known as the transfer distance, an $O(s^3)$ matching algorithm, where *s* is the sum of the number of subsets in each partition, exists to calculate the distance efficiently (Gusfield 2002). This algorithm observes that the number of decisions is fixed and hence:

$$m(\vec{x}_1, \vec{x}_2) = n - \max_{\theta} \left(\sum_i \left| S_i^{\vec{x}_1} \cap S_{\theta(i)}^{\vec{x}_2} \right| \right) \quad [6]$$

where θ is a mapping that pairs subsets of $\vec{x}_1$ with subsets of $\vec{x}_2$ ($S_i^{\vec{x}_1}$ and denoted $S_{\theta(i)}^{\vec{x}_2}$ respectively) and $|\cdot|$ means set cardinality. When $(\vec{x}_1, \vec{x}_2)$ are fixed, it is simply necessary to calculate all intersections of subsets and then pick the optimal matching, which can be done with the Hungarian Algorithm (Denœud 2008; Gusfield 2002).

2.3.4 Decision-Option

In decision-option problems, better known as catalog design, a design consists of a set of components, each of which is taken from options of a catalog. If we give the elements of the catalog for each decision indices $[1, c_i] \subset \mathrm{N}$ then the decision option is given by $\mathrm{X} = [1, c_1] \times [1, c_2] \times \ldots \times [1, c_n]$ where we have *n* decisions to make. Decision option problems usually arise when choosing components of different types or from competing vendors to fulfill defined roles in a system (Selva, Cameron, and Crowley 2017). Again, we use the Hamming distance

$$m(\overrightarrow{x_1}, \overrightarrow{x_2}) = \sum_{i=1}^{n} \mathrm{I}((\overrightarrow{x_1})_i \neq (\overrightarrow{x_2})_i) \quad [7]$$

where $I(E) = \begin{cases} 1 \text{ is E is true} \\ 0 \text{ if E is false} \end{cases}$ is the indicator function.

2.3.5 Continuous Design Spaces

In addition to the discrete decision problems described above, many design problems involve continuous variables. For our purposes, we will assume that any continuous design decisions have been normalized and thus are non-dimensional and similar scale. Hence, we take the classic Euclidean distance as a measure of dissimilarity. This is frequently the case where the operators that define performance are assumed to lie on the usual L_2 Hilbert space of functions. The Euclidean distance is fast, simple and has strong mathematical properties that make it easy to compute and optimize.

2.3.6 Distance Themes and Summary

The distances we provide for down-selection, assignment and partitioning problems are summarized in Table 4. These chosen "default" distances are all edit distances—the number of decisions that must be changed to transform one design into another. In continuous design problems, there is not an immediately natural analog for "number of changes" because there is not a discrete "next" level for a given design to take. Instead of counting the number of changed decisions, the distance will give a radius of a ball that describes sets of decision changes of a similar scale. The Euclidean ball is then the most familiar such ball. If only one decision is changed at a time when creating new designs, the Euclidean distance between each pair in the sequence will recover the size of each change.

When combining individual distance functions to form composite distance functions (e.g. as in section 2.7.4), it is likely that some decisions will be much more significant than other decisions. These decisions should be given higher weight in the distance function. If the value of

a given decision disallows values for other decisions, the former decision should also be more significant than the decisions it disallows.

Problem	Domain	"Default" Distance
Down-select	$\{0,1\}^n$	$\Sigma_{i=1}^{n}\lvert(\overrightarrow{x_1})_{\mathrm{i}} - (\overrightarrow{x_2})_{\mathrm{i}}\rvert$
Assignment	$\{0,1\}^{n\times b}$	$\sum_{\mathrm{i}}\sum_{\mathrm{j}}\left\lvert\left(\mathrm{A}^{\vec{x}_1}\right)_{\mathrm{ij}} - \left(\mathrm{A}^{\vec{x}_2}\right)_{\mathrm{ij}}\right\rvert$
Partitioning	Partitions of a set of size n	$n - \max_{\theta}\left(\sum_{i}\left\lvert S_i^{\vec{x}_1} \cap S_{\theta(i)}^{\vec{x}_2}\right\rvert\right)$
Decision-Option	$\Pi_i^n[1, c_i]$	$\Sigma_i I((\overrightarrow{x_1})_i \neq (\overrightarrow{x_2})_i)$
Continuous	$X \subset \Re^n$	$\sqrt{\Sigma_{\mathrm{i}=1}^{\mathrm{n}}((\mathrm{x}_1)_{\mathrm{i}} - (\mathrm{x}_2)_{\mathrm{i}})^2}$

Table 4: Summary of Problems Presented

2.4 Cityplot Drawing/Technique

Drawing the visualization consists of the four steps below. An example plot is Figure 8

1. Perform Multidimensional Scaling (abbr. MDS, see section 2.4.2) on the decisions of a chosen set of designs, usually the Pareto frontier, to 'flatten' it to $\Re^2$ (city locations)
2. For each design, plot the criteria values as bar graphs in the 3rd dimension (skyscrapers)
3. Draw lines to connect close designs and give an indication of their distances (roads). Add a legend to give numeric interpretation to roads (e.g., red means distance =1 in design space *X*).
4. Use the mouse interrupts to show text information about a design of interest when clicked. This is left out of static images as it only shows one design at a time.

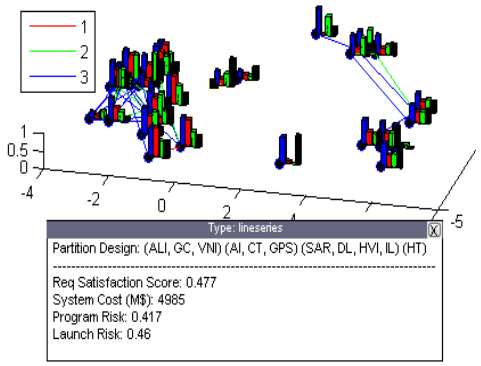


Figure 8: Example plot. The *data cursor* window (text box) displays the design and the criteria values of the selected design (black square). The legend labels road distances.

Reading the plot is intuitive due to an analogy with buildings, cities, and roads. *Cities* represent designs. *Roads* connect cities and the color of a road (which is labelled in the legend) gives the distance in the original design space. The heights of *skyscrapers* in each city give the values of each criteria for that design (each skyscraper is a separate criterion for a given design): the taller the building, the better the value of that criterion. The color of a skyscraper represents which criterion the given skyscraper represents; the colors are labelled elsewhere.

When reading a Cityplot, the following facts are important to remember: (1) Multidimensional Scaling considers all distances regardless of whether a road is drawn; roads are purely for visual interpretation and help combat distortion resulting from the dimension reduction. (2) **The x and y axis of the Cityplot result from the MDS output and are meaningless**—the absolute position of a city in the plot is irrelevant; instead, **the relative positions (Euclidean distances between designs) in the XY plane** are important and approximate the distances in the original design space.

2.4.1 Skyscraper Normalization and Visibility

The value of each objective is normalized separately so that the drawing space can be better utilized to show the changes in performance. This is similar to normalizing axes in a

parallel axis plot; the normalization gives more visual range to show changes and makes changes in different criteria more visually comparable. The normalization used was:

$$B_i^j = \frac{\overrightarrow{f(\vec{x}^J)}_i - \min_j \overrightarrow{f(\vec{x}^J)}_i}{\max_j \overrightarrow{f(\vec{x}^J)}_i - \min_j \overrightarrow{f(\vec{x}^J)}_i} \quad [8]$$

Where *j* is an index over the designs to visualize and *i* is the i^{th} objective. The *z* axis of the Cityplot is the value of each B_i^j.

3d rotation features can be used to get a good view of the design space as represented in the Cityplot. Usually, the best angle is isometric to limit the occultation of points by buildings and allow for viewing the most data; the exact optimal settings vary with the exact layout of each Cityplot. Zoom and pan tools provide a way to look at regions of interest.

2.4.2 City Placement with Multidimensional Scaling

Classical *multidimensional scaling* (MDS) attempts to map distances between elements in a dataset to a lower dimensional space such that the inter-point distances are well represented in a 2-d plane with the Euclidean distance. The Euclidean distance is preferred in the plane as it is the most natural to the human eye (Borg and Groenen 2005).

Classical MDS consists of the following minimization:

$$\underset{z_i}{argmin} \sum_{i<j} \left| \left| \vec{z_i} - \vec{z_j} \right|_2 - m(\vec{x_i}, \vec{x_j}) \right|^2 \quad [9]$$

where $\vec{z_i} \in \Re^2$ is the position of the i[th] design in the reduced (drawn) space, $\vec{x_i}$ is the position of the i[th] design in the original design space and $\left| \vec{z_i} - \vec{z_j} \right|_2$ is the Euclidean distance between the designs i and j in reduced space (Borg and Groenen 2005). Intuitively, this objective states that we seek to minimize the average difference between the distances in the 2d onscreen representation and the original distances in the design space. Unfortunately, it is impossible to guarantee the method will always reduce the dimension cleanly, but one purpose of this chapter is to show that this can still be useful in many cases (Borg and Groenen 2005).

As pointed out in (Hastie, Tibshirani, and Friedman 2009), classical MDS is equivalent to PCA when the set to visualize is centered about $\vec{x} = \vec{0}$ and the distance metric *m* is Euclidean. However, MDS is capable of representing a wider range of distances, including non-Euclidean distance metrics and can perform nonlinear mappings.

Alternative MDS weighting schemes are available (Borg and Groenen 2005). *Sammon mapping* normalizes the distances as per:

$$\underset{z_i}{argmin} \frac{1}{\sum_{i<j} m(\vec{x_i},\vec{x_j})} \sum_{i<j} \frac{\left|\left|\vec{z_i}-\vec{z_j}\right|_2 - m(\vec{x_i},\vec{x_j})\right|^2}{m(\vec{x_i},\vec{x_j})} \quad [10]$$

This normalization has the effect of amplifying small distances in the design space and is useful when differences in scale in distance values became too large to be cleanly represented or would result in colocation of designs (Sammon 1969).

For Section 2.7, a MATLAB version of Cityplot was used and MATLAB's *cmdscale* (classical MDS) (The Mathworks n.d.) and *mdscale* (non-classical MDS, including Sammon mapping) (The Mathworks n.d.) were used. For section the experiment described in sections 2.8 through 2.13, a Unity-based virtual reality implementation was used and the scikit-learn MDS method was integrated into the VR system (Scikit-learn Developers n.d.).

2.5 Application Methodology And Considerations

The proposed methodology to apply this technique is as follows:

1. Choose a distance function *m* in the design space
2. Choose a set of designs to visualize
3. Draw the Cityplot (see section 2.4)
4. Examine the Cityplot (see section 2.6)

The method is interactive and iterative: after step 4, users can use the data cursor to get information about specific designs and take representatives of regions of the Cityplot. The

Cityplot can be rotated or rescaled to give a better view. Queries can be answered by defining appropriate functions and redrawing the plot with these functions as additional criteria. It may also emerge that the distance function is flawed or the plotted set is uninteresting, in which case the user may revisit steps 1 and 2 and redraw the plot.

To elaborate on step 1, the distance function is encoding what it means for two designs to be alike or different. While the most natural distance functions will emerge from the nature of the problem itself, we provided some default distance functions for different types of problems in section 2.3 and Table 4. The presented distances from the previous section should be sufficient in many cases.

The selection of the set to visualize is tightly related to the original multi-criteria design formulation we wish to develop for. Visualizing the entire design space will usually not be feasible nor desirable due to the very large number of designs that would need to be evaluated, organized and visualized. For this reason, we recommend first isolating the Pareto frontier

In multi-criteria decision-making, the Pareto frontier is the set of all "rational" options. Picking a design amongst the frontier represents trading between criteria, which requires expressing subjective preferences. This set (or an approximation of it) can be found via meta-heuristic algorithms as in (Selva 2012; Woodruff, Reed, and Simpson 2013b) or classical multi-criteria decision making techniques such as those described in (Marler and Arora 2004). Unless otherwise noted, all Cityplots in this chapter visualize the Pareto set of their respective problems for reasons discussed in section 2.6.2.

Alternatively, it is possible to take a random subset of the design space; this will yield approximate global information about the design space and objective functions (see section 2.6.2).

Other problem-dependent sets of interest can also be visualized, but are outside the scope of this chapter.

2.6 Features To Extract From Cityplot

By nature, the Cityplot plots similar decisions together and provides a natural way to organize the criteria values to be queried by a user. This means that with the data cursor, a user can find decisions that tend to lead to criteria values. However, the strength of Cityplot lies in its ability to extract global information.

2.6.1 Smoothness, Design Families and Tradeoffs

The Cityplot can inspect how sensitive designs performance is to small changes in design features. We define $(\vec{f}, X)$ to be *smooth* if the distance between any two designs, $m(\overrightarrow{x_1}, \overrightarrow{x_2})$ is small implies $\left|\overrightarrow{f(x_1)}_i - \overrightarrow{f(x_2)}_i\right|$ is also small for each criterion *i*. Intuitively, if the distance function *m* represents the changing of decisions then the criteria values usually will not change dramatically around an isolated design. **This definition does not require the decision or objective spaces to be continuous or discrete**.

If the space is smooth with respect to the distance function then smoothness should be reflected in the images as a *slow transition* in the values of the criteria across the image when the Cityplot is drawn. This also makes it easier to find the decisions that drive criteria values as it bounds the changes one would expect in criteria around a design that can be extracted from the data cursor. Knowing whether a given design space is smooth is useful. For example, smoothness indicates robustness to changes in the design. As an example of such transitions, see Figure 11 or Figure 15.

Smoothness is not a requirement to apply Cityplot; in fact, Cityplot can be used to assess the smoothness in a region of the design space. If the drawn Cityplot has sharp

transitions in the drawn criteria, it demonstrates that the problem is sensitive to “small” changes in designs. It is possible to be smooth over most of the domain and still have unsmooth regions. As example of non-smoothness, see the discussion of Figure 13.

When plotting the Cityplot, clusters can become visible as closely located cities in the xy plane. These communities represent *design families*, which are sets of similar designs. Such families emerge naturally from the application of MDS to the visualized set. Because criteria are a function of design variables, when the function is smooth, designs within the family will have similar criteria values. In such cases, a single design can be used as a representative of the family instead of many designs to reduce cognitive load. For examples of such families, see Figure 10, Figure 11 or Figure 15.

As the Cityplot includes skyscrapers representing the objective function values, tradeoffs and objective value relationships between design families and individual designs become quickly apparent. For example, regions changing color due to the heights of given skyscrapers will represent tradeoffs between objectives.

2.6.2 Interactions of Distance and Set Choices

The features just discussed are dependent on the choice of set. Clearly, the distance function is also important as it defines the notions of smoothness and design families.

Particular attention should be paid when using the Pareto frontier as the set to visualize. Because inclusion in the set is dependent on the objective function values, clusters represent **design families among the optimal designs** and objective function values are included in the creation of design families indirectly. Additionally, tradeoffs indicated in the Cityplot will be only amongst the Pareto optimal designs, which coincide with the usual interpretation of

tradeoffs being defined between designs in the Pareto optimal set. Notice that it is possible for the Pareto frontier to be smooth while the rest of the design space is not (because $P \subseteq X$).

Alternatively, when choosing designs at random, clustering should be ignored in most cases, especially when the sample size is very small compared with the size of the design space. However, tradeoffs represent relationships of the criteria in the more general design space. Choosing a random subset allows for a visual query (and possible disproof) of the smoothness of the general space and gives a designer insight into overall behaviors of the objective functions.

2.7 Examples

2.7.1 Toy Example 1: Effect of distance function choice

To demonstrate how the distance function affects the Cityplot and the interpretation, we begin with a down-selection problem in 10 decisions. The objective function produces a vector of three components as follows:

$$\overrightarrow{f(\vec{x})} = \begin{bmatrix} \sum_{i=1}^{10}(1-x_i) * 10^{-\frac{(i-1)}{3}} \\ \sum_{i=1}^{10}(1-x_i) * 10^{-\frac{(10-i)}{3}} \\ \sum_{i=1}^{10} x_i \end{bmatrix} \quad [11]$$

That is, the function value is simply the weighted sum of decisions of $\vec{x}$ with weights exponentially distributed from 1 to 10^{-3} for the first objective. The second objective simply reverses the order of the sum. This weighting scheme is simple, provides a wide range of effects of decisions and is consistent with a 'power law' belief in the influence of factors (Andriani and McKelvey 2007). The last objective simply counts the number of 0's. We perform analysis by exhaustively enumerating the design space. In the case shown in Figure 9, we follow section 2.3.1 and use the Hamming distance as the distance between designs.

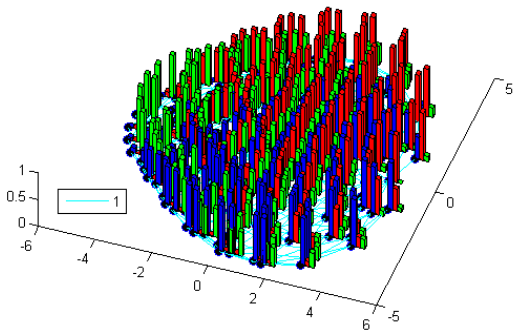


Figure 9: Hamming Distance Demonstration. Blue skyscrapers are 1st criteria, red are 2nd, green are 3rd. Roads indicate distance (see legend).

To demonstrate the significance of the distance function, we define an alternative distance function:

$$m'(\vec{x_1}, \vec{x_2}) = \sum_{i=1}^{10}((\vec{x_1})_i - (\vec{x_2})_i) * 10^{-\frac{(i-1)}{3}} \quad [12]$$

This distance function closely resembles the form of the 1st criteria but not the 2nd or 3rd criteria. It might emerge from a scenario in a multiscale problem where each decision exists on a different scale from the others. Figure 10 is the result.

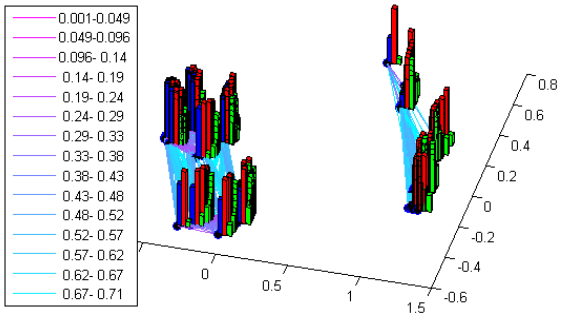


Figure 10: Exponentially Weighted Distance Function. Blue skyscrapers are 1st criteria, red are 2nd, Green are 3rd. Road colors indicate distance (see legend).

Figure 10 appears to group the designs into communities based on the 1st (blue) objective because the distance function directly corresponds to the calculation of the 1st objective. This results in more variation within the communities in the other objectives. The clustering is a result of the vast difference in weights of the distance function among the first couple of design decisions (components of $\vec{x}$) when compared to the last couple of design decisions.

In Figure 9, the tradeoffs can be seen amongst criteria. The right side of the image has designs maximizing the 1st and 2nd criteria at the cost of the 3rd; the left has designs maximizing

the 3rd objective at the cost of the other 2. The designs at the top of the image are minimizing the 2nd objective and the designs at the bottom are minimizing the 1st objective. Looking closely, it is possible to see some sharp transitions in the region with low 3rd objective as the addition of a nonzero element picks a position that causes a roughly ½ reduction in the total possible value of the 1st or 2nd objective.

Neither Figure 9 nor Figure 10 are necessarily "wrong." As noted in section 2.5, the distance function should represent the wider problem context. The former represents a case where decisions are equally important for determining design similarity and the latter represents a case where some decisions strongly dominate others.

2.7.2 Toy Example 2: Continuous Function

To demonstrate the applicability of the method to continuous spaces and spaces with many criteria, we design a simple example of a continuous objective function, perform simple optimization and show the usefulness of the Cityplot. The objective function is defined as:

$$\vec{f}(\vec{x}) = -\begin{bmatrix} \sum_{i=1}^{dim(\vec{x})}\left(\sum_{j=1}^{dim(\vec{x})}(j^i + 0.5)\left(\left(\frac{(\vec{x})_j}{j}\right)^i - 1\right)\right)^2 \\ (\vec{x}_1 - 1)^2 + \sum_{i=2}^{dim(\vec{x})} i(2(\vec{x})_i^2 - (\vec{x})_{i-1})^2 \\ \sum_{i=1}^{dim\vec{x}-1} 100((\vec{x})_{i+1} - (\vec{x})_i^2)^2 + ((\vec{x})_i - 1)^2 \\ 100\left|\left|\vec{x} + [3,3,3,3,3]^T\right|\right|_2^4 \\ 100\left|\left|\vec{x} - [1,1,0,-1,-1]^T\right|\right|_2^4 \\ \sum_{i=1}^{dim(\vec{x})}(2.5i - 2.5)|(\vec{x})_i| \end{bmatrix} \quad [13]$$

The first component is commonly known as the Perm function (Hedar n.d.), the second component is the Dixon-Price function (Jamil and Yang 2013) and the third component is the Rosenbrock function (Jamil and Yang 2013). The next two functions are the usual Euclidean norm raised to the 4th power with a minimum at the -3 vector and the [1, 1, 0, -1, -1] vectors respectively. The last function is simply a weighted L_1 norm on the components of $\vec{x}$. To ensure

the existence of optima, the solutions are restricted such that $x \in X = [-5,5]^5 = [-5,5] \times \ldots \times [-5,5]$.

MATLAB gamultiobj is used as an easy method to approximate the Pareto frontier. As there are 6 criteria, the Pareto front is a 5-dimensional space. We use the Euclidean distance as our distance as in section 2.3.4. Figure 11 is the resulting Cityplot for 150 Pareto architectures:

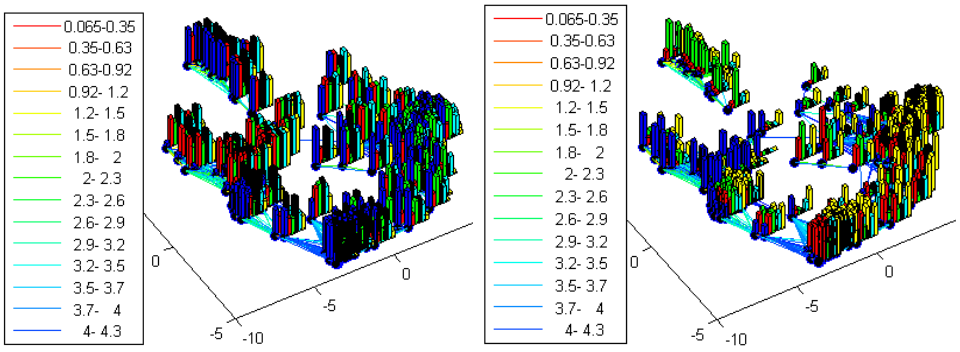


Figure 11: Continuous Toy Problem Cityplot. Skyscraper colors: blue 1st criteria, red 2nd, green 3rd, black 4th, cyan 5th, yellow 6th. Road Colors are distances (see legend). (top) taller building correspond to optimized criteria values (bottom) flipped heights so taller building correspond to sacrificed criteria values.

Figure 11 indicates that the space of Pareto optimal designs can be thought of as a multiple families of architectures. The top-left family consists of designs, which are usually around $\vec{x} = [4.5, -3, -3, -1, 1]$ with the latter two components being the most variable and ranging from -3 to about 0.25. This family accepts a penalty on the 3rd objective and a slight penalty to the 6th criteria to optimize the other objectives, especially the 4th. The left-bottom family usually looks like $\vec{x} = \left[-4, -3, -2, -\frac{1}{2}, 0\right]$, and although there is significant spread in the actual decision values, the first component is large and negative. The family accepts a penalty on the 1st criteria and occasionally the 6th criteria to optimize primarily the 2nd and 4th criteria with some designs also performing well on the 5th and 6th criteria. Continuing to move counterclockwise there is a pair of families which are usually around $[-2.5, -2.8, -3.1, 4.6, -2.0]$ and $[-2.5, -2.3, -3, 4.6, 0]$. Relative to other designs families, these families balance the criteria values. From here there is a large family comprised of the many designs from

$[-3, -3, -3, 4.7, 4.8]$ to $[2.5, 3, 3, 1.62, 2.76]$. The large number of designs in this family indicates that there are many ways to adjust these decisions and remain Pareto optimal. Generally, these designs tend to sacrifice in the 6th and 2nd criteria in favor of the 1st and 3rd criteria. There is considerable diversity in how the 4th and 5th criteria are traded with other criteria within this family. On a very close inspection, the family can be thought of as two sub-families with a number of intermediary designs between the two. One family usually keeps the first three decisions negative; the other family has all the decisions positive. The former sub-family usually performs better on 4th criteria whereas the latter prefers the 5th criteria.

2.7.3 Real-World Example 1: Earth Observing System

In addition to the previous two toy examples, we also present two real-world examples to show the applicability of Cityplot in more realistic cases.

In 1990, the US congress authorized the $17 billion Earth Observing System (EOS). Five years later, it had dropped to below $8 billion and had been de-scoped significantly. In (Selva 2012) a retrospective case study is performed on the program. After down-selection of candidate instruments to 16, they were placed onto spacecraft for launch. This can be treated as a partitioning problem with the instruments being partitioned and each subset representing a spacecraft. The transfer distance is used as the distance function.

In this problem, the Pareto set is approximated with a genetic algorithm and it finds only three designs. When the Pareto frontier is small, it is beneficial to consider more designs to gain a better understanding of the near-optimal solution space. Hence, we include not only the Pareto front, but also the next two Pareto frontiers found after removing the previous Pareto frontiers (this is known as the set of points of Pareto rank 3 or less). Taking all these frontiers together produces a set of 27 designs of roughly optimal objective values to visualize.

The Cityplot for the EOS dataset is seen in Figure 12. Science return is the retention of original science objectives in the re-scoped project, cost is the estimated re-scoped cost, operational risk is the estimated risk of failure in operation and blue is the estimated risk of mission failure due to launch failure. As is seen in the figure, transitions tend to be smooth with a couple exceptions in costs, but the cost transitions much faster than the science score, indicating that there are many of cheaper-but-not-much-less-effective ways to partition the instruments. Designs in the far left of the image tend to be mid-high science and lower cost. Very low cost designs are at the bottom and differ strongly from the other architectures. The highest-science high-cost architectures are at the top. Operational risks tend to follow cost and launch risks.

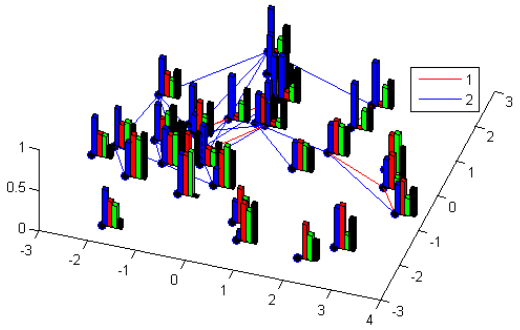


Figure 12: Cityplot of EOS Partitioning Problem. Skyscraper colors: Blue is science return, Red is cost. Green is operational risks, Black is launch risks. Road colors indicate distance (see legend).

Due to the sparsity of the Pareto front, an interesting question is: how do the criteria vary outside the Pareto front? To answer this question, we create a random subset of designs and run them through the analysis of (Selva 2012) to extract criteria values. The resulting Cityplot is shown in Figure 13.

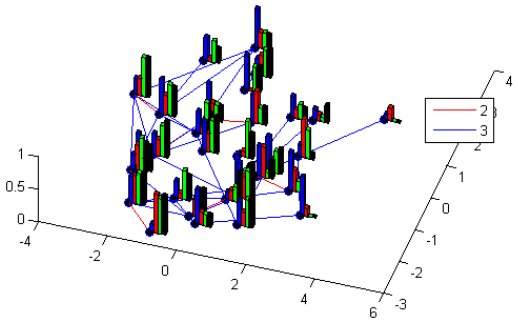


Figure 13: Cityplot of a random selection of designs in the EOS partitioning problem. Skyscraper colors: Blue is science return, Red is cost, Green is operational risks, Black is launch risks. Road colors indicate distance (see legend).

As can be seen in the legend of Figure 13, the partition space is connected with relatively few changes with designs. It comes then as no surprise that the space has some sensitive criteria. Looking at the very bottom or very left red edges of Figure 13 has dramatic changes in the 1st criterion (roughly 50% of the range of the criterion). This contrasts with the smoothness of the approximate Pareto frontier in Figure 12. Unlike in Figure 12, objectives do not always trade consistently in the randomly selected set—there are many ways to partitioning the instruments that create a suboptimal design.

2.7.4 Real-World Example 2: Guidance, Navigation and Control System

Dominguez-Garcia et al. (Dominguez-Garcia et al. 2007) conducted a study of how the components of guidance, navigation and control (GNC) systems are connected in historical NASA spacecraft. A follow-up study looking at the potential to develop a family of GNC systems was used for this chapter. In the follow-up model, a design consists of not only how many computers and sensors are included but also components are selected and how the sensors are connected to the computers. Complete designs are then evaluated in terms of weight and reliability (weight is a proxy for cost in space systems so the negative weight will be maximized).

This problem involves multiple related decisions in multiple simple problem classes (down-selection, decision-option and assignment) from section 2.3. Regardless, it still admits a

natural distance function created by composing distances of constituent sub-problems. The number of sensors/computers dictates how many can be taken, and the components dictate the connectivity pattern leading to the decision hierarchy in Figure 14.

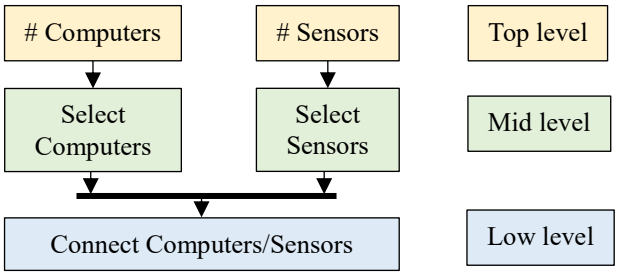


Figure 14: Decision Hierarchy for the GNC problem.

Following Figure 14, we design the following custom distance function:

$$\begin{aligned} M(\vec{x}_1, \vec{x}_2) = M_m \left(\left| \overrightarrow{x_1}^{\#S} - \overrightarrow{x_2}^{\#S} \right| + \left| \overrightarrow{x_1}^{\#C} - \overrightarrow{x_2}^{\#C} \right| \right) + \\ M_l \left(\Sigma_i I \left(\overrightarrow{x_1}^{C_i} \neq \overrightarrow{x_2}^{C_i} \right) + \Sigma_j I \left(\overrightarrow{x_1}^{S_j} \neq \overrightarrow{x_2}^{S_j} \right) \right) + \\ \Sigma_{(i,j)} \left| \overrightarrow{x_1}^{C_i \wedge S_j} - \overrightarrow{x_2}^{C_i \wedge S_j} \right| \end{aligned} \quad [14]$$

where $M_l = 9$ and $M_m = 54$ is the maximum distance achievable by just changing the low-level and mid-level or below decisions respectively. Superscripts C_i and S_j correspond to the i^{th} computer and j^{th} sensor respectively, *I* is the indicator function and $C_i \wedge S_j$ means "computer i and sensor j are present and connected." The distance function can be seen to follow the hierarchy of Figure 14 with the higher levels in the hierarchy demanding more weight to be completely distinguishable from the lower levels.

The Sammon map was used as it aids in visually distinguishing elements of the large cluster on the left.

An outside sensitivity analysis reveals that the key driving factor for the various criteria is $\min(\#S, \#C)$, which is consistent with reliability theory. This is captured very nicely in Figure 15 as a clustering behavior as the Pareto frontier distinguishes the number of sensors/computers and the distance function then correctly places these as distinct families with jumps in the log reliability.

There is a very large number of designs which use as many computers and sensors as possible. This is because when there are more sensors or computers there is more flexibility in the other choices. Looking at the substructure of the family, the distinguishing decision is the computer and sensor selections.

The criteria transition smoothly within and across designs families. This is most evident in the performance of different families when moving to the right of each figure. There are also decisions, such as number of connections or selection of computer/sensors that cause slight changes in weight and reliability, which roughly corresponds to the y-axis.

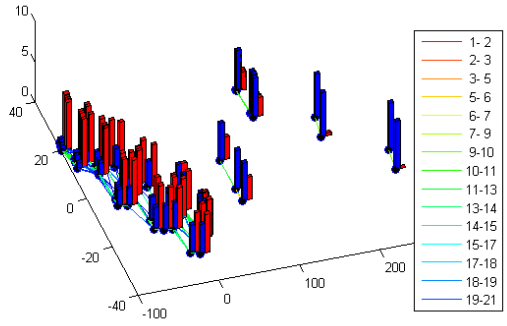


Figure 15: GNC Cityplot. Blue is weight, Red is log-reliability. Road colors indicate distance (see legend).

2.8 Validation Study Research Objectives

The research objective of this study is to assess the effectiveness of Cityplot in helping users perform a number of tasks related to design space exploration compared to the more traditional linked plots—represented by the RAVE tool. We are particularly interested in the cases where there are more than three criteria to be compared and a systems engineer wishes to consider the relationships of designs with regards to those criteria. While it does not perfectly control all possible factors, the experiment investigates whether Cityplot and to some extent other similar VR tools are promising avenues for future research in design space exploration.

At a high level, we ask if Cityplot is able to more intuitively represent the design space and allow users to get better insights faster. We also ask if users prefer Cityplot or more

traditional approaches subjectively. In designing the experimental measures, we focus on the process of knowledge discovery and ability to answer questions about the design space quickly and accurately. The assumption is that if users are not unduly inconvenienced and able to generate accurate understanding of basic facts quickly, then they will tend to have better overall understanding of the design space.

More concretely, we measure performance, confidence, and time taken to perform the following tasks:

T 1. Finding a design that meets certain requirements on the criteria values (Requirement Satisfaction).

T 2. Finding a design as similar as possible to a given design but with a slightly different set of criteria (Requirement Change Satisfaction).

T 3. Finding a design such that a small decision change leads to a large change of a criterion (Decision-Criterion Sensitivity).

T 4. Assessing if or to what extent distinct clusters or groups (families) of similar designs appear in the *decision* space (Design Family Assessment; Product Platform Analysis).

T 5. Assessing if or to what extent distinct clusters or groups of similar designs appear in the *criteria* space (Design Family Assessment; Analysis for Marketing).

See sections 2.10.1 through 2.10.4 for more information on the specific tasks, as well as the exact question text and scoring scheme for each task. Note that the tasks are numbered in an approximate order of increasing level of abstraction, from task 1 –a basic information retrieval task—to tasks 4 and 5, which concern higher-level information processing.

We hypothesize the following:

H1: Users will obtain better answers on tasks 2-5 with Cityplot (where "better" is defined differently for each task but generally has to do with accuracy or distance of the answer with respect to the true or best answer).

H2: Users will perform tasks 2-5 faster with Cityplot, but users will perform faster on the task 1 with RAVE.

H3: Users will feel more confident with Cityplot on tasks 2-5 and more confident with RAVE on task 1.

H4: Users will find Cityplot easier to use for tasks 2-5 and RAVE easier for task 1.

H5: Users will find Cityplot easier to learn, more engaging, more practical and better integrated than RAVE in subjective questioning.

We hypothesize that RAVE is easier for the first task because it supports filtering capabilities that allows a user to quickly isolate the subset of satisfactory designs by merely adjusting the filter sliders. We expect Cityplot will tend to perform better on the other tasks as Cityplot represents the relations between decisions and the decision→criterion mapping naturally in the visualization (which is particularly useful for T3).

2.8.1 RAVE Motivation, Purpose and Use

RAVE is a tool developed at Georgia Tech to build and manage simulations, optimize engineering model performance and visualize the results. For this experiment, RAVE was chosen as benchmark tool as it represents the state of practice, characterized by commonly used visualizations with a means to link them (see section 1.2.2.1).

We opted to use a pre-built option as opposed to developing our own benchmark tool in Python or Matlab to save experimenter effort and provide a baseline that's easier for other experimenters to replicate. RAVE is a published free and open source option with some history

in the engineering community that supports these plotting and linking behaviors (Daskilewicz and German n.d.).

A primary advantage of the RAVE tool is familiarity. The interaction model is very conventional, relying on mouse clicks and selecting variables. There is no need for keyboards, user-generated code or calculations (as is the case with MATLAB or scipy). The plots can be dynamically reconfigured by selecting other data variables to plot or interacting with the plot itself. Additionally, the plots used are familiar to most engineers. The linked plotting capabilities allow for corresponding the data from one plot to another, which facilitates understanding of the underlying dataset by interacting with the plots. Finally, filtering and linking allows some plots to "control" the others. A user can create a plot with a variable set as a constraint, modify that constraint, and see how other variables responds to that constraint by seeing which designs are highlighted or filtered in the other plots.

We chose the scatterplot, histogram and parallel axis plots as these are common visual techniques for visualizing multi-objective engineering design data. We opted not to include a table as there were concerns it could lead to users scanning through the numbers in the table over using the visual elements we wanted to test.

2.8.2 RAVE Basics

RAVE allows the freeform creation of drag-and-drop GUIs consisting of interactive plots and tables. For repeatability, rather than have users create their own UI within RAVE, we created a GUI consisting of three linked plots (scatterplot, parallel axis and histogram) for subjects to use (Daskilewicz and German n.d.) which can be seen in Figure 16. While users were permitted to modify the interface, none did so.

Linking the plots strongly improves the visualization, as selecting an element on one plot with a mouse highlights the design in the other plots and filtering the results on one plot greys out designs on all the plots (see the purple elements in Figure 16).

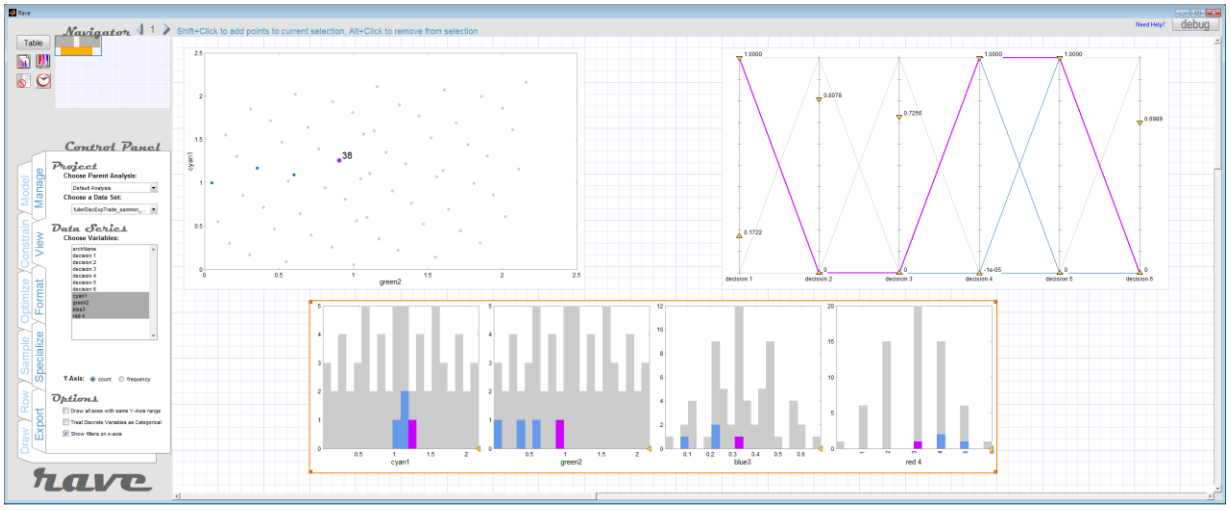


Figure 16: Example RAVE Window

RAVE's scatterplot (Figure 16, top left) allows users to plot up to two attributes and plots each design as a point in 2d. Hovering the mouse over a point shows a number corresponding to the design name or identifier, and designs can be selected individually by clicking on them or by clicking and dragging over an area to select multiple designs. The attributes shown on the axes can be selected by the user in the dialog box on the left.

RAVE's parallel axis plot (Figure 16, top right) allows users to plot arbitrarily many attributes by selecting the quantities in the view menu. Each design is then a line and can be selected or hovered to get the design name. Each axis has a filter that can be set by dragging a triangular indicator to grey out designs that do not meet a specified threshold value for the filtered axis.

RAVE's histogram plot (Figure 16, bottom) allows users to see roughly how many designs have a certain value or value range for a quantity. Multiple histograms can be plotted at once to see multiple variables and how they relate.

RAVE treats decisions and criteria equivalently; that is, it does not distinguish between design decisions and criteria, they are all considered design attributes. In practice, this means that decisions can be plotted against other decisions, criteria can be plotted against criteria and decisions can be plotted against criteria.

2.8.3 Experiment Specific Details

Design Decisions were labelled as "decision #" and were numbered in ascending order; criteria were labelled by a color and a number (e.g. "cyan1"). The number corresponds to the position of the decision or criteria in the RAVE drop downs. This was to aid users in finding the decisions and criteria and prevent users from attempting to make guesses on the relationships

between decisions and criteria based on the names. For criteria, the color corresponds to the coloration of the building for Cityplot (see next section).

2.9 Experimental Design and Methodology

The goal of the experiment is to compare the performance of users of Cityplot and RAVE in various tasks related to design space exploration. The experiment was then designed to try and mitigate certain confounding factors that could affect the results of the experiment.

2.9.1 Basic Process

For a given subject, the experiment went as follows: The subject was greeted by the experiment supervisor and given a sheet with an image to define the term "grouped" for tasks T4 and T5 (cf. section 2.10.4 and Appendix) to be used during the tutorial and the experiment. The subject was then told that they could use the keyboard or the controller when the program told them about both interfaces and they could switch between the two at will.

The remaining experimental procedure was automated using Python. The python script ensured every subject had signed an informed consent form and then brought up the first tool (Cityplot or RAVE, according to the blocking in section 2.9.2) and ran a tutorial to show subjects how to perform basic tasks with the given tool. After completing the tutorial, subjects would then use the tool to answer questions on each of the two datasets. The tutorials scripts are available in the appendix. Questions were asked via a standard Windows 7 CMD shell, which can be seen in Figure 17. RAVE used a SikuliX script (Hocke 2018) and loaded automatically with the GUI. CityplotVR had command line integration and loaded itself using command line calls in Python.

The script also measured time. If the user went over the specified time limit for a question, it would cause the command line window to flash and inform the user that time had

expired every 5 seconds for 23 seconds (0.75 seconds were used for the flash). Afterwards, it would flash a different color and inform the user it would be possible to skip the question.

The automated script would then repeat the tutorial for the other tool and ask questions using the same two datasets with the other tool. Finally, the script asked a series of subjective questions and a short informal exit interview was conducted to ask users if they had any further comments.

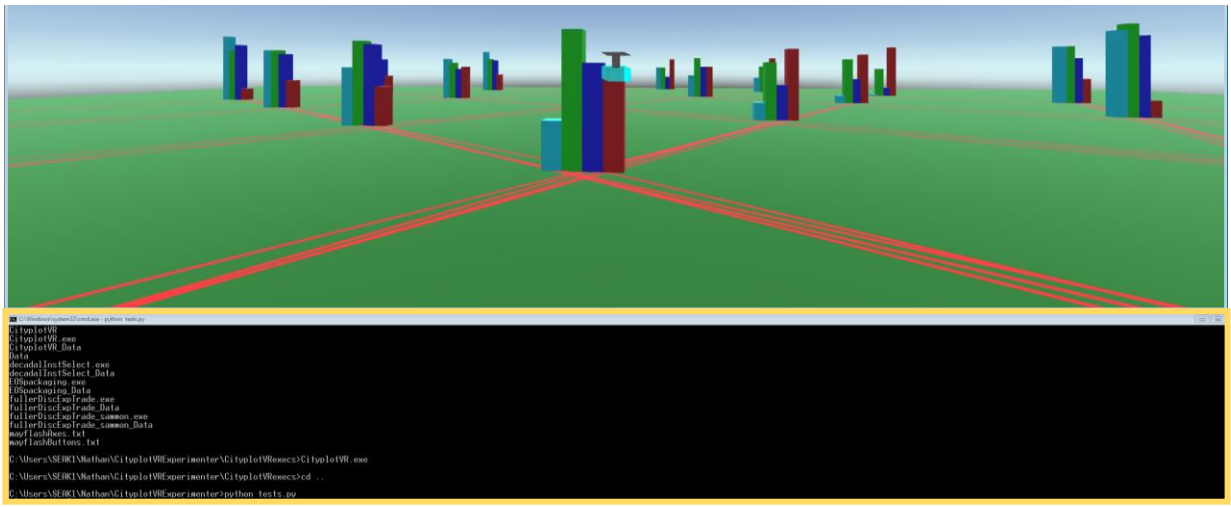


Figure 17: Question Window (highlighted with yellow box)

2.9.2 Confounding Variable Controls

Some variables that affect the result of the experiment are:

1. Personal factors, such as user technical abilities when using a computer or initial knowledge regarding visualization and design space exploration.
2. Learning effects, where subjects learn the questions and tools as the experiment continues (for example, the second tool a subject uses might get a higher score due to the benefit from understanding of the tasks generated from the first).

3. Human interaction factors, where experimenter staff or perceptions from the subjects on the nature of the experiment may bias the results of the experiment (the classic “Clever Hans” phenomenon (Ferguson 2019)).
4. Dataset factors, due to the selection of the specific datasets used in the experiment
5. Display and mode of interaction factors, due to the different mechanisms used to visualize and interact with the data. Cityplot is fundamentally 3d whereas RAVE is fundamentally 2d. Relatedly, users tended to prefer the game controller (but not always) for Cityplot. While this is a part of the experiment, it makes it difficult to distinguish which differences between tools are driving differences in results as the tools confound the large number of factors that could impact performance.

Personal factors, dataset factors and learning effects were partially addressed by blocking and randomization so each user runs each tool on multiple datasets (within-subjects experimental design), and the order in which datasets and tools were presented was balanced to eliminate bias. Test subjects were randomly assigned to balanced blocks (Montgomery 2013) in which the order of datasets and visualization methods were permuted. Since there were 2 datasets and 2 tools, this led to 4 blocks, and users were assigned to blocks randomly.

Human interaction factors were addressed by automating the asking of questions and giving of instructions during the tutorial.

The display and mode of interaction factors are inherent parts of each tool and thus hard to control. A potential avenue for future work might be to create versions of Cityplot or RAVE that begin sharing the visual (such as a 3d scatterplot version of RAVE) and interaction characteristics of the other program (such as adding global filters in Cityplot). Further research might also quantify how minor changes to the conditions impact performance (e.g. does using an

Xbox controller cause significant changes from a Nintendo GameCube controller) but such concerns were considered outside the scope of this experiment.

A small, 6 person, pilot study was run to verify the experimental process and data collection software worked and experimental time limits were met. Additionally, the above described experimental procedures were refined to eliminate confounding variables and question wording was slightly modified to be less confusing. The results of the pilot were not used in analysis sections 2.12 and 2.13.

2.9.3 Datasets Used

The two datasets used in the experiment and a third dataset used in the tutorial are described below.

For the purposes of the experiment, while the "real world" datasets have names for each of the decisions (e.g., instruments, orbits) and criteria (e.g., science value, cost), the decisions were renamed "decision #" and the criteria were renamed "cyan1," "green2," "blue3" and "red4" to make it easier for subjects to associate building colors with criteria, to make it easier to find the criteria in the list of options in RAVE, and to eliminate confounding variables. For example, a clever user might be able to infer based on the variable names (e.g., "has synthetic aperture radar") relationships to other variables and criteria (e.g., implies high costs due to it being a complex high-energy instrument). However, background knowledge is non-uniform among participants and allows "bypassing" RAVE or Cityplot tools. Hence, the variables were all renamed to the nondescript names mentioned before that emphasize their role within the tools rather than their role within the original design problems.

We selected datasets that would have four criteria as this exceeds the normal capacity of a 3d scatterplot and thereby tests the ability to represent and visualize more than three criteria.

Some real world datasets were chosen to prove applicability to realistic problems a system engineer might encounter (though the context was removed as per the previous paragraph) and a synthetic dataset was used to represent a case where the dataset shows perfect non-grouping behavior, as this behavior was not found in any of the real-world problem to which we had access. Ideally, more datasets including datasets with more criteria could have been used, but maintaining user interest and recruiting subjects for beyond an hour was seen as too difficult and limited the size of the study.

Dataset 1: This dataset was used in the tutorial phase and is related to a retrospective study of the NASA Earth observing system (EOS) (Selva 2012). The goal was to see if the implemented set of three missions (Terra, Aqua, Aura) carrying a total of 16 instruments was close to optimal under the technical and programmatic assumptions of the time. The design space consists of all possible partitions of the set of 16 EOS instruments. For each design in the dataset, there are four criteria: science benefit, lifecycle cost, programmatic risk, and launch risk. A genetic algorithm was used to find an approximate Pareto set of 32 designs. This dataset was chosen because it contained examples of both grouping and non-grouping behavior in the design space (see definition of tasks T4 and T5 in section 1). For more information on this dataset see (Selva 2012). The initial appearance of Dataset 1 can be seen in Figure 18.

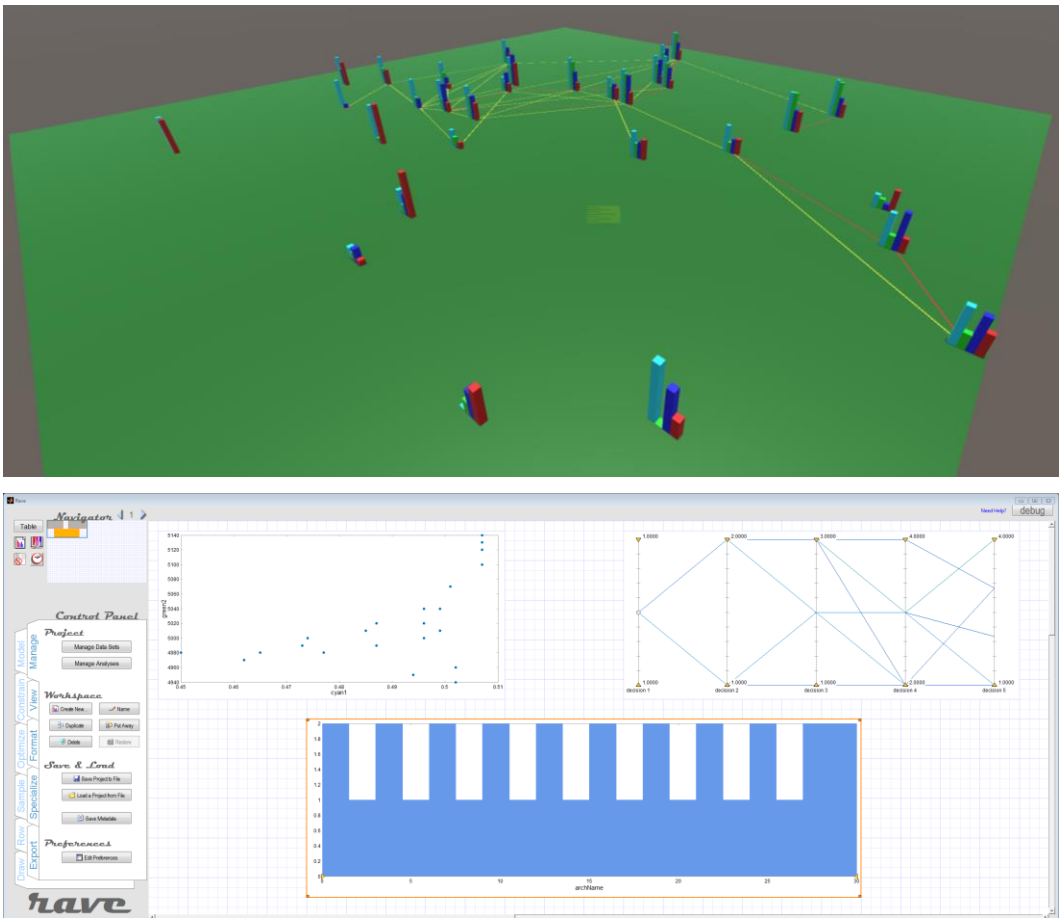


Figure 18: Dataset 1 as seen in (top) Cityplot-VR (bottom) RAVE

Dataset 2: This dataset is related to a study of the 2007 Earth Science Decadal Survey. The Decadal report recommended a set of 17 satellite missions for NASA and NOAA to implement, but in light of the rise in cost estimates and budgetary constraints, a study was conducted to identify de-scoping options to meet most science criteria at lower cost (Selva, Cameron, and Crawley 2014). The design space consists of all possible subsets of a set of 39 candidate instruments. There are four criteria in this dataset: science score, cost, programmatic risk and fairness. A genetic algorithm was used to find an approximate Pareto set of 58 designs that constituted the dataset. This dataset was chosen to represent a "grouped" design space. For more information on this dataset see (Selva, Cameron, and Crawley 2014). The initial appearance of Dataset 2 can be seen in Figure 19.

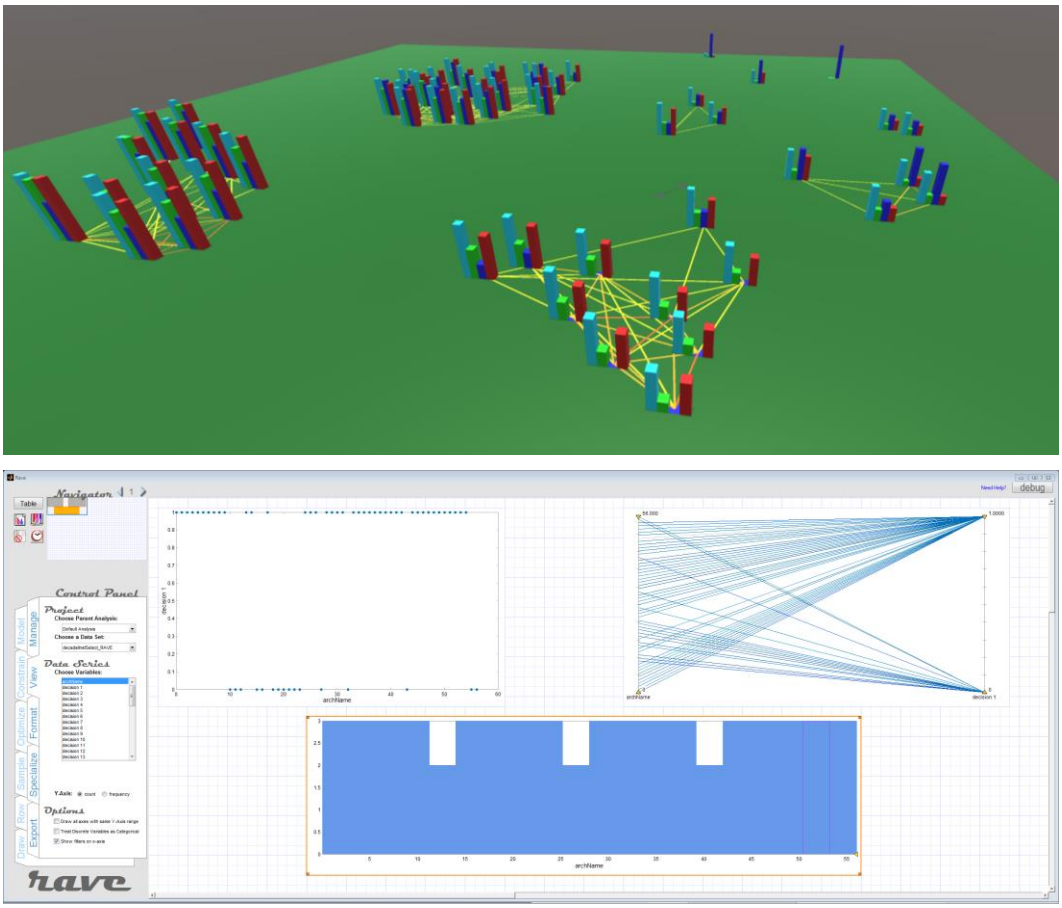


Figure 19: Dataset 2 as seen in (top) Cityplot-VR (bottom) RAVE

Dataset 3: To test the condition where designs were highly un-grouped and regular, we used a synthetic problem created by the following:

Designs are represented by a 0/1 vector of length 6 (denoted x_i for each component) and the criteria are then evaluated as:

$$y_1 = \sum_{i=0}^{5} 6^{-\frac{i}{3}} \cdot x_i \quad [15a]$$

For criteria 1 (cyan1),

$$y_2 = \sum_{i=1}^{5} 6^{\frac{i}{3}-6} \cdot x_i \quad [15b]$$

For criteria 2 (green2)

$$y_3 = \sum_{i=1}^{6} \frac{e^{-\frac{(v_i-3)^2}{18}}}{3\sqrt{2\pi}} \cdot x_i \quad [15c]$$

For criteria 3 (blue3) and

$$y_4 = 6 - \sum_{i=1}^{6} x_i \quad [15d]$$

for criteria 4 (red4). All criteria are to be minimized. This dataset is thus designed so that cyan1, green2 and red4 are always in conflict. Minimizing cyan1 requires the later elements of x_i to be 1 while minimizing green2 requires the edge elements to be 1. However, attempting to minimize both cyan1 and green2 simultaneously by setting all *x* components to zero maximizes red4; forcing cyan1, green2 and red4 to trade against each other directly. blue3 acts as another possible means for Pareto optimality and prefers the "middle" elements of x_i to be 1. The initial appearance of Dataset 3 can be seen in Figure 20

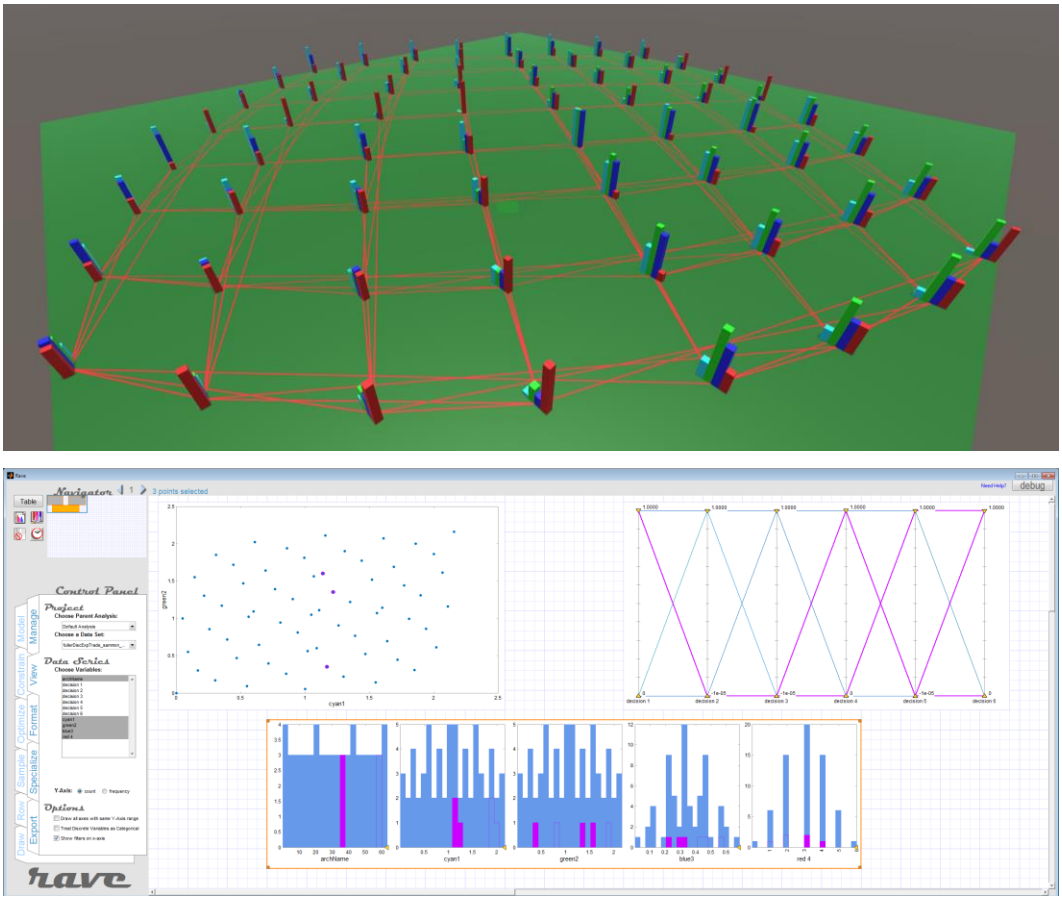


Figure 20: Dataset 3 as seen in (top) Cityplot-VR (bottom) RAVE

2.10 Tasks, Questions, And Scoring

The bulk of the experiment consisted of asking users questions and having them perform certain tasks to answer them. We elaborate on the questions and tasks in this section. The task-based tests broke into five tasks with two questions each. More tasks and questions would have been desirable, but time limitations for subjects constrained not only the number but also the length of questions that could be asked. Based on findings from the aforementioned pilot study, some questions had attached time limits, which would advise subjects on how long it was expected they would take to answer the question. Such time limits are noted in the following paragraphs (each paragraph is a question). The questions were designed to be "fair" to both RAVE and Cityplot by avoiding references to any feature of either tool or the appearance of any feature within a tool.

2.10.1 T1: Requirement Satisfaction

The first task simulates a system engineer checking if a design satisfies the requirements for a program or evaluating how an individual design meets the specifications.

The wording of the question was: "*Find a design that has criteria1 value ≥ #, an criteria2 value ≤ # and an criteria3 value ≥ #*" where the values were chosen to be within the upper half of the first criteria, lower ¾ of the second criteria, and upper ¾ of the third criteria, leaving about 28% of the designs as permissible. Subjects would enter the design identifier of a design they found appropriate. As the automated script would prevent users from choosing designs that did not exist or failed to meet specifications, users would inevitably give correct answers, so this question wasn't scored for accuracy. Instead, the time taken to find a feasible design was recorded. In addition, a follow-up question asked "*How certain are you that the design is within specifications [1-5 scale: 1 basically guessing, 5: completely sure]*"

2.10.2 T2: Program Change Satisfaction

The second task was to simulate the common situation in a program where changes in the client's priorities can force a shift in what design is selected. Oftentimes, to prevent extensive rework, the system engineer will want to stay as close to a design-in-progress as possible while adjusting to meet the new requirements. This is also a step in evaluating if a proposed design will have sufficient flexibility should the program priorities change later or if there will need to be a progression of solutions to make a final implementation from a currently extant system (staged deployment).

The specific question wording was "*You selected design <name of design from before>. Make as small a design change as possible (as few decisions changed as possible) such that the new design is smaller in criteria 3. Input the new design.*" The automated script prevented users from inputting designs that failed to reduce criteria 3 (but allowed other criteria to fail to meet the criteria of the previous question). Scoring was done with:

$$S_m = \frac{\Delta x}{L} \quad [16]$$

Where Δx is the distance between the design chosen in the previous question and the design chosen in this question and L is a characteristic (maximum) distance set by the dataset, to facilitate comparisons across datasets (L=29 for Dataset 2 and L=6 for Dataset 3). Observe that for this question and this question only, smaller numbers are preferred for score. The subjects were given a follow-up question "*How certain are you that the design is as close as possible while being smaller in the desired criteria [1-5 scale: 1 basically guessing, 5; completely sure]*"

2.10.3 T3: Decision-Criteria Sensitivity

The third task was to evaluate the sensitivity of the dataset defined in a manner analogous to Lipschitz continuity, where a distance ε in the decision space was fixed and users were asked to then try and estimate the biggest change in a criterion value within that fixed distance. This

task was designed to act as a proxy for an engineer trying to assess how important design decisions are in driving the criteria. It serves as an extension of task 2 (described in section 2.10.2): in task 2 we look at changes close to a design of interest whereas here we attempt to characterize the global sensitivity to changes overall. This can be useful from a programmatic point of view as it can suggest if a decision, or set of decisions, must be made early as they will have a major impact on later system flexibility.

The question was worded: "*Input any pair of designs for which a small design change (<distance> decision changes or fewer) creates a big change in the <criteria> criteria (ideally, as large as possible, but you don't need to go out of your way to find the largest).*" The distance ε was chosen to be $\varepsilon = 1$ for the synthetic dataset and $\varepsilon = 5$ for the decadal instrument selection dataset. Compared to the previous tasks, this one is more complicated, so it was asked twice in the experiment with 2 different criteria (the 2nd and 4th criteria, to be precise). A timer was set for 3 minutes for this question. Scoring was done with the index:

$$S = \frac{\Delta y_d}{\max_{\mathrm{d,i,j:m}(x_i,x_j)\leq\varepsilon} |y_{di} - y_{dj}|} \quad [17]$$

where Δy_d is the change in criteria for the designs found by the subject, y_{di} is the value of the dth criterion for the ith design, and $m(x_i, x_j)$ is the distance in the decision space between designs i and j. ε is set by the dataset and is the fixed distance used to define "close" (5 in Dataset 2 and 1 in Dataset 3). That is, the index score is calculated as the change in criteria values for the designs the participant picked divided by the largest possible change in criteria subject to the constraint that the decision changes were within the values specified. A follow-up question was asked "*How certain are you that the designs you picked are the largest criteria change for the design change allowed [1-5 scale: 1: basically guessing, 5: completely sure]*."

2.10.4 T4, T5: Design Family assessment

The last two tasks were to assess the presence of design families, i.e., sets of designs that are similar in terms of their decisions and/or criteria. To do this, we considered "clustering behavior" of the Pareto-optimal points, where we defined clustering in a manner analogous to single-linkage clustering (Hastie, Tibshirani, and Friedman 2009). Design families are useful in three cases. First, they can be used to reduce the size of the alternative space and account for the presence of uncertainty, so the system engineer chooses a design family as opposed to a single design, as in set-based design (Avigad and Moshaiov 2009). Second, this also represents a case where a system engineer realizes they may need to make changes to the system to change the performance from one regime to another due, for example, to changes in customer priorities, and considers the sequence of design changes to implement the transition from a reference system. Third, the presence of families of similar designs in the decision space can suggest the creation of similar products that achieve a wide range of performance to meet the needs of different markets (Simpson 2004; Simpson, Maier, and Mistree 2001). A not clustered space will allow for a more gradual transition from an initial design to a target design (second case mentioned above), whereas a more clustered space will open up the possibility of distinct design families.

The first such question evaluated the clustering in terms of the **decision changes** and was worded: "*Would you say there are groups of designs in the DECISIONS? A group of designs is a set of designs such that designs within the group can be transformed from one to another by only making <number> changes at a time. A non-grouped dataset will let you transform any design into any other design. A grouped data set will have groups which can't transform into each other. (see printout for an illustration) [y/n]*" The referenced printout is available in the appendix. The *number* asked for in the question was 1 for the synthetic dataset and 5 for the

decadal instrument selection dataset. The datasets were specifically chosen to have different behavior on this question, with the synthetic dataset being not grouped and the decadal instrument selection being grouped. The automated script only accepted answers of the form 'y/n' or 'yes/no.' A time limit was set for 2 minutes. Scoring was based on whether users answered the yes or no question correctly (1 point for a correct answer, 0 for an incorrect answer). A follow-up question was asked "*How certain are you that the set of designs is/is not grouped? [1-5 scale: 1: basically guessing, 5: completely sure]*"

Analogous to the previous question, it is possible to think of the grouping in terms of the **criteria values**. The question was then worded: "*Would you say there are groups of designs in the CRITERIA? This is like the previous question except instead of specifying a number of design decision changes, we are looking for jumps in the criteria. A group of designs is a set of designs such that designs within the group can be transformed from one to another by only making small criteria changes at a time. A non-grouped dataset will let you transform any design into any other design. A grouped data set will have groups which can't transform into each other (see printout for an illustration). Remember there are multiple criteria so a single criteria plot will likely be insufficient. [y/n]*." Scoring, answer checking, the time limit and the follow-up were the same procedure as the previous question.

2.10.5 Subjective User Evaluations

Following the task-based questions with both datasets and both tools, users were asked a number of subjective questions based on standard usability questionnaires (Brooke 1996). Pairwise preference between the two tools was chosen as a metric instead of a Likert scale, as it is easier for subjects, more consistent and less subjective (Carterette et al. 2008). The questions were worded as follows:

1. *Which tool was easiest for picking a design that meets target criteria values? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
2. *Which tool was easiest for finding a design similar to another design? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
3. *Which tool was easiest for assessing how much changes to design decisions change criteria values? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
4. *Which tool was easiest for assessing if there was grouping of design decisions, to see if one could hop from one design to another with small changes in decision values? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
5. *Which tool was easier to use (think in terms of manipulating the tool to find information overall?) 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
6. *Which tool was easier to learn? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
7. *Which tool was more engaging? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
8. *Which tool was more practical? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
9. *Which tool was more cohesive and well-integrated? 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*
10. *If you were to redo this experiment with only one tool, the tool I would use is: 1-Scatterplots, Histograms and Parallel Axis 2- Cityplot-VR*

As all these questions are pairwise comparisons, we simply count the number of subjects who preferred one tool to the other as a binomial random variable.

These questions were meant to serve as a kind of final “what did people prefer” approach to comparing the design visualization tools. It serves to confirm that users actually preferred a given tool and help explain the objective measures. For example, if a tool were harder to learn

but more engaging, it could be inferred that users put more effort into the tool. If given a choice, what would users find more useful for a given task (regardless of actual performance)?

2.11 Subjects

For the experiment, we had 23 subjects. Two were excluded for nonsensical answers that managed to slip through our answer validity checks (e.g., answering the question "choose a design that…" with the sentence "cyan1 decreases as red4 increases" instead of a design identifier). A third subject was eliminated due to irregularities in the experiment conditions (the tutorial sheet was missing and not all the equipment worked as intended). Of the 20 subjects included for analysis, all but two were students in a STEM subject, with the exception of a lab technician and an economics student. Of the 18 students, 3 were self-identified PhD students, 7 were some other kind of graduate student and the remaining 8 were undergraduates. All subjects were affiliated with Cornell University. 9 answered that they would "frequently play 3d video games", 12 would have "frequently used 3d modelling." All but 2 had frequently used scatterplots, all but one had frequently used histograms but only 2 had frequent experience with parallel axis plots and all but 3 claiming to have never heard of parallel axis plots. No subjects had frequent experience with Cityplot, with all but 4 having never heard of it. Based on this data, we believe we have a non-expert sample that has some experience in 3d virtual spaces and has heard of the most common plots, but few of the more uncommon options (parallel axis plots and Cityplot). The experimental design and procedure was approved by the Cornell IRB.

2.12 Results

The results for the tasks are shown in Table 5. Scores for each question are calculated according to section 2.10. The mean is the average score over test subjects and datasets. The data was tested for significance using a paired Wilcoxon significance test. As outlined in section 2.9,

subjects and datasets were paired across tools to control for confounding factors in the Wilcoxon test. That is, for each test subject, quality, effort and time quantities were subtracted between the Cityplot and Rave tools, the differences were ranked, and the sum of positive ranks was compared to the theoretical values to produce a p-value (overall means and standard deviations for tools are for illustration purposes only).

Table 5: Paired test results of task-based performance questions. Bold and indented means are preferred values. Wilcoxon is paired across users and datasets to control for confounding factors. P-values lower than a significance level of 0.05 are highlighted in bold.

	Mean		Standard Deviation		Paired Wilcoxon Test
Question	Cityplot-VR	Rave	Cityplot-VR	Rave	(p-value)
T1. Find a Design	Always Correct				
Sure Design Works	***4.73***	4.33	0.82	1.02	***0.0278***
T2. Find a similar Design	***0.35***	0.45	0.21	0.22	***0.0273***
Sure smallest possible	***3.88***	2.93	1.09	1.27	***0.0014***
T3. Maximize Sensitivity	***0.71***	0.46	0.32	0.40	***<0.001***
Sure Picked Biggest	***3.40***	1.85	0.98	1.19	***<0.001***
T3. Maximize Sensitivity	***0.81***	0.64	0.32	0.43	***0.0011***
Sure Picked Biggest	***2.95***	1.75	0.93	0.87	***<0.001***
T4. Decisions Grouped	***0.58***	0.43	0.50	0.50	0.0915
Sure Decisions Grouped	***3.28***	2.15	1.18	1.08	***<0.001***
T5. Criteria Grouped	***0.53***	0.45	0.51	0.50	0.4579
Sure Criteria Grouped	***3.28***	2.35	1.01	1.17	***<0.001***

As users could input nonsensical answers and the script would simply ask the question again, Num Asks (cf. Table 6) indicates the number of times users were asked the same question. A significant difference in the number of asks then indicates user uncertainty or that users could have resorted to guessing, so a smaller number of asks is preferred.

Additionally, since users could give up if they took too long on a timed question (see section 2.10 for the timings), we look at the number of times users gave up. Users very rarely chose to give up with either tool. We combine the time taken, number of times a question got asked and the number of times users had to give up as proxies for effort and give the results in Table 6. Users only failed to complete the task when maximizing sensitivity, in which there was a significant difference in the number of give ups between RAVE and Cityplot. Similarly, the time taken by the subjects was highly statistically significant suggesting that the number of give ups is significant in these cases.

Table 6: Effort Table: Paired results (total time and number of asks) favor cityplot when positive. Give ups is the actual count. Wilcoxon results are p-values. Times are seconds.

	Paired Number of Asks		Number of Give Up		Total Time Elapsed	
Question	Cityplot-Rave	Wilcoxon	Rave	Cityplot	Cityplot-Rave (s)	Wilcoxon
T1. Find a Design	-0.600	0.111	0	0	-180	0.0975
Sure Design Works	0.0250	0.814	0	0	129	1.00
T2. Find a similar Design	-0.350	0.692	0	0	-44.8	0.279
Sure smallest possible	0	0.177	0	0	2.29	1.00
T3. Maximize Sensitivity	1.78	***<0.001***	***9***	***1***	522	***<0.001***
Sure Picked Biggest	0	0.835	0	0	1.07	1.00
T3. Maximize Sensitivity	1.15	***0.00135***	***5***	***1***	208	***0.00115***
Sure Picked Biggest	0	0.941	0	0	0.482	1.00
T4. Designs Grouped	-0.0250	0.605	0	0	5.11	1.00
Sure Designs Grouped	0.0250	0.364	0	0	-0.308	1.00
T5. Criteria Grouped	0.0250	0.330	0	0	0.338	0.773
Sure Criteria Grouped	0.0500	0.532	0	0	1.07	0.346

The results of subjective questions are shown in Table 7. The columns are simply counts of users who preferred one tool to the other. Using a binomial distribution for N=20, the calculated p-values in Table 7 use the exact distribution for a hypothesis test.

Table 7: Results of Subjective Questions.

Question Form	Question Evaluation	RAVE	Cityplot	p-value (Binomial)
Easier tool for	Meeting target obj.	8	12	0.240
	Finding similar Design	3	17	**0.00217**
	Decision-Criteria sensitivity	3	17	**0.00217**
	Assessing design families	3	17	**0.00217**
Which tool was	Easier to Use	9	11	0.320
	Easier to Learn	7	13	0.148
	More engaging	0	20	**<0.001**
	More Practical	7	13	0.148
	Well-integrated	6	14	0.0739
If I redid the experiment, I would use:		1	19	**<0.001**

2.13 Discussion

For the most part, the objective and subjective results suggest that Cityplot is more effective, more user-friendly and preferred by significantly more users in the test.

Before users were given the tasks and evaluations, we collected basic background information on the test subjects as described in section 2.9.1. Breaking into groups and running independent t-tests resulted in large p-values ($p=0.4$ or greater) for the vast majority of tests and adjusting for the repeated testing revealed that there were no significant factors from background in explaining the performance differences in the experiment (cf. section 2.11 for demographic information).

In the remainder of this section, we describe the results of the hypotheses posed in section 2.8.

2.13.1 Answer quality (H1)

Support for the first hypothesis can be best assessed via the shaded rows in Table 5. We find that Cityplot outperforms Rave on most tasks and seems to perform no worse in any of the tasks that was selected. Since Cityplot is designed to present decision changes and criteria together, it is unsurprising that it should perform better at finding a similar design in response to changes in performance requirements (task 2, $p=0.0273$). On task 3 (sensitivity), Cityplot produces markedly higher quality answers ($p<0.001$ on both iterations). Task 4 (grouping in decision space) is more difficult to judge as it is only marginally significant ($p=0.09$), but it does favor Cityplot. Cityplot is likely easier to tell grouping at a glance than RAVE by nature as the display is one of designs represented in 3d space instead of numbers and lines on a screen. There is no clear result for distinguishing if the criteria are grouped (Task 5). We hypothesize that this is because the visual display for Cityplot is less clear when attempting to group on criteria

instead of decisions; additionally, the grouping task itself is relatively difficult to understand. It should be noted that in no particular task did Cityplot perform worse in the average than Rave.

2.13.2 Answer speed and effort (H2)

For the most part, the time it took users to answer questions was highly inconsistent, as can be seen in the p-values in Table 6. As such, we see no evidence supporting hypothesis H2 regarding the users performing task 1 faster with RAVE nor can we say that users performed tasks 2 or 4 faster with Cityplot. We cannot say the causality goes the other direction either, only that users seem to follow the old adage "the time expands to meet the allowed time of the project."

That said, there is a clear advantage for Cityplot in task 3 (sensitivity). Not only did users take significantly less time, but there is a very large gap in the number of users who were unable to answer the question in reasonable time and had to give up. This is also the task with the largest performance gap (see subsection 2.13.1) so there is strong evidence that Cityplot is better for evaluating global sensitivity than RAVE.

2.13.3 Answer Confidence (H3)

As can be seen in the shaded rows of Table 5, users felt much more confident in their answers with Cityplot than RAVE, supporting H3. This result holds to be extremely significant regardless of the task in question. We believe that this is because, while RAVE uses graphs that are more commonly recognizable and widely used, Cityplot is more "intuitive" in the sense that it provides a view that users are more likely to recognize and internalize, which leads to a higher degree of confidence in the answers given. As will be seen in the next subsection, this agrees with the general user opinion that Cityplot is easier to use and understand.

2.13.4 Ease of Use and User Experience (H4, H5)

The results of the user end-survey are given in Table 7. Of note, regardless of the question asked (easier to perform each of the tasks, easier to use or learn, more engaging, practical or well-integrated), more users preferred Cityplot than Rave, supporting H4 and H5. Using a binomial significance test indicates a statistically significant preference for Cityplot as being easier to use to perform all tasks except task 1 (which we hypothesized users would perform faster with RAVE). In general ease of use, users broke nearly evenly with a slight preference for Cityplot. Ease to learn and practicality were Cityplot-favored but not to an amount that would pass a $p<0.05$ threshold. *All* users found Cityplot more engaging. Based on the results of the background questions, it is clear that users were more familiar with the basic ideas behind RAVE before the experiment and this likely colored their opinions of the general ease of use and ability to pick up the tool. It should be noted that the general ease of use question followed the task-based ease-of-use questions (in which users would have to think about the mechanics of how they had actually performed the experiment) so it is likely that this may have affected opinions on this one question. The unanimous opinion that Cityplot was more engaging is sensible when one thinks of the relative task of navigating a space as if in a video game as opposed to selecting items from lists and looking at well-established traditional plots.

The vast majority of users answered the question "if I redid the experiment with only one tool I would use" in favor of Cityplot, which is an overall endorsement of Cityplot for the tasks in the experiment.

2.14 Conclusions

We have presented a visualization technique that combines design similarity-based dimensionality reduction and multi-criteria analysis to aid in systems engineering and design

decision-making. In this technique, designs are compared by a similarity measure/distance function and the space of Pareto optimal designs is reduced to a 2-dimensional plane with the values of decision criteria plotted on the same chart to allow for examining how the criteria values change with changing decisions.

Because all criteria are plotted together for multiple designs, it is possible to examine the tradeoffs between decision criteria while maintaining a notion of what designs and decisions make which tradeoffs. The tradeoffs can be seen as a transition in the criteria values as one moves among the space of Pareto optimal designs. This also makes it possible to visually see sensitivity and smoothness of the Pareto surface or the general space. The Cityplot is also able to visually extract Pareto optimal design families where these families are defined by design space similarity.

We have demonstrated the significance of picking a suitable distance function and suggested some simple distance functions for common problems. We have also shown that changing the plotted set can give slightly different information to a designer. Finally, we have demonstrated the applicability of the technique to real world problems and studies.

The ability of Cityplot to visually gather all this information easily is of import for design-by-shopping paradigms. This allows for a human understanding of the tradeoffs in engineering design and decision making as well as how the tradeoffs relate to designs and design decisions, which can help fix decisions early in the design process or suggest key features of a final architecture. Furthermore, the visualization itself places no constraints on the functions to model and hence is widely applicable to many engineering and decision-making scenarios.

Results of the experiment support the overall hypothesis that Cityplot-VR has potential to support system engineers in typical tasks performed when exploring design alternatives. Users

subjectively preferred Cityplot to a more traditional interactive method (RAVE) by a wide margin on questions of usability and engagement. Users were also able to return better answers when using Cityplot over RAVE with the exception of evaluating how clustered designs were in the criteria (in which case testing is inconclusive). Although it is not possible to state if one method is significantly faster for users to perform analysis with, Cityplot does tend to return better answers with more user confidence and ease of use.

That said, this study has numerous confounding factors. It did not separate the effects of visualization type (3d/2d) from those of the visualization method (Cityplot vs RAVE), nor did it carefully inspect the effects of user choice of interaction interface (keyboard and mouse for the different tools). Another potential limitation is the separation of what is due to the pure visualization method versus its implementation. This is not an easy limitation to address as the ability of an experiment to test a given tool is intricately tied to the implementation of that tool. However, future studies might be able to implement a number of scatterplot-based and Cityplot-based implementations and carefully investigate which features and properties are useful for investigating user performance.

Cityplot itself also has some limitations. Multidimensional scaling was chosen in this application because it closely resembles what would be desired when trying to represent a high-dimensional graph in 2d space and has well-studied history and developed tools. However, it does not guarantee an accurate approximation of the space (Borg and Groenen 2005). A natural follow-up question is: when do other dimension reduction or machine learning techniques more accurately represent the design space or Pareto front?

When plotting a very large number of designs (300+) the plots can become unacceptably crowded. Thus, one possibility would be to group designs into neighborhoods that can be

approximated with a single design to reveal the similar designs within. It might also be possible to automatically detect design families and group the families into representative designs.

We do not foresee the future of Cityplot as a single standalone solution for all design analysis. A simple but important area of development would be integrating Cityplot into a larger optimization and visualization framework.

Regardless of these limitations, this study provides evidence that Cityplot is useful for visualization in engineering design and expresses optimism that 3d VR frameworks can provide a cogent view of decision-criteria relationships and change analysis in engineering design. Code is available upon request.

3 MOMS-CFCS: MULTIPLE OBJECTIVE MINING FOR SIGNIFICANCE—CLUSTER AND FIND CHANGES

MOMS-CFCS is a new approach to understanding the sensitivity of design decisions to criteria changes. The technique is built on the common observation that engineering designs, when plotted by the objectives, will often form bands, clusters or distinct groups. The usual instinct in such cases is to attempt to cluster the bands and treat the bands as separate clusters for further analysis. MOMS-CFCS instead takes the view that such clustering is often going to be the result of the decision→criteria mapping having key sets of interlinked decisions that drive wide gaps in performance—meaning such bands are the natural result of a kind of "high commitment" corresponding to a high gradient. Indeed, choosing which cluster to be in represents a high commitment of the system criteria whereas within a single cluster there is more flexibility for slight adjustments to criteria. Observe this means banding can appear even in continuous design spaces with uniform sampling.

Importantly, this method provides an interpretation and application for observed "banding" behavior that is independent of the actual appearance of banding; the appearance of banding is an incidental effect of the underlying design space rather than a one-off observation; this means that the method is applicable without explicit observation of banding and is a general description of cases that can cause banding to occur. Furthermore, it is a fundamentally nonlinear and parametric approach.

MOMS-CFCs then presents this idea that such banding is a natural feature and attempts to exploit it to mine the original linked decisions that drive the largest commitment—i.e. the largest changes in performance. In doing so, it presents a new application of single linkage clustering and matching to capture and study the banding structure of the design space by

searching for similar changes between two clusters. Essentially, it inverts the thinking behind the appearance of clustering of designs on a scatterplot. We find a set of "close" curves in the objectives upon which designs change, and then find corresponding decision changes that are defined by these curves. In doing so, we exploit geometric structure in the objective space to effectively allow for any order of interaction. This idea is elaborated in sections 3.1 and 3.2.

3.1 Technique Intuition—Banding and Grouping Changes

A common observation in scatterplots of the objective space is the presence of "banding" or "clustering" structure, as seen by the clustering of blue designs in Figure 21.

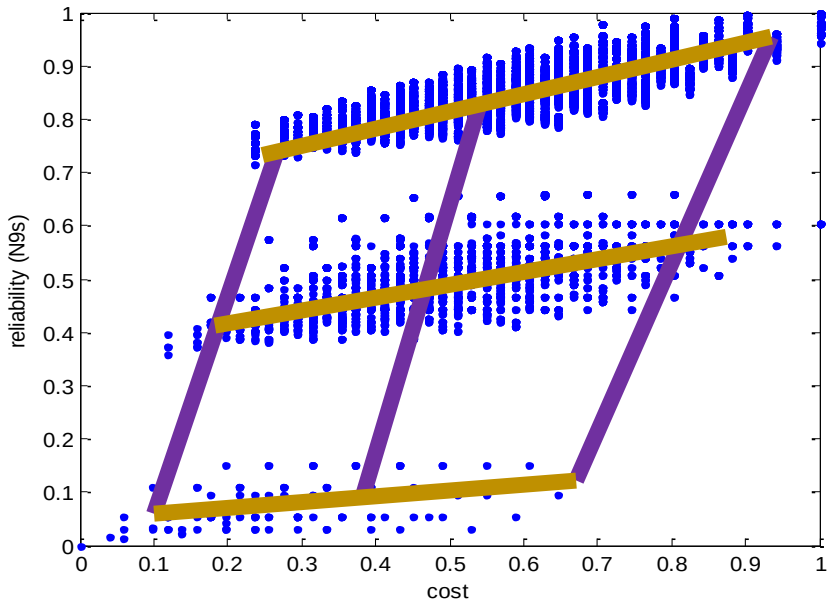


Figure 21: Banding and Matching of a Guidance Navigation and Control reliability study (cf. section 3.3.4). Observe designs from bands (gold lines) form roughly 3 distinct levels of reliability. We seek to move along the directions shown by purple lines to determine changes in decisions that consistently translate designs from one cluster to another.

The first fundamental intuition of this chapter is that the effect of an important set of decisions will produce larger changes in the values of objectives than changes in less important decisions, and this will appear as clusters in the objective space (gold lines in Figure 21). The basis of our technique is to find these clusters, and then look at which decisions are *consistently being altered to move between clusters* (note that this is different from looking at the decisions

that designs in a cluster have in common). In other words, small changes in the most important decisions are driving big changes in the objectives and, as a result, the separation of designs in the objective space after changing these decisions is large (assuming changes in the decision space are performed uniformly). By relating the designs across these gaps, we seek to find what these important changes are based only on the sample of designs from the simulation.

To elaborate a bit, as will be seen in 3.3.4, the clusters in Figure 21 actually correspond to the number of computers and sensors on the system. The vertical axis is reliability and hence the separation in the vertical corresponds roughly to the minimum redundancy of computers and sensors due to failures. As the horizontal axis is cost, adjustments to interconnections and selections of components gives a great deal of cost granularity. The purpose of this method then would be to interpret the large granularity in the risk performance and map it to the number of computers and sensors. Ideally, this can be done nearly automatically. Importantly, in cases where the decisions are more integrated (suppose we also had to deal with heat dissipation) it will group together related decisions, such as the multiple components involved in the heat dissipation system that must happen in response to adding more components.

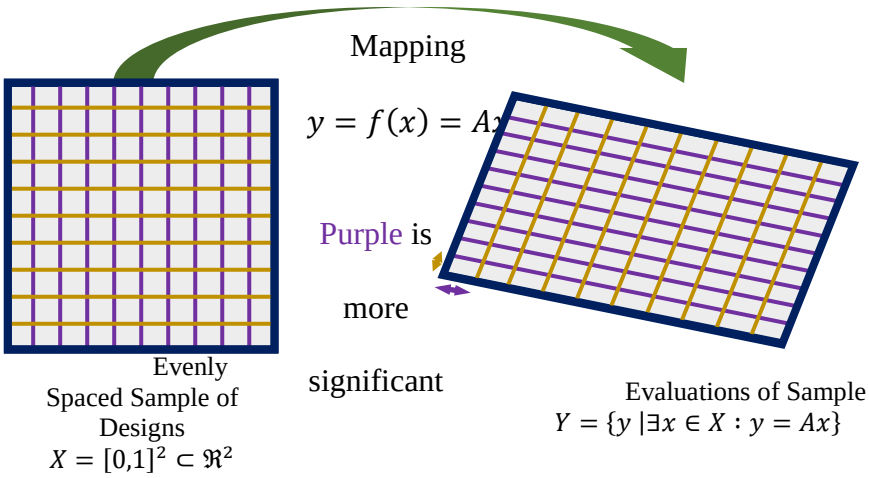


Figure 22: Simple Linear Example of Contraction of Less Important decisions. Observe that the less important decision axis (gold) has less space between designs (intersections of gold and purple lines).

To demonstrate this idea, consider the example shown in Figure 22, which depicts the simplest case: an evenly spaced grid of two design decisions, and a linear mapping between the two design decisions and two objectives. Figure 22 shows that the spacing between parallel purple lines has become much smaller than the spacing between gold lines. This means that if one were to change the purple decision by one unit, it would cause a larger change in the criteria space than a unit change in the gold decision. The idea is to then cluster all the designs along the gold lines in the objective space (the less significant axis in the objectives) and find the changes along the more significant purple axis. If we were to think of selecting a design from the objective space, we would first decide upon a gold group by altering a reference design along the purple axis; we would then choose a location in a gold group by altering the design along the gold axis. By following such a procedure, the most significant decision would be made first.

Following the "most-important-decision-first" procedure, we define *commitment* as how much the objective values change when a design decision is changed. Because the most influential separations/clusters are selected first, this means that the effect from making a particular kind of change bounded from above by the size of the separation of the clusters. Selecting a design within a cluster will be a lower commitment than selecting the cluster. Indeed, that within a cluster, there guaranteed to be another design nearby such that the commitment from one change is bounded above and the commitment between clusters is bounded below. This grouping is sensible as it sets up groups of small changes *regardless of the shape of how the designs move through the space*—that is we maintain the bounds without reference to any model form or geometric assumptions of the mapping or space. Hence, we should be able to find something like the gold lines in Figure 21 or Figure 22 in the case when the effects of design decisions has a nonlinear or complicated geometry in the objective space (beyond just lines).

Figure 21 depicts the expected result in a more realistic and typical example and provides the inspiration for the approach; the problem depicted in Figure 21 is neither linear nor are the decisions real-valued, but the basic intuition applies—there is a clear formation of three clear high-level clusters, and vertical stratification within each cluster. We define commitment to be the same as before, we can move more easily within clusters than between clusters. We analyze the problem that gives rise to Figure 21 in section 3.3.4.

The second fundamental intuition is that some *groups* of decisions are more important than other *groups*. The fact that these are *groups* of decisions is important. Fundamentally, if we were to focus on only the base decisions (or pairs of decisions or triples or higher at the cost of exponential complexity), then we would limit the possible interactions of decisions; this is the basic limitation in most traditional sensitivity analysis(Saltelli et al. 2008). However, many systems do not simply stop at some particular order of interaction. For example, to increase the onboard processing capability of a satellite, we might be tempted to add another central processor core; however, in doing so we must then provide extra power and increase thermal heat dissipation, which will have cascading effects throughout the system resulting in many changed design variables. Hence, simply "adding a computer" becomes a set of changes across coupled subsystems and decision axes. It follows that we must consider multiple simultaneous changes to design variables.

Looking at Figure 21, within each cluster, we can see vertical bands. This suggests that after the "highest-level" decisions that choose a cluster in Figure 21 corresponding to a level of reliability, we can look within each cluster for sub-clusters. We can then repeat the reasoning we used on clusters for sub-clusters and find decision changes that would cause designs to be on one sub-cluster or another, thus finding a "lower-level" set of decisions. As is clear from Figure 21,

these sub-clusters would correspond to values of cost. We can thus repeat the "cluster-and-find-changes" procedure recursively to find a hierarchy of sets of decision that determine how a design is affected by changes.

Observe the primary interest is actually the changes between clusters as opposed to the clusters themselves. We want to take advantage of how the designs are laid out by the decision → objective map instead of just the cluster membership of a design. Another way to see this is to imagine that a cluster is taken, translated and deformed to form the cluster closest to it and we are trying to look at what is changing to control this translation and deformation. This is the image to take out of Figure 21 and Figure 22.

3.2 Technique—Methodology

The methodology has three primary steps. The first step performs cluster analysis to find the clusters to match and corresponds to finding the gold lines in Figure 21. The second step compares designs between clusters and corresponds to finding the purple lines in Figure 21. The third and final step consists of extracting the design decisions from the corresponding designs of the previous step and finds the design changes that produced the effects seen in the sample (i.e., change of cluster). As will be seen, the 2nd step is the most computationally expensive, so exploiting the clustering procedure for the 3rd step can be useful. The following sections detail each high-level step individually.

3.2.1 Cluster Analysis

The first step is to find the clusters. We accomplish this with single-linkage hierarchical-agglomerative clustering. This method is chosen, as, by definition, it upper bounds the maximin inter-design distance within a cluster and lower bounds the intra-design distance between clusters. This follows the idea of *commitment* as defined earlier, where we can negotiate among

elements of the cluster with bounded amounts of change in the objectives. Hence, if we think of the cluster as defining a region of possible objective values, the clusters will be "minimal commitment" decisions for the number of clusters chosen—after choosing the number and size of the clusters, the amount of control we have to pick specific values (by selecting a value and then searching for a close design) for the cluster is bounded.

This clustering technique is a standard method from machine learning and dates as far back as the publication of the asymptotically optimal $O(n^2)$ SLINK algorithm in 1973 (Sibson 1973; Hastie, Tibshirani, and Friedman 2009). To understand the clustering, we start by defining an *inter-cluster distance*:

$$m(C^k, C^l) = \min_{y^k \in C^k, y^l \in C^l} m(y^k, y^l) \quad [18]$$

Where C^k, C^l are two clusters in the objective space and $m(x, y)$ is the distance between designs in the objective space (note: we reserve the use of superscripts to denote distinct objects and subscripts for components of vectors and designs). For this chapter, $m(x, y)$ will be taken as the Euclidean distance of the normalized objective values for simplicity. Other distances are possible and a full discussion of the selection and implications of a distance function is outside the scope of this chapter. Clearly, a different way of quantifying the change behavior of the objectives is going to have an impact on the meaning of sensitivity and the results of this analysis. The clusters are formed by initially treating every design in the sample as its own individual cluster. The closest clusters (as defined by the inter-cluster distance) are then merged iteratively until all the designs are merged into a single overarching cluster. At some point, a human stops the merging process ("cuts the tree") and the clusters are taken as representative of the clustering of the dataset.

Each merge step can be drawn in a tree known as a dendrogram (see figures in Section 3.3 for examples). In the dendrogram, the vertical branches represent sub-clusters to merge,

horizontal lines represent merged sub-clusters at a merge step, and the height of a "branch" represents the inter-cluster distance between the merged sub-clusters. Such a tree gives a picture of how closely related designs are in the objective space. Of note, the inter-cluster distance of the merged designs will lower bound the distances of all designs in different clusters at that merge step and upper bound the maximum-minimum distance of designs within a cluster, so we gain some global insight about the design sample, S, and the objective space, Y, by looking at the dendrogram. Furthermore, the dendrogram can be drawn for any dimensional space, and it is a nice visualization of the structure of the space for objective spaces of high-dimension.

Looking at all the clusters across merge steps defines a *hierarchy* of design change. After a design has been placed into a given cluster with other designs, it will always be grouped with the other designs in its cluster. Similarly, any design will be closer to another design within its cluster than the inter-cluster distance of the next merge step. Plotting the tree represents this hierarchy visually. Clusters closer to the bottom of the figure contain closer designs and are smaller in size (i.e., the number of designs) than clusters at the top of the figure.

Actually finding the clustering is then a matter of deciding when the inter-cluster distance or number of clusters has exceeded a threshold. This can be guided by the aforementioned dendrogram or some other criterion. For this chapter, the number of clusters was found by visual inspection and the merging was stopped when the number of disjoint clusters hit the desired number.

Single-linkage hierarchical clustering is a common technique implemented in mathematical and statistical software packages. To ease programming by retaining the clustering steps, the minimum spanning tree was also found and used to find the designs being merged at a

given step—the minimum spanning tree is equivalent to the merges made between designs in the single linkage clustering procedure (Hastie, Tibshirani, and Friedman 2009).

3.2.2 Inter-cluster matching

Following our intuitive picture from Section 3.1, the clustering step has found the gold lines in Figure 21 as a clustering of designs. Now we need to find the purple links between clusters. Effectively, this is a matching problem: given a pair of adjacent clusters, we attempt to pair each point from one cluster to a point of the other cluster. Intuitively, we want to do so between sequential clusters so we do not consider changes between designs which are not "minimal" in they require more changes than are necessary to alter designs. This step is broken into sub-steps in the following sub-sections.

3.2.3 Perfect and Partial Matchings

The *perfect matching* problem is a classic problem in combinatorial optimization that takes two equally sized disjoint sets and finds pairs of elements (one pair is one element from each set) such that the total distance of all pairs is minimized. Mathematically, this is written:

$$\min_{M:C^k \to C^l} \sum_{y^k \in C^k} \left|\left| y^k - M(y^k) \right|\right|_2^2 \quad [19]$$

Where C^k, C^l are two disjoint clusters that are due to be matched, and M is a bijective function (every point in each cluster is used once and only once) that pairs the elements of the two clusters. Although it is typical to allow any nonnegative cost function to be used between elements of the clusters, for this chapter we will use the squared Euclidean distance to be consistent with the clustering procedure and to best reflect the visual intuition we seek to explore. The perfect matching problem has a classical $O(|C|^3)$ solution in the Hungarian or Kuhn-Munkres algorithm ($|C|$ is the size of either cluster) (Fredman and Tarjan 1987; Kuhn 1955).

Clearly, the constraint that the sets be the same size is too restrictive in our setting. The *partial matching* problem merely requires that every element in the smaller set have a corresponding element in the larger set. This is solved by simply adding dummy elements to the smaller cluster to make it the same size as the larger cluster, solving the resulting perfect matching problem, and removing all matches that used a dummy element.

A related formulation instead requires that: a) all the elements of the larger set be used once and only once, b) the smaller set must have each element used at least once, and c) linkages minimize the total distance. The resulting *bipartite minimum weight edge cover* problem can be solved by taking the partial matching solution and greedily adding the smallest weight edge of the unmatched designs. The perfect matching, partial matching and bipartite minimum weight matching are all equivalent when the cluster sizes are equal. For this thesis, we adopt the bipartite minimum weight edge cover over the partial matching when dealing with differently sized sets as it will tend to generate too many potential pairings rather than too few. It also removes the odd behavior of the partial matching where there are designs in one cluster that have nothing related to them in the other and we cannot reach the other set by changing a decision.

It should be noted that the above formulations require only considering pairs of clusters. The number of clusters is expected to be much greater than 2 in practice. To resolve this problem, we consider each pair of clusters and perform the partial matching sequentially on each pair. If there are K clusters then there are $\frac{K(K-1)}{2}$ pairs of clusters to compute. This is the most expensive step computationally. Ideally, we would only need to perform this step for pairs of adjacent clusters, but we shall see in the next sub-step that the matching is used in finding which clusters are adjacent. Faster methods and approximations exist for the matching, but were not

tested here (see, for example (Drake and Hougardy 2003; Agarwal and Varadarajan 2004; Mehlhorn and Schafer 2002; Duan and Petti 2010)).

3.2.4 Adjacency Checking

Following the "minimal change" intuition, we want to find the single changes in the design decisions, x, which consistently generate changes among clusters. Looking at the purple lines in Figure 21, we can see that the clusters appear to lie in a line. Intuitively, we expect that the changes from one cluster to the next should be sequential if there are multiple "levels" to a set of decision changes. For example, suppose our Guidance Navigation and Control (GNC) system has the option of including any of 1, 2 or 3 computers. It is reasonable to assume that the design with 2 computers will have objective values between that of the design with 1 computer and the design with 3 computers. However, thus far we have not considered the geometry and relative positioning of the clusters and hence have no way to recognize (outside of plotting the designs and looking at the clusters) that the clusters appear to lie on a line. We need to not consider matching the bottom-most cluster with the top-most cluster in Figure 21. This is where the adjacency check comes into place.

We define two clusters to be *adjacent* if there is not a third cluster between them; where "between" means that the line connecting matched points from adjacent clusters do not come too close to any designs in any other cluster; we take "too close" to be the inter-cluster distance of the last merge performed when stopping the clustering procedure and used to form the clusters. This acts as a filter on matching pairs of clusters.

For this chapter, we used the Euclidean distance in the objective space. Hence, we can find the distance between the line connecting points and designs in other clusters as follows: let

$y^k \in C^k, y^l \in C^{l \neq k}$ be designs in two different clusters and $y^c \in C^{c \neq k,l}$ be a design that might be too close. Find the projection onto the line segment connecting the matched designs:

$$t = \frac{(y^c - y^k) \cdot (y^l - y^k)}{\left|\left|y^l - y^k\right|\right|_2^2} \quad [20]$$

Where · is the ordinary dot product and $||\cdot||_2$ is the Euclidean norm. Then calculate:

$$r = \begin{cases} \left|\left|y^c - y^k\right|\right|_2 & \text{if } t \leq 0 \\ \left|\left|y^c - t(y^l - y^k)\right|\right|_2 & \text{if } 0 \leq t \leq 1 \\ \left|\left|y^c - y^l\right|\right|_2 & \text{if } t \geq 1 \end{cases} \quad [21]$$

Where *r* is defined as the distance from the design y^c to the line segment defined by $y^k \rightarrow y^l$. For every pair of clusters, we calculate this value for all matched designs against all designs not in the cluster. As noted earlier, if there is a point y^c such that the calculated distance *r* is less than the distance taken to stop merging clusters, then the clusters are considered non-adjacent. Although there are many computations to perform, it is easy to parallelize the computation.

After performing this step, we know which pairs of matched designs from the previous step occur between adjacent clusters. We can take these pairings and then retain them for the analysis of decision changes described in section 3.2.6. We refer to using these pairs in section 3.2.6 as *matching in the objective space*.

3.2.5 Minimum-Decision Linking

In the typical case where the decision → objective mapping permits the assumption that small changes in the decisions will yield small changes in the objectives, it makes sense to match considering distances in the decision features as opposed to the objectives. Effectively, instead of looking for the smallest changes in the objective space to move between clusters, we look for the smallest change in the decisions. This requires a distance in the decision space, such as an edit distance. This method lacks the clear intuition and generality of finding matching designs in the

objective space but empirically we have found that linking and examining designs in this way can yield results that are more intuitive.

For this, we find the matching solving the following optimization problems:

$$\begin{aligned} &\min_{I^1(x^i \in C^k \leftrightarrow x^j \in C^l)} \sum_{x^i, x^j} d(x^i, x^j) I^1(x^i \leftrightarrow x^j) \;\; subject\ to\ \sum_{x^i} I^1(x^i \leftrightarrow x^j) > 0 \\ &\min_{I^2(x^i \in C^k \leftrightarrow x^j \in C^l)} \sum_{x^i, x^j} d(x^i, x^j) I^2(x^i \leftrightarrow x^j) \;\; subject\ to\ \sum_{x^j} I^2(x^i \leftrightarrow x^j) > 0 \end{aligned} \quad [22]$$

Where $I^p\left(x^i \leftrightarrow x^j\right)$ should be read as the indicator function on the expression "$x^i \in C^k$ is most similar to $x^j \in C^l$". Alternatively, we can refer to this state as "$x^i \in C^k$ is linked to $x^j \in C^l$ for linking problem p." Notice that the roles of the clusters in the indicators are switched, the precedence of the optimization being the partial matching versus minimum weight edge cover distinction from section 3.2.3. The minimum distance linkage between designs $x^i \leftrightarrow_M x^j$ is then defined by thinking of the indicators as defining sets of linked designs and taking the union:

$$x^i \leftrightarrow_M x^j \Leftrightarrow I^1(x^i \leftrightarrow x^j) + I^2(x^i \leftrightarrow x^j) > 0 \quad [23]$$

Observe the sum in the optimization problems is taken over all designs in the two disjoint clusters C^k and C^l and there are no constraints on how many times a given design can be linked to designs in the other cluster. Thus, we allow for true minimum-changes without concern for the geometry of mapping the clusters in the objective space. Only using one optimization in Eq. 5 can lead to the unexpected result where all designs in one cluster link to only a single design in the other set; furthermore, since each optimization only guarantees one cluster has every design linked, we would be left to wonder which set is the important one to fully link and which one we should link against. By using both optimizations in Eq. 5 we guard against such a case and will tend to look at to many possible changes as opposed to too few. Effectively, we seek to correspond points between the clusters using as little change as possible in the original decision space while requiring every point to be used at least once.

We take the minimum distance linkages as defined in this section and pass them to section 3.2.6 as an alternative to the objective space matching from the previous step.

3.2.6 Extracting Important Decisions

Once we have found the corresponding designs from the previous step by either matching in the objective space and retaining the matched designs or finding minimum changes in the decisions space, we seek to find the changes in the decision space (hereafter defined as a *changeset*) between the matched or linked designs. We assume the existence of a function $\Delta(x^k, x^l)$ that can take two designs and find a representation of the changes between designs as a changeset. Because the sample S contains both the decisions made for each design *and* the resulting objective values, we can simply take each pair of matched designs and compare the corresponding decisions regardless of whether the designs are compared in the objective space or the decision space.

For all the lower-level changesets, it is possible to take a shortcut to the matching procedure and gain some understanding by examining the designs that are merged by the single-linkage clustering of section 3.2.1. This works because there are a large number of clusters for the smallest changes and a large number of links between these clusters in the merging procedure. In fact, one motivation to perform the matching or minimum-decision linking is because the single-linkage procedure fails to generate a large sample of corresponding designs at the most-significant change level in the hierarchy (two such links in the example of Figure 21 as there are 3 clusters and hence 2 merges to merge all clusters into one super-cluster). This also provides some information on which changes are considered small by the clustering procedure and ranks the changesets it produces by the distance of the merge in the merge procedure. As

hierarchy provides easy insight into the multi-level nature of a problem, it is used on all the problems demonstrated in section 3.3.

To actually create a changeset practically, it is typical for designs to be represented as some kind of vector. For real variables, this is simply a real vector. For discrete problems, it is typical to use binary vectors for categorical decisions and integer vectors for discrete decisions for which cardinality is meaningful. In such cases, it makes sense to simply subtract the vectors (and hence the use of Δ). This does have a slight problem where the direction of subtraction is not really defined in the matching/linkage, so any results have to be taken with a possible multiplication by ± 1 to be sensible. For our toy problems, a vector representation was readily available. For the GNC problem, a quick script was written to find and present the design changes as structured text (which is why this should be seen as a function instead of strictly a subtraction—a vector representation may not be immediately readily available).

When compared across all matchings between adjacent clusters, these changesets indicate consistent modifications to designs that lead to significant changes in performance. This approach allows for analyzing much higher dimensional designs or objective spaces than one could manually analyze using traditional visualizations. In particular, Example 2 has a 6-dimensional decision space and Example 3 has designs which have a variable number of decision dimensions, but as many as 15. Such high-dimensional problems will confound many visualization techniques, as describing the mapping and significance of design changes would require plotting the number of design dimensions + the number of objective dimensions, which is quite large.

Making sense of the changesets is a challenge. For this chapter, we accumulated common changesets, counted the number of occurrences of the decision changes across all pairs of

adjacent clusters, and plotted the distinct decision change counts as a histogram. Data mining is a potential way to possibly automate analyzing the large number of changed design data being provided by this procedure. To that end, this entire analysis could be seen as a feature engineering procedure on the data to allow other methods to look at *changes between designs* instead of only looking at mapping full designs. This alternative data can be more consistent between designs and uses the geometry of the objective space. This also would naturally extend the ability to move into increasingly high dimensions. Performing data mining with techniques such as association rules (cf. Hastie, Tibshirani and Friedman, 2009) is left as future work.

In this chapter, we shall present the results of when the changesets are derived from matching in the objective space and compare it to when the changesets are derived from minimum-decision linking.

3.3 Example Problems

For ease of understanding, we demonstrate the technique on three synthetic examples. To show applicability and see some pitfalls, we apply the technique to a real-world problem of designing a GNC system.

3.3.1 Circular Contours

Let $X = [0,1]^3 \subset \Re^3$ be the space of input decisions. We discretize X by dividing it into boxes of length 1/9. That is, $S^X = \left\{0, \frac{1}{9}, \frac{2}{9}, \dots, 1\right\}^3$. Observe that x can be thought of as a vector $[x_1, x_2, x_3]$. Let the design to objectives map be:

$$f(x) = \frac{(x_3+\alpha)}{\beta}\begin{bmatrix} sin(\pi(\lambda x_1 + (1-\lambda)x_2) \\ cos(\pi(\lambda x_1 + (1-\lambda)x_2) \end{bmatrix} \quad [24]$$

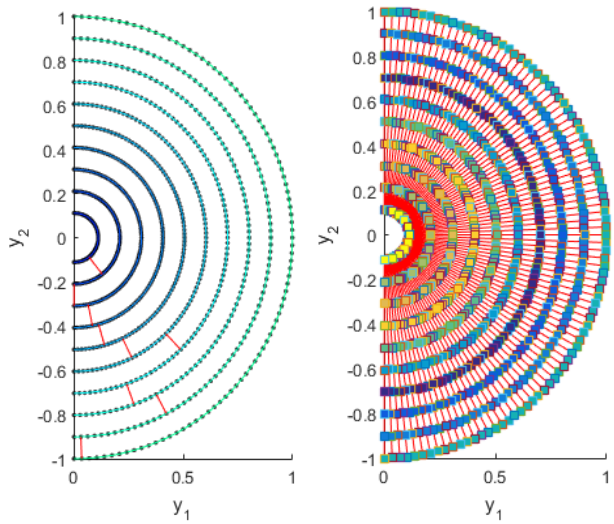


Figure 23: Circular Clustering in Continuous Circle Problem. (left) close distances in the objective space. Colors correspond to length. Red lines will be cut in creating clusters. (right) designs are colored by cluster membership (same cluster=same color), lines indicate matched designs across clusters

This function is nonlinear in x_3 and has a nonlinearity in the convex combination of x_1 and x_2. Intuitively, this is a set of circles with the radius determined by x_3 and the angle dictated by x_1 and x_2. Letting $\lambda = \frac{1}{10}, \alpha = \frac{1}{9}, \beta = \frac{11}{9}$, we get Figure 23 when we apply our technique.

The left side plot of Figure 23 shows the designs within each circle being merged to form circular clusters in the first step of the technique (section 3.2.1). The clustering procedure is stopped before the red lines would link the circular clusters together and we get the colored clusters shown on the right of Figure 23. Designs in different clusters are then mapped between each other as in section 3.2.2. The matched designs are then linked by lines in the right side of Figure 23.

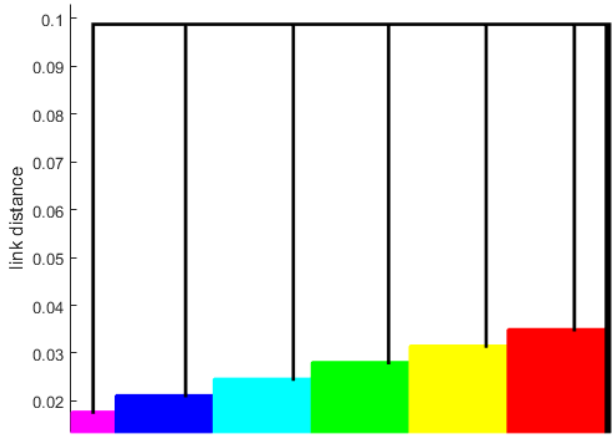


Figure 24: Dendrogram of the clustering for section 3.3.1. The clusters become more separated within each other as the radii increase. Some clusters not shown for clarity

To visualize the distances merged between steps of the clustering procedure, we draw the dendrogram described in section 3.2.1 as Figure 24 (note, the dendrogram is truncated for visibility). As is clear from the figure, the designs are merged within one ring of Figure 23 and then the rings are all merged with roughly equal distances between them. As can be seen from the very large relative jump in distance, the clusters are distinct, meaning that adjacent points within a circular cluster are much closer together than adjacent points across clusters.

The last step in the technique is to find the decision sets, i.e. the changes in decisions that correspond to the merging operations. If we look at edges within circles, the changesets (see Figure 23 left) are consistently $\left[1, -\frac{1}{9}, 0\right]$ for the first 90 merges, where the first 2 components control the angular position and the 3rd controls the radial distance. Observe the vector $\left[1, -\frac{1}{9}, 0\right]$ corresponds to the nullspace of the $[\lambda, (1-\lambda), 0]$ matrix that multiplies the design vector before being fed into a sine/cosine. In other words, the first 90 merges correspond to points that occupy the exact same position in the objective space. The next 890 merges are either $\left[\pm\frac{1}{9}, 0, 0\right]$ or $[\pm\frac{8}{9}, \mp\frac{1}{9}, 0]$. These are all cases that correspond to a change in angle of $\pm\frac{\pi}{9}$, i.e. the merging of adjacent points within a circular cluster. The two are different as there is some ambiguity in

which designs to match when the objective values are the same. Finally, we get $[0,0,\pm\frac{1}{9}]$ for the remaining 9 merges, which also correspond to overlapping points. Observe the vectors can be multiplied by -1 as the linkages are non-directional.

If we look at the changesets from matching the circular clusters (see Figure 23 right), we find that the changesets are $[0,0,\frac{1}{9}]$ or $[1,-\frac{1}{9},\pm\frac{1}{9}]$, which correspond to increasing the radius of the circle. Note that in this latter case, the first 2 components line up with the nullspace vector and thus this corresponds to points that are exactly in the same angle coordinate. Again, the latter matchings result from the ambiguity of matching designs when the designs share the same objective values. Alternatively, if we look at changesets from the minimum-decision linking with the distance in the decision space (defined by the Euclidean distance) then the only feature set in the highest-level changeset is $[0,0,\frac{1}{9}]$ as there is no ambiguity in performing the matching in the decision space.

Overall, these results agree with what we would expect, where the 3rd component places designs on a circular cluster according to the radius.

3.3.2 Radial Contours

To show how changes in the problem or parameters of a problem can change the results of this method, we re-parameterize eq. 6 with $\lambda = 0.5\,, \alpha = 1, \beta = 1$ we get Figure 25 instead, which shall be referred to as the *radial clustering case.*

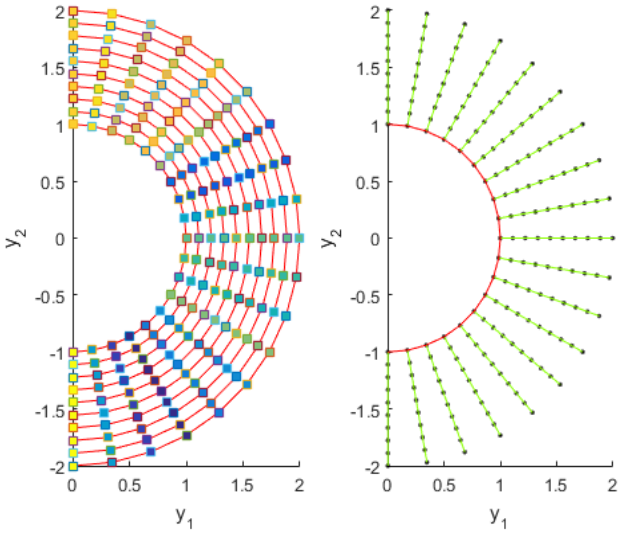


Figure 25: Radial Clustering in Continuous Circle Problem (left) close distances in the objective space. Colors correspond to length. Red lines will be cut in creating clusters. (right) designs are colored by cluster membership (same cluster=same color) lines indicate matched designs across clusters.

In contrast to the parameters given in the previous case, these parameters enlarge the circles, which makes the angular distances grow compared to the radial distances. This is reflected in a different clustering pattern of the designs in Figure 25—the clusters are instead formed along radii of the circles and the designs are compared at the most significant change as the angular changes.

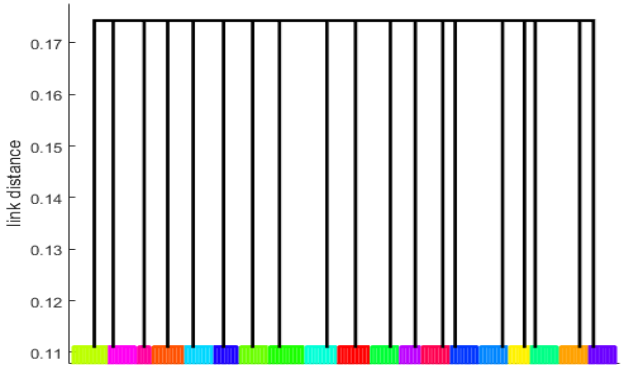


Figure 26:Dendrogram for the clustering for section 3.3.2. Observe the jumps in height correspond to merging the radial clusters together

Figure 26 shows the dendrogram for these new parameters. Whereas Figure 24 had the rings form sequentially, Figure 26 has the radial clusters forming nearly simultaneously at roughly equal distances before the rings are merged with effectively no change in distance.

When matching points within radii clusters (see Figure 25), the changesets for the next 810 links are all $[\frac{1}{9}, -\frac{1}{9}, 0]$, which corresponds to the null space of the new $[\lambda, (1-\lambda), 0]$ vector. The next 100 links are all $\left[0,0,-\frac{1}{9}\right]$ which corresponds to moving along the radial direction. The last 9 links are all $[\pm\frac{1}{9}, 0,0]$ or $\left[0, \pm\frac{1}{9}, 0\right]$, which is the smallest change in the sampling that causes a change in the angle (observe that neither is preferred because λ=1/2).

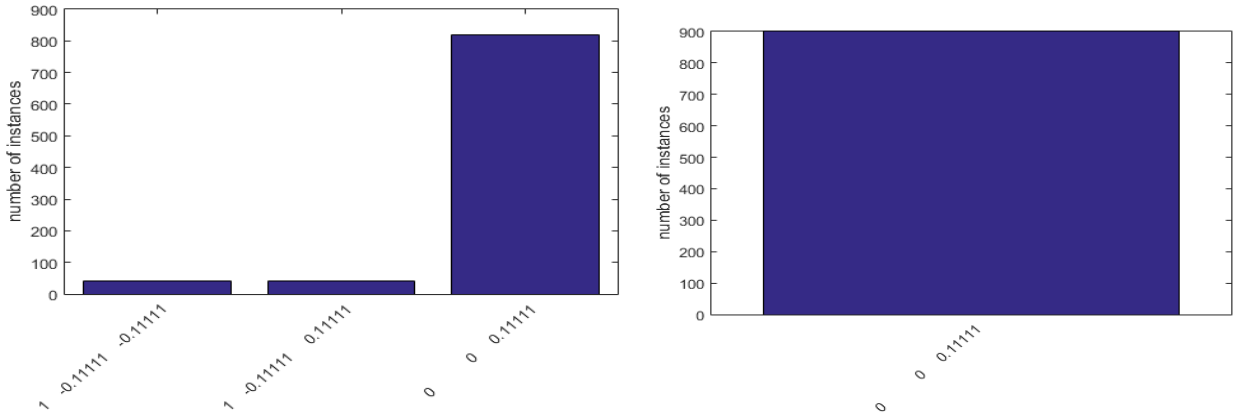


Figure 27: Bar plots of changesets for the matching of circular clusters. (top) matching in the objective space; (bottom) minimum-decision linking. Axis ticks are the changesets Δ($[x_1, x_2, , x_3]$). Observe the design space matching has only one feature set represented, whereas the objective space matching also shows two other effectively equivalent changesets due to ambiguity in matching overlapping points.

If we look at the changesets from matching the radial clusters (see Figure 25 right), the changesets are $\left\{\left[\pm\frac{1}{9}k, \mp\frac{1}{9}(k+1), 0\right] : k \in [-1,9] \subset Z\right\}$ which shows that the difference is that the 1st and 2nd decisions sum to $\pm\frac{1}{9}$, so again the smallest possible change in angle. The range of possible changesets is due to the ambiguity of selecting designs that all share the same objective values; the possible values of k correspond to k additions of the nullspace changeset $[\frac{1}{9}, -\frac{1}{9}, 0]$.

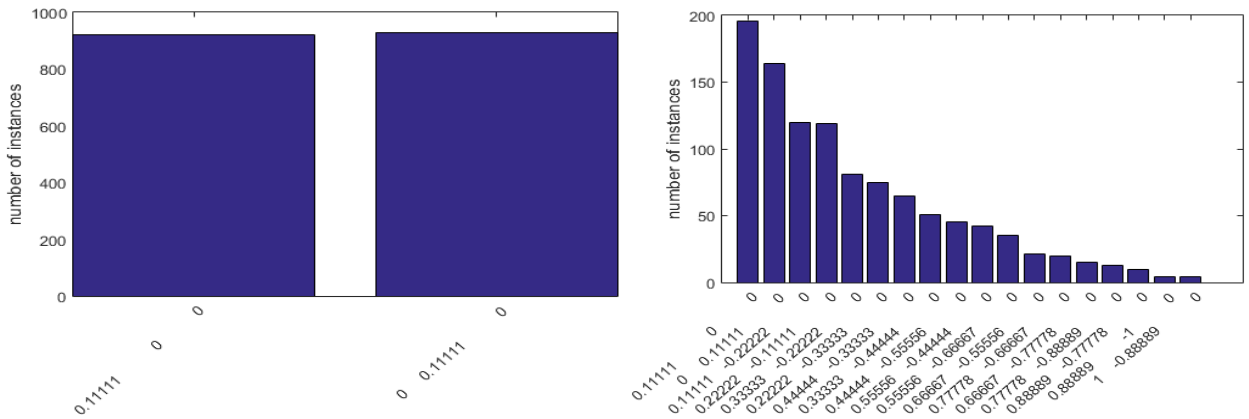


Figure 28 Bar plots of changesets for the matching of radial clusters. (top) matching in the objective space; (bottom) matching in the design space. Axis ticks are the changesets Δ([$x_1, x_2, , x_3$]). Observe that the objective space matching always leaves a difference of [0, 0.1111] in the first 2 components and (significantly) the final component is always 0.

With the links found via the minimum-decision linking we get the result $\{[\pm\frac{1}{9}, 0,0]$ or $[0, \pm\frac{1}{9}, 0]\}$ for the highest-level change, which again is the smallest sampling change to produce a change in the angle. Both appear with roughly equal frequency reflecting the choice of λ=1/2 not preferring over variable over the other to cause an angle change. Minimum decision linking is able to remove the influence of the nullspace because it deducts the addition of such vectors as excess distance when linking designs.

The changesets of this example demonstrate that the method is able to retrieve jumps between design decisions, which indicate tiered performance levels. Of note, the two different parameterizations demonstrate that the technique is able to reflect a new geometry when the problem specification redefines what is a "small change." In both cases, we are able to find changesets which hint at the variable effects of the artificial problem: x_1, x_2 control the angular position of the circle and x_3 controls the radial position, and depending on the parameters of the problem the significance of these components changes.

3.3.3 Discrete Decisions

As an example of a discrete space, let $X = \{0,1\}^6$. For this example, we can exhaustively enumerate this space and avoid sampling issues. Let the design to objectives map be:

$$f(x) = \begin{bmatrix} \frac{1}{6}\sum_{i=1}^{6} x_i \\ \frac{\frac{1}{64}\sum_{i=1}^{6} x_i \, 2^{6-i}}{1+\frac{1}{6}\sum_{i=1}^{6} x_i} \end{bmatrix} \quad [25]$$

The first component is a normalized count and the second component turns the binary *x* (most significant bit first) into an integer before dividing by the first output of *f*. This apparently simple problem is actually sophisticated enough to show some of the challenges related to discrete problems. The value of the first objective is strictly defined by the number of high bits; however, the second objective is influenced by the value of the first objective and values different components of the decision vector differently. Furthermore, the underlying decision space does not transform as uniformly under this function as one might expect; there are many more designs with an intermediate value of the 1st objective than designs with small or large values for this 1st objective. The results of the analysis follow.

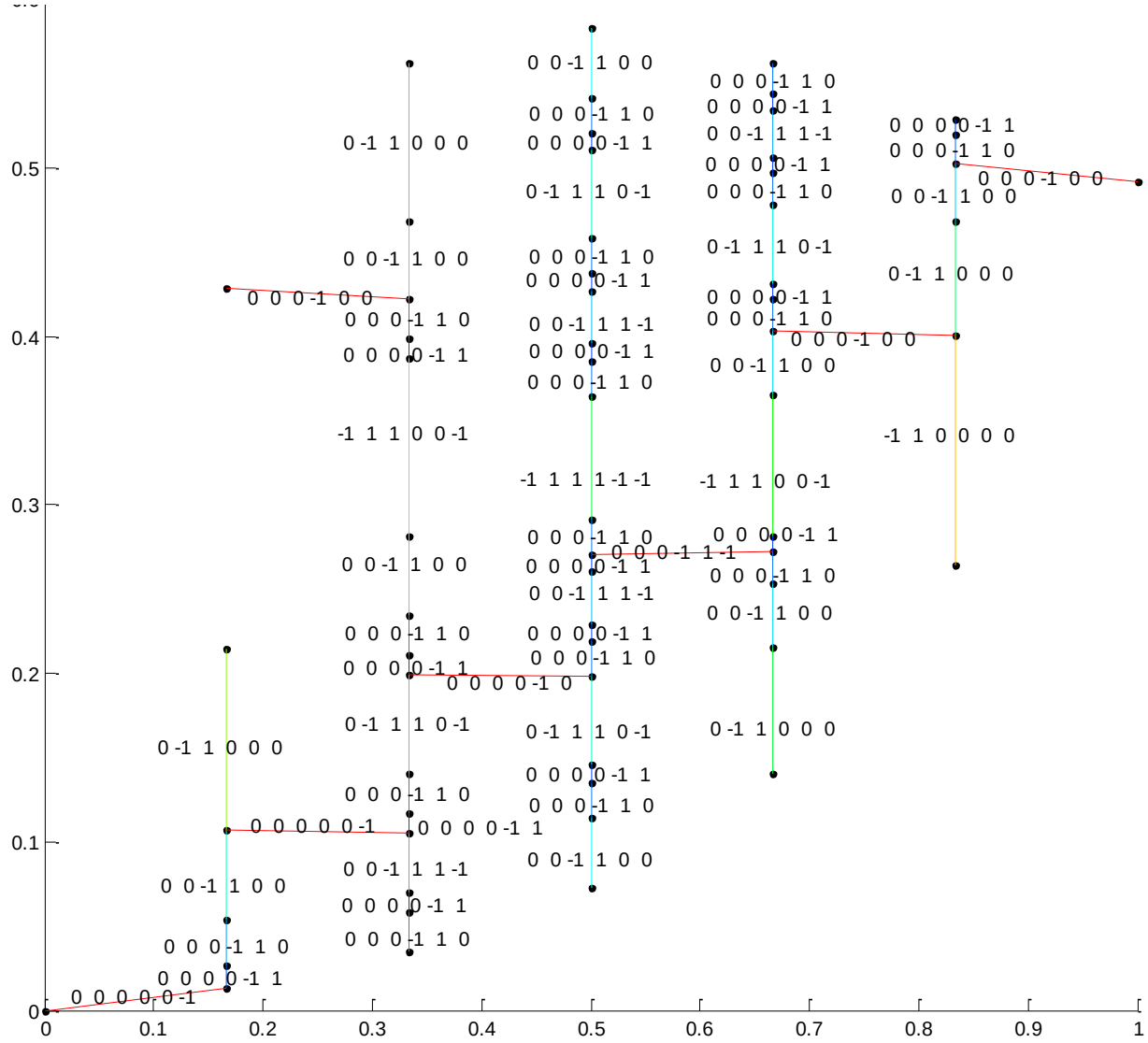


Figure 29: Discrete Synthetic Problem Objective space scatterplot, clustering links and changesets. Line colors Correspond to Sizes of Clusters merged by the link. Warmer colors (blue→green→red) correspond to longer distances. Red lines are cut for forming clusters.

Figure 29 plots distances between close designs that are merged when performing the cluster analysis from Section 3.2.1; warmer colored edges indicate designs that are more distant when merged and labels are the changeset in the decision space that corresponds to a given edge. Under the Euclidean norm used in this problem, the 1st objective is the more important objective for separating designs, and hence it is the number of decisions that are 1 which is the most committal feature for separating designs. It follows that the changesets for designs within a vertical cluster sum to 0 (we flip as many bits high as we do low). Because components of *x* can

only take on values 0 or 1, we do not expect to see a long string of identical changesets causing a monotone increase in the value of an objective. It is, however, possible to see that at similar levels in the single-linkage merges, we observe consistent feature set changes. At the smallest changes (dark blue), the least-significant bit flips value with the second least significant bit. The next smallest change consists of swapping the 2nd and 3rd least significant bits. The next smallest change swaps the roles of all 4 least significant bits to swap the 4th least significant bit and return to the old value of the least significant bit. Watching the least significant changes first and corresponding to changes will reveal that the least significant bits are in the smaller changesets overwhelmingly and the most significant bits will appear later.

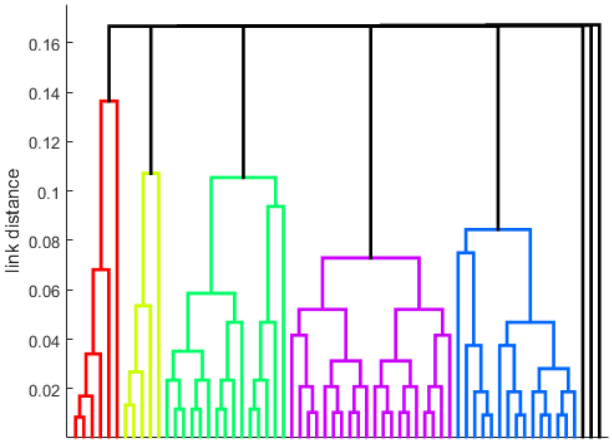


Figure 30: Dendrogram for Discrete Synthetic Problem. Observe more jumps than in the problems in section 3.3.1 or section 3.3.2

Figure 30 is the dendrogram for the discrete problem. Because the design decisions are 0-1, the objective function has fewer levels of values it can use for a given variable or group of variables. Hence, changesets are not as consistent (we simply run out of options too quickly). The fact that the dendrogram has multiple levels and appears more complicated than Figure 24 or Figure 26 is a sign of the greater complexity of the problem (as viewed by the sensitivity to design decisions) as compared to the circular clustering problem in section 3.3.1 or the radial clustering problem in section 3.3.2.

Looking at the highest level formed by breaking the red links in Figure 29 and taking the resulting linked designs as clusters, we can compare the changes between the clusters. We can collect the decision changes by taking one matched design and subtracting from the other. Collecting into a bar chart of how frequently a feature set appears we obtain Figure 32.

As expected based on the most-distant links in the single linkage clustering, the matching finds that the changes between designs are principally just due to changing one bit—flipping just one bit in isolation moves the design to a corresponding design on the next cluster along the horizontal axis. The matching on the objective space produces more unique changesets than the minimum-decision linking. However, looking at these newer changesets we see the appearance of changesets that correspond to very small changes in the location along a cluster (such as $[0,0,0,1,-1]$) being added to the extant single bit change (so the overall change is $[1,0,0,1,-1]$). Both methods seem to recover that the key step is increasing the number of bits of value 1 in the design.

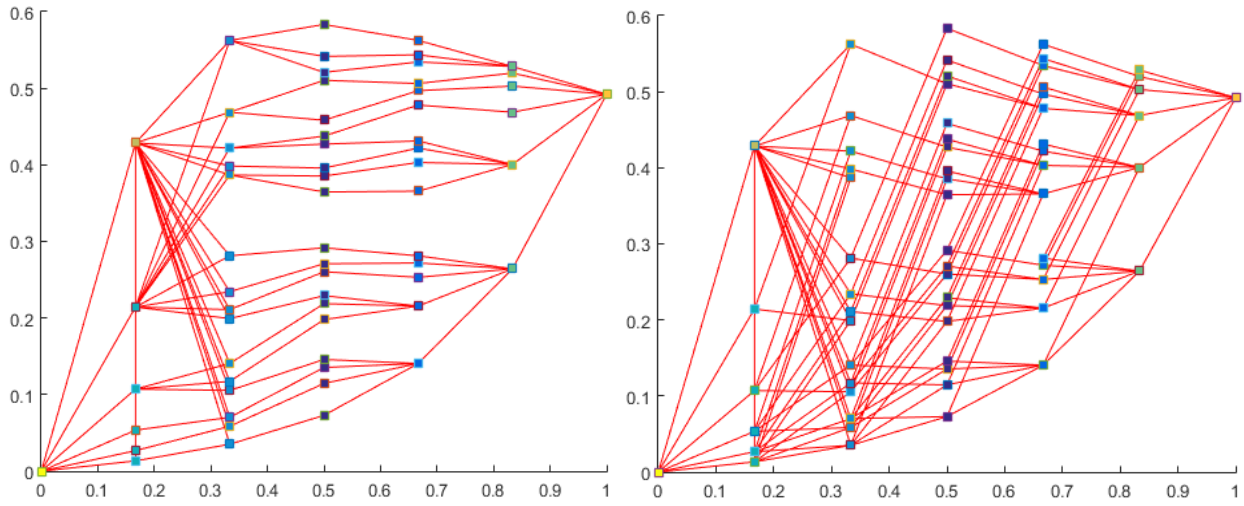


Figure 31: Between-cluster analysis matching of designs as viewed in the objective space (top) matching to minimize objective space changes. (bottom) matching to minimize decision space changes. Design colors indicate cluster (same color=same cluster). The large number of links between the clusters at y_1~0.2 and ~0.3 is due to a singleton cluster in the top left. Although it is tempting to assume it is an offshoot of the cluster below it, when compared in the objective space, it is closer to the cluster at ~0.3 rather than ~0.2

One question is why there is a large number of changesets that appear once in both linkage methods. This is actually a product of the clustering: the top-left point at y ≅ [0.2, 0.4] (yellow in Figure 31) is a singleton cluster, which then is matched against the entire third cluster from the left. This creates a large number of extra changesets. It is tempting to group the singleton with the cluster directly beneath it (second from the left), but the distance between the singleton and the cluster beneath it is larger than the distance between the horizontal separation between clusters so by the commitment definition, it's more related to the cluster to its right than the cluster directly beneath it. This is not a bad way to consider the problem: if we were going to pursue the singleton as a design and then had a design change, it would be less of a change to go to the third cluster than the second. Another consideration is that if the problem were in higher dimensions, then the "shape" of the clusters might not be as intuitively obvious and it is less tempting to group it with the cluster directly beneath it. Of note, the cluster does not contribute to the count of the usual change to move vertically and hence counting the unique changesets is indicative of this change in problem's least-change behavior. As characterizing clusters is outside scope of this work, so is automatic detection of such unusual clusters.

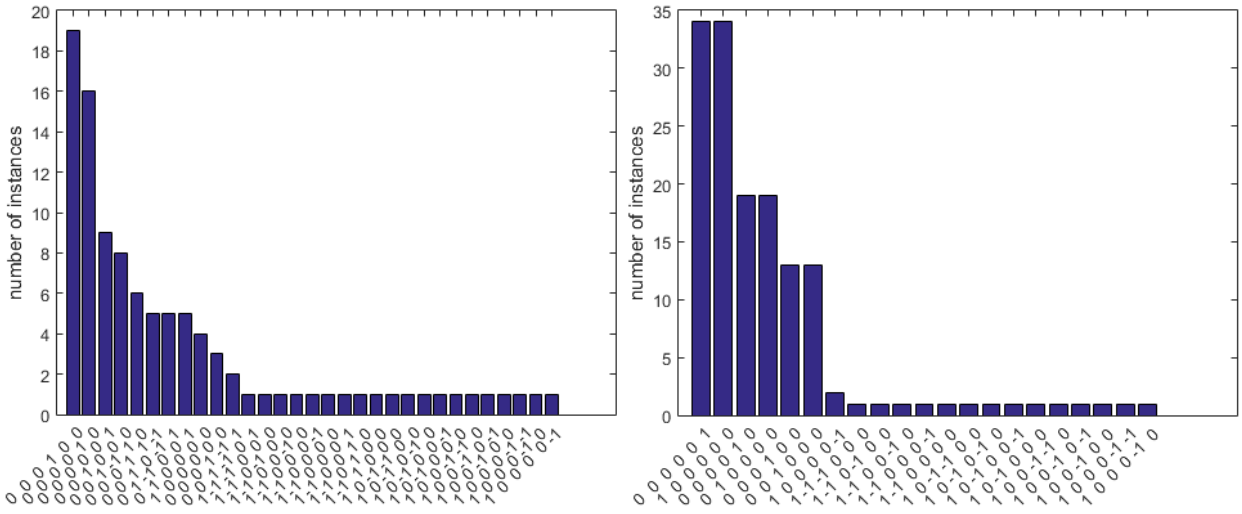


Figure 32: Bar plots of changes for discrete toy problem. (top) matching on objective space (bottom) minimum-decision linking. Observe high concentration of changes to the extreme bits in either case, with an emphasis on the low order bits

The changesets in the clustering process and the matching show that the analysis is capable of working on discrete problems and the information on the significant variables can be recovered in a human readable way. We have also seen what happens if the clustering criterion finds designs which break the pattern of other designs and produces and unexpected changeset. as in the continuous problems, minimum-distance linkage finds simpler changesets with fewer changed designs by altering the matching slightly but the objective space matching is able to recover the important decisions given that it will (relative to the minimum-distance linkage) translate the matching along lower-order changes.

3.3.4 Guidance Navigation and Control

In 2007, Dominguez-Garcia et al. (2007) conducted a study of different architectures of the Guidance, Navigation and Control System on historic NASA spacecraft. A model was constructed to compare alternative designs in terms of reliability and weight. A full enumeration of all designs that are fully connected between computers and sensors was found and simulated. A full description of the model and dataset is outside the scope of this chapter—we only use it to demonstrate our technique under real-world constraints and applications. Of note, the designs are

composed of the following discrete decisions: selecting the number of computers, selecting the number of sensors, picking the selected number of computers and sensors from a catalog, and defining the connections between chosen sensors and computers. This sequence of decisions can be difficult to analyze with traditional analysis. For instance, it can be challenging to encode that it is not possible to connect a computer that does not exist. For the purposes of the analysis, this is treated as a black-box with a few extra subroutines (finding changesets and quantifying dissimilarity between designs in the decision space).

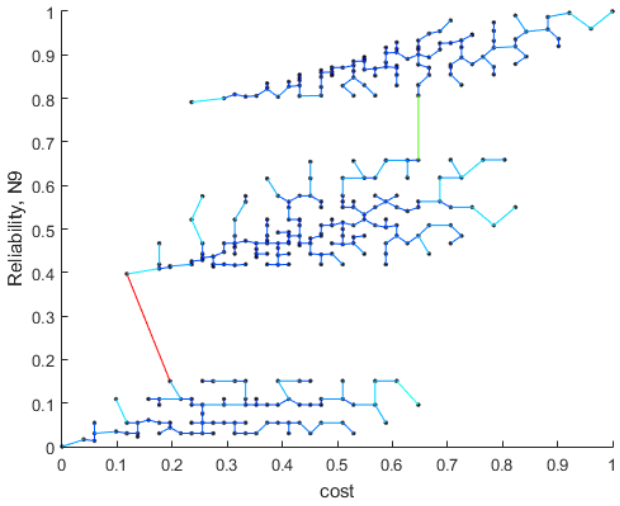


Figure 33: Cluster Formation for the GNC Problem. Observe that the reliability (normalized number of 9's) forms clusters with small-scale changes in normalized cost providing finer control

Figure 33 depicts the formation of clusters for the GNC problem with a full enumeration of the fully-connected architectures only, which are the important architectures when connections are cheap compared to sensors and computers. Observe that whereas there is a large spread in weight, the reliability makes very significant and noticeable jumps in performance. Clearly, understanding what drives these jumps is of import for designing robust GNC systems. The image indicates that there is significantly more flexibility in adjusting mass (cost) than adjusting reliability.

For comparing designs in the decision space, we defined the distance function as in section 2.7.4 (reproduced here for convenience):

$$\begin{aligned} M(\vec{x}_1, \vec{x}_2) = M_m \left(\left| \overrightarrow{x_1}^{\#S} - \overrightarrow{x_2}^{\#S} \right| + \left| \overrightarrow{x_1}^{\#C} - \overrightarrow{x_2}^{\#C} \right| \right) + \\ M_l \left(\Sigma_i I \left(\overrightarrow{x_1}^{C_i} \neq \overrightarrow{x_2}^{C_i} \right) + \Sigma_j I \left(\overrightarrow{x_1}^{S_j} \neq \overrightarrow{x_2}^{S_j} \right) \right) + \\ \Sigma_{(i,j)} \left| \overrightarrow{x_1}^{C_i \wedge S_j} - \overrightarrow{x_2}^{C_i \wedge S_j} \right| \end{aligned} \quad [14]$$

This distance function defines a hierarchy of decisions in terms of their importance: the number of components is more important than choice of components, which is more important than the choice of connections. More precisely, the distance function first looks at the difference in number of computers and sensors. If the two architectures have the same number of computers or sensors, it then attempts to find how many computers or sensors must be swapped out to change one architecture into the other. Finally, if the architectures are identical except for the connections, it looks to count the changes in how computers and sensors are connected. Computing the distance in this manner corresponds to being able to add computers or sensors, swap out computers or sensors and being able to reconnect computers or sensors and each decision down the tree accounting for all possible changes due to the decision to eliminate or add a system. Each action is assigned weight so that the addition is always more expensive than swapping which is always more expensive than reconnecting. The changesets use this space of actions to turn one architecture into another by simply finding what these actions should be. For the designs sampled in this chapter, the assignment component term will always be 0 as the designs connect all computers with all sensors.

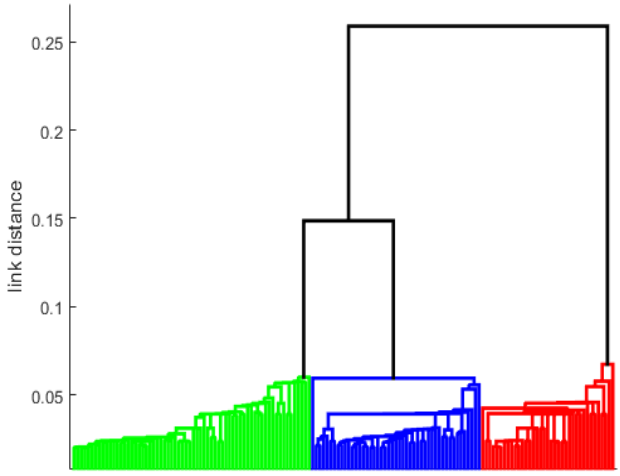


Figure 34: Dendrogram for the GNC problem. Observe distinct clustering into 3 groups

Figure 34 is the dendrogram for the GNC problem. As is evident, there is a large jump in performance between the 3 groups of designs. Comparing values of these clusters can quickly find that the difference is in the reliability as seen in Figure 33.

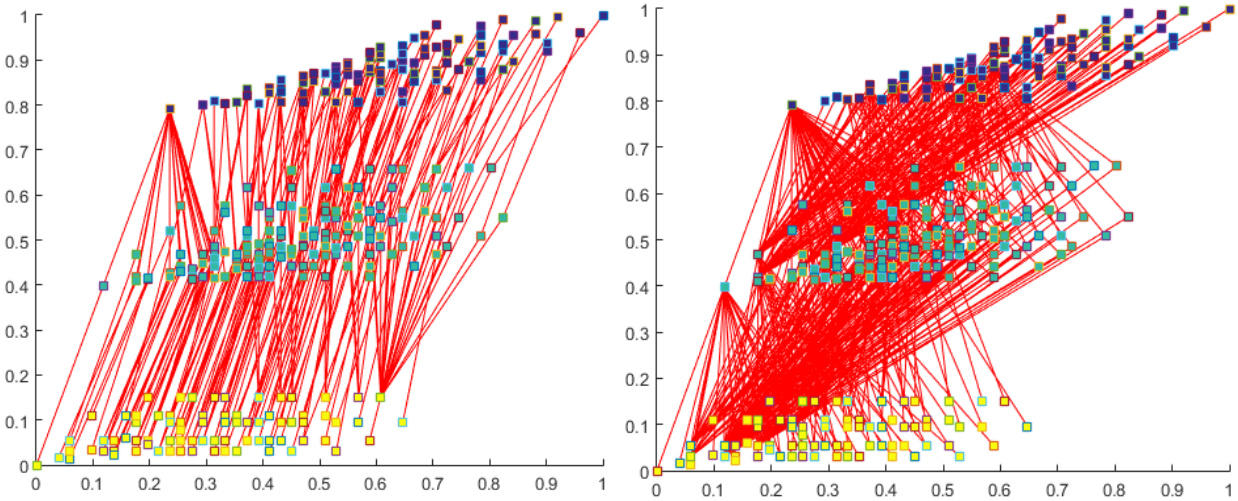


Figure 35 Matching for the GNC problem (top) matching based on objective space. (bottom) matching based on decision space.

After breaking the problem into clusters of similar reliability, we can look to match designs to move between the clusters.

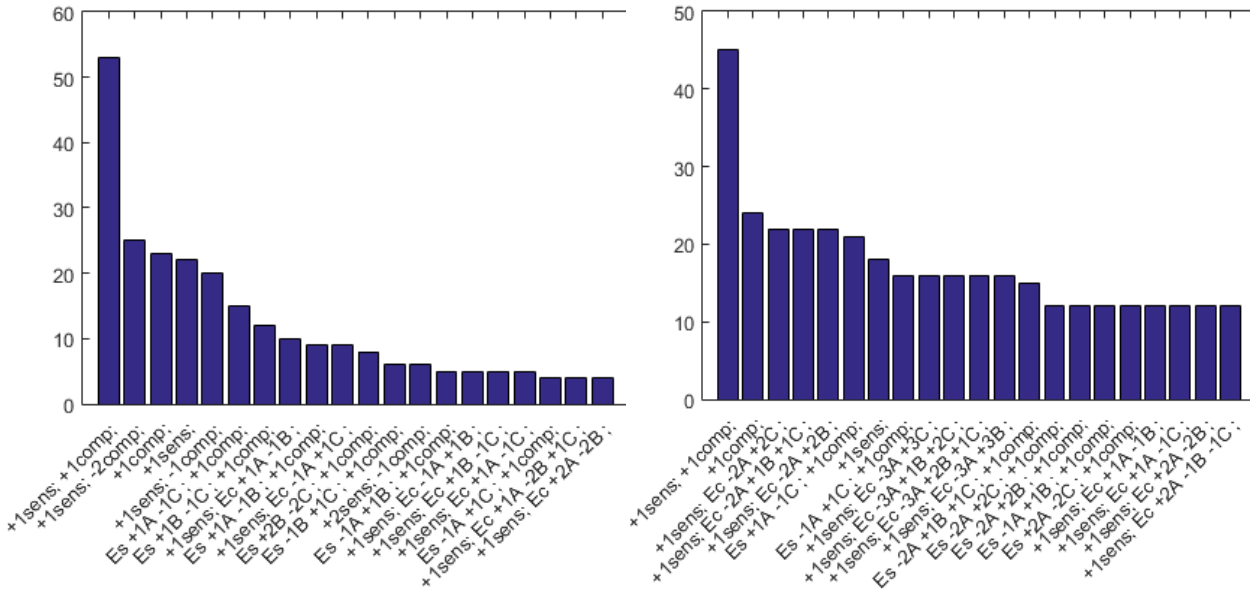


Figure 36: most frequent changesets found for the GNC problem (not a complete list). (top) objective space matching (bottom) minimum-decision linkage. To read the changesets on the x horizontal axis: "Es" stands for exchange sensors, "Ec" is exchange computers, "+" means add a computer or sensor, "-" means remove a computer or sensor, and "+comp" or "+sens" means changing the number of computers or sensors. Following the distance metric in the decision space, if the number of computers changes, the added computer and connections are considered unchanged. Observe a tendency to add computers and/or sensors

Following the procedure described before, we create the bar plots to count changesets, and we find that the number of sensors and computers are an extremely significant feature for the problem. Both the matching in the objective space and the minimum-decision linking find that the most common change is adding a sensor and a computer. Following the observation that the number of sensors and computers seems to be the key feature, we combine the matchings with the paired designs and run some simple guesses on what couple well-categorize the behavior of the designs at this level. We considered the average number of computers and sensors, the total number of computers and sensors and the minimum number of computers and sensors. Simply counting the occurrences in a slightly separate analysis (guessing and checking the composite features based on the design and the corresponding change) in the dataset, we find that the minimum of number of computers and sensors changes in every single decision pair. This corresponds to the expected result from classical reliability theory that the minimum cut set of

smallest cardinality in a system tends to dictate reliability (Hanuschak et al. 2009), assuming that components have similar reliabilities.

3.4 Conclusions

This chapter has presented an analysis technique that utilizes clustering and matching to find the changes in designs that lead to significant changes in multi-objective performance values. The clustering is chosen to conform to a definition of such that the distance between designs on the curve is bounded within the cluster and objective values of designs outside the cluster are bounded away from those of designs within a cluster. Adjacent clusters represent stepped changes to a set of decisions. Finally, either the designs are matched across clusters to minimize the distance in the objective space between corresponding designs, or the designs are linked to minimize some measure of change in the decisions. These matched designs are then used to find changesets between designs that correspond to the clusters chosen by the problem. The analysis has been demonstrated on a number of toy examples to illustrate the technique. A real-world guidance navigation and control problem has been briefly examined to show applicability to real world data analysis.

The primary limitation of the method is that it is not immediately clear how to properly extract and make sense of the change sets. For the purposes of this chapter, we simply extracted complete change sets and examined them for abnormally high frequencies of certain common elements. There is also the related question of what precisely the changesets mean. Matching purely on the objectives is the most intuitively sensible and practically justifiable—the amount of objective space movement for a design+changeset is limited by the clustering constraint. However, as we have seen, this may not always lead to the most intuitive lists of changes. Minimum-decision linking gives more intuitive, simpler and more consistent changesets in some

cases, but it loses the nice properties that changes would have defined bounds and substitutes the intuition that the changesets should be minimal in some sense in the decision space. Selecting between these two or adopting a better method of mapping between clusters remains an open question in general. The method chosen will have an impact on the interpretation of the changesets. For this chapter, we simply accumulated unique changesets and counted the number of occurrences in pairing up all adjacent clusters. More sophisticated data mining analysis, such as association rule mining, may be able to better summarize the changesets, adjust for noise or incorporate more information such as the location of the changeset and constraints on the designs being compared. In this light, this method may be a pre-filter stage in a general analysis of sensitivity. Such data mining procedures remains as future work.

One significant limitation is a high sensitivity to noise (e.g., due to random sampling or stochastic simulations). Single linkage clustering can be highly sensitive to the sample used for analysis. If single linkage clustering becomes confused, then the changesets will be similarly poor. Additionally, the estimation of commitment can become poorly represented as it becomes possible to fail to link designs that should be connected by a missing third design. In the continuous case, it is also possible to under-sample the space and fail to observe certain features (for instance, if the circular example in section 3.3.1 was only sampled with steps of ½, the technique would not detect the directions which cause no change). Similarly, discrete domains may need to be sampled heavily (full factorial or close) to perform truly accurate matching which severely hurts scalability. The discretization of continuous space for analysis however is a complex subject in its own right, well outside the scope of this discussion, and commonly appears in numeric analysis. We believe one solution to this challenge is to penalize the linkage

procedure to prefer models of certain forms and interpolate between missing designs, however, the exact forms that we should initially assume reasonable is an open question.

Despite limitations, the technique can be used to create a sequence of sets of decisions that lead to very different changes in performance in heavily nonlinear, interacting problems and design spaces. It can also be used in cases when design decisions are both continuous and discrete. We believe such analysis shows promise as a new way of thinking about clustering and change analysis with application in engineering design.

4 MAPSA: MEAN ANALYSIS PLANE AND SURROGATE ANALYSIS

Much of generating high level "understanding" or "rules of thumb" of the tradespace can be thought of a kind of compression problem. As a conjecture, if the Pareto frontier can be adequately summarized in a few key numbers, it may be possible to understand the Pareto frontier without large amounts of exhaustive visualization and analysis. MAPSA directly characterizes the shape of the Pareto Frontier with a mathematically understandable surrogate model intended to ease the understanding and computation of the tradeoff rates between conflicting objectives. In doing so, the surrogate has defined bounds and applicability guarantees and, when sufficiently sparse, can quickly summarize the overall shape of the Pareto frontier. Additionally, as it serves as an interpolation, calculations and optimizations can be done on the surrogate. In this chapter we show that a reasonably general representation method for Pareto frontiers can be found that allows for the calculation of the shape, and we compare a few models for the surrogate itself.

4.1 Technique Overview

The technique is comprised of three primary steps:

1. Find the hyperplane which best approximates the Pareto frontier on average, and define a coordinate system based on that plane.
2. Surrogate analysis is performed with a nonlinear generalized linear model to approximate the shape of the Pareto frontier relative to the plane found in the previous step (i.e., we approximate the residuals).
3. The coefficients of the surrogate analysis are presented with appropriate text or graphics and the tradeoffs between objectives are calculated based on the surrogate and visualized.

The mean plane provides a method to parameterize the locations on the Pareto surface and approximates the overall orientation of the Pareto surface. The surrogate analysis then attempts to quantifiably characterize the shape of the residuals by considering it as the superposition of relatively easy to understand shapes (sinusoidal functions).

Overall, the method accomplishes the following:

1. Provides a means of parameterizing Pareto-optimal designs in a *d*-1 dimensional representation (mean plane position)
2. Quantifies the overall curvature as compared to a reference function
3. Quantifies multimodal Pareto front behaviors
4. Provides a surrogate model to interpolate the Pareto surface, which can be further analyzed or used.
5. Provides a means to "compress" the information content in higher dimensions allowing for descriptions of the Pareto frontier shape when direct visualization fails.

The algorithmic steps are elaborated individually in the following sections.

4.2 Model Theory and Calculation

The overall method can be summarized in a single equation (hereafter referred to as the MAPSA equation) which will be explained over the next several sections and comprises a primary contribution of this chapter.

$$\vec{y} = T\vec{z} + r(\vec{z})\,\vec{n} + \bar{y} \qquad [26]$$

Where the transformation matrix T, normal vector $\vec{n}$ and coordinate $\vec{z}$ will be explained in 4.2.1 and the surrogate function $r(z)$ will be explained in section 4.2.2. The basic principle is that a coordinate transformation will induce a hyperplane with an associated coordinate system.

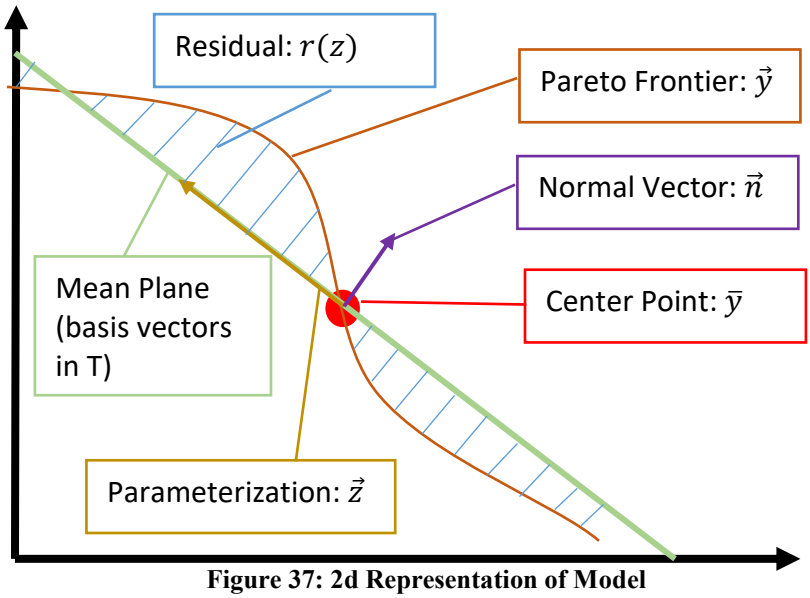


Figure 37: 2d Representation of Model

The hyperplane will be chosen to include the most basic information on the behavior of the Pareto Frontier (e.g., the global trends between objectives). The residual then includes higher order information on the Pareto frontier and provides an accurate representation for further analysis.

4.2.1 Mean Plane Analysis

4.2.1.1 Ensuring a Defined Mapping and Parameterization

Given that we wish to create a model of the form given in the MAPSA equation, we need to guarantee that $r(z)$ will, in fact, be a function—any given coordinate z will have one and exactly one value for interpolation. Fortunately, because *P* is a ParetoPareto front, we can show that function values will be positive and that the ParetoPareto front can be always represented by a mapping from a *d-1* subspace with a $\Re^{d-1} \rightarrow \Re$ function.

Lemma: Any line defined by a point $\vec{y} \in P$ on the Pareto surface and a strictly positive vector $\vec{n} > 0$ (all components are positive) will only intersect the ParetoPareto frontier once (at $\vec{y}$).

Proof: Assume that a line defined by $\vec{y}$ and $\vec{n} > 0$ intersects P at a second point $\overrightarrow{y'} \in P$. By the Pareto-optimality definition, for all Pareto-optimal pairs $\vec{y}, y'$ we have the existence of elementary basis vectors $\overrightarrow{e_i}, \overrightarrow{e_j}$ corresponding to non-dominated objectives such that $(\vec{y} - \vec{y}') \cdot e_i < 0 \iff \vec{y}_i < \vec{y}_i'$ and $(\vec{y} - \vec{y}') \cdot e_j > 0 \iff \vec{y}_j > \vec{y}_j'$. For the $\vec{y}, \vec{y}'$ to be collinear along some $\vec{n} \geq 0 \Leftrightarrow n_i \geq 0 \; \forall i$ it follows that $\vec{y} - \vec{y}' = nt$ for some $t \in \Re$. But then $(\vec{y} - \vec{y}') \cdot e_i = n_i t < 0$ contradicts $(\vec{y} - \vec{y}') \cdot e_j = n_j t > 0$ because the components of $\vec{n}$ are all positive and hence the product with t cannot be both positive and negative. ■

Theorem: We can define a subspace $\mathcal{Z}$ where every point $\vec{y} \in P$ on the Pareto surface has a corresponding point $\vec{z} \in \mathcal{Z}$ in the subspace.

Proof: Fix some $\vec{n}$ ($\vec{n} = \vec{1}$ being a trivial example). From there it is possible to apply a Gram-Schmidt process to create an orthonormal basis from $\vec{n}$; let this basis be denoted by $\{\vec{b}_i\}_{i=1}^{d-1} \cup \{\vec{n}\}$. Because this is a complete orthonormal basis, *any* $\vec{y} \in \Re^d$ can be mapped onto this new basis. In particular, define

$$\begin{bmatrix} z_i \\ r \end{bmatrix} = \begin{bmatrix} \vec{b}_i \\ \vec{n} \end{bmatrix} \vec{y} = \mathrm{T}^{\mathrm{T}} \vec{y} \qquad [27]$$

Where the $\vec{b}_i$ and $\vec{n}$ are row vectors stacked vertically. Because we can perform this exercise with every point on the Pareto surface, $z_i = \vec{z}$ defines a subspace for any point on the surface.

Theorem: By creating an orthonormal basis from a fixed $\vec{n} > 0$, we can obtain a well-defined function mapping vectors in $\mathcal{Z} = \{\mathrm{T}^{\mathrm{T}} \vec{y} : \vec{y} \in P\}$ to residuals $\{r = \vec{n} \cdot \vec{y} : \vec{y} \in P\}$. This mapping is unique in the sense that for any $\vec{z} \in \mathcal{Z}$ we have one and only one $\vec{y} \in P$ that corresponds to it. (And hence a bijective parameterization)

Proof: By the previous lemma and theorem, we have some $\vec{z}$ for any $\vec{y}$ and this can be constructed via some $\vec{n} > 0$. To see that the function $r(z)$ is well defined due to the lemma, assume for contradiction we have $\vec{y} \neq y'$ both on the Pareto surface but satisfying $T^T\vec{y} = T^T y'$so that both will share the same $\vec{z}$. Then if we let $r = \vec{n} \cdot \vec{y}$ and $r' = \vec{n} \cdot y'$ then we see: $n(r' - r) + \vec{y} = y'$ and hence $nt + \vec{y} = y'$, where $n > 0$ is the normal vector and $t = r' - r$, which contradicts the non-collinearity of the Pareto frontier in positive directions. Thus, the mapping $r(z)$ is well-defined and $r(z), \mathcal{Z}$ parameterizes the Pareto frontier■

This leads to some interesting corollaries For example, a common approach to try and interpolate a Pareto frontier in two dimensions is to pick one of the objectives arbitrarily and then use best fit methods to estimate the other objective. However, there has been a question if this works in general or only on the examples seen by the community. This shows that it is in general not possible for the Pareto frontier to 'wrap in' on itself and create a situation where the surrogate is forced to interpolate two separate values. This theorem proves that not only does choosing an objective as a dependent variable works in general (simply allow $\vec{n} = [0, \dots, 0, 1]^T$) but that approximation strategies will work for any positive vector.

Importantly, we have a representation of the Pareto frontier in d-1 variables ($\vec{z}$) and it applies to all Pareto frontiers. Hence we can represent Pareto Frontiers in d-1 independent dimensions. Now we show that this is a tight bound—the space of general Pareto frontiers will require at-least d-1 dimensions to represent.

Observation: To represent the general space of all d-dimensional Pareto Frontiers requires at least d-1 degrees of freedom

Proof: Let $\vec{y}_j \in \Re^{d-1}$ be any bounded collection of points in *d-1* dimensional space. Let the space be bounded by the box defined by:

$$L_i = \max_j(\vec{y}_j^i) \ \ for\ each\ criterion\ i \quad [28]$$

Then we can define the $\Re^d$ space in which the original collection is all Pareto-optimal by:

$$[y_j']_i = \begin{cases} [\vec{y}_j]_i \ \forall i \leq d-1 \\ |[L_i]_i - \vec{y}_j| \end{cases} \quad [29]$$

That is, we stack the original points as the first d-1 coordinates and then for the final component, find the distance to the outermost corner of the bounding box. It follows that all $\vec{y}_j{}'$ are Pareto optimal because for any pair $\vec{y}_1', \vec{y}_2'$ either: (1) $\vec{y}_1$ does not dominate $\vec{y}_2$ which trivially implies $\vec{y}_1'$ does not dominate $\vec{y}_2'$ or (2) $\vec{y}_1$ does dominate $\vec{y}_2$, in which case $L_i - \vec{y}_1^i > L_i - \vec{y}_2^i \ \forall i$ which implies the norms (take any p-norm) satisfy $|[L_i]_i - \vec{y}_1| > |[L_i]_i - \vec{y}_2|$ which means the final component of $\vec{y}_2'$ is preferred over $\vec{y}_1'$ (remember, we are attempting to minimize) which in turn implies $\vec{y}_1'$ does not dominate $\vec{y}_2'$

Hence, any representation of Pareto Frontiers in d-space must be able to represent general d-1 space. If, for example, we had a representation in d-2 variables, $z_{-1} \in R$ we could create sets $z = \begin{cases} z_{-1}^i \ \forall i \leq d-1 \\ 0 \end{cases}$ and $z' = \begin{cases} z_{-1}^i \ \forall i \leq d-1 \\ 1 \end{cases}$ then re-apply earlier constructions to get distinct Pareto frontiers for both that share the same non-unique representation.■

What this implies is that we are efficiently representing the entire space of Pareto-frontiers with our parameterization as we have constrained ourselves to exactly the number of dimensions one would need to specify a location on the Pareto-frontier in general. Further improvements would require further constraints on the kinds of Pareto frontiers we are willing to represent as the general space of Pareto frontiers is at least d-1 and we have a d-1 representation.

Unfortunately, this does not necessarily imply any parsimony in parameters for the representation of the residual function $r(z)$, nor does it instill any useful insights how to achieve such parsimony.

4.2.1.2 PCA-Like Calculation of Normal Vector

As a first step to simplify this section, begin by finding the mean and centering the dataset:

$$\bar{y} = \frac{\vec{1}Y}{N}, \hat{Y} = Y - \vec{1}\bar{y} \quad [30]$$

A first observation is that we want our transformation to be orthonormal as an easy way to guarantee that we will be able to apply the transformation.

Intuitively, we might want to capture as much information in the mean plane as possible. Alternatively, this means having as little overall deviation from the mean plane as possible (hereafter referred to as the mean plan equation):

$$\min_{\vec{n}>0,|n|=1,T:T^TT=I,T^Tn=\vec{0}} \left|\hat{Y}n\right|^2 \quad [31]$$

Observe that we have inserted the additional constraint that the normal vector $\vec{n}$ be positive in all components ($\vec{n} > 0$) from section 4.2.1.1.

As we will use the standard L2 norm for vectors and matrices, this has a straightforward solution that corresponds well with Principle Component Analysis and will hence yield.

This can be seen by taking the singular value decomposition

$$\hat{Y} = U\Sigma V^T \quad [32]$$

Where U is the orthonormal matrix of left singular vectors, Σ is a diagonal matrix of decreasing positive eigenvalues and V is the orthonormal matrix of right singular vectors. Let

$$V\begin{bmatrix}0\\0\\...\\1\end{bmatrix} = V_{-1} \quad [33]$$

be the least significant eigenvector. In the special case where $V_{-1} > 0$ then we can simply let $n = V_{-1}\ and\ T = V \backslash V_{-1}$ and we see that the optimization problem has simply reduced itself to Principle Component Analysis with the mean plane normal vector $\vec{n}$ taking the place of the least significant eigenvector. This special case tends to be the “normal” case when the Pareto frontier spans more space in the positive orthant than its own overall thickness, which would

include all Pareto frontiers between two objectives and the examples in section 4.5. As an example of where this fails, we can consider the following arc of a hemisphere in 3d. The thin width of the arc causes the PCA to optimize in a direction that doesn't point directly away from the origin:

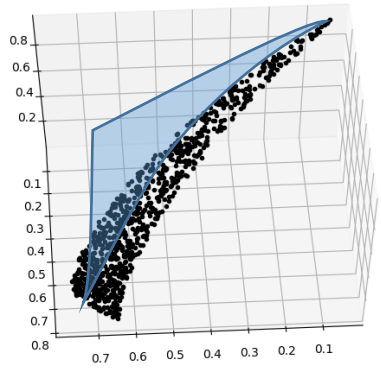


Figure 38: A Pareto Front that breaks normal PCA

In such a situation that the positivity constraint becomes active, the augmented mean plane equation is a Quadratic-Constrained Quadratic Program. Indeed, the normalization constraint makes the problem non-convex as the constraints have a large interior region around the origin that is infeasible.

4.2.2 Residual Analysis Methods

As we know that the Pareto frontier gets reduced to a set of d-1 independent coordinates that represent the spanning of the hyperplane, it remains to find the actual locations of the designs on the Pareto frontier.

There is not a single "correct" answer for this step. Any analysis will likely depend on the complexity of the Pareto frontier and also the capacity of the engineer to understand the complexity of the model chosen in this step. Hypothetically, any $\mathfrak{R}^{d-1} \rightarrow \mathfrak{R}$ approximation method should be useable depending on the Pareto frontier in question.

We chose a number of simple, classic methods for simplicity of analysis and understanding. Fourier Decomposition and Legendre Polynomials are spectral methods which is advantageous as the components will be necessarily global and are theoretically independent, which is good for characterizing the overall shape of the Pareto frontier. The Radial Basis Function network is a non-spectral method that draws a contrast as it can naturally represent local features of the Pareto frontier but is more complicated to understand.

4.2.2.1 Fourier Decomposition

The Fourier Transform is a classic technique for engineering analysis that consists of decomposing a $\Re \rightarrow \Re$ signal into sines and cosines. The higher-dimensional analogue consists of decomposition into multidimensional ($\Re^d \rightarrow \Re$) sinusoids. Intuitively, this can be thought of as the difference of waves crashing against a beach versus the single-dimensional travel of waves on a string. It can be proven that multidimensional Fourier Series can represent any continuous function (Zeidler 1995a).

The most obvious application of Fourier analysis for our problem is to find the number of "humps" in the Pareto frontier by allowing the frequency to take the role of a counter in the number of deviations from the mean plane and also allow the amplitude of the components to indicate the rough shape given by deviations from the mean plane.

Compared to the polynomial spectral methods of section 4.2.2.2, Fourier analysis has the advantage of a direction (defined here as a wave vector, which has the wavelength as the magnitude and the direction that the sinusoid "travels") and phase which grant flexibility in representing higher dimensional objects. The phase is particularly useful as it allows for "shifting the peak." This does complicate the understanding relative to the polynomials.

The Multidimensional Fourier decomposition of a signal r(x) is given by:

$$R(\vec{f}) = \int_{\Omega} r(\vec{x})\, e^{-2\pi i\, \vec{f}\cdot\vec{x}} d\vec{x} \quad [34]$$

To calculate the Fourier Transform, we start by taking the mean plane from section 4.2.1 and finding the projection of each design in the mean plane $\vec{x}$ and corresponding residual distance to the mean plane.

For this chapter, we directly computed (at $O(N^2)$ cost per frequency, N being the sample size) the discrete Fourier transform with:

$$R(\vec{f}) = \sum_n r(\vec{z}_n)\, e^{-2\pi i\, f\cdot\vec{z}_n} \quad [35]$$

f's are chosen to be fundamental frequencies that fit in the range of the corresponding ranges of x (i.e. the 1st frequency for each component is $f = \frac{1}{\max(x)-\min(x)}$, the 2nd frequency for each component is $f = \frac{2}{\max(x)-\min(x)}$ and so on until the frequencies hit the Nyquist frequency dictated by the sample density). Observe that there are negative frequencies for each real positive frequency to ensure the reconstruction remains real.

In theory, Fourier Series are able to span a very wide range of residual To help combat overfitting and to also simplify the model, when we have the spectrum of the residuals, $R(\vec{f})$, we then truncate the spectrum by setting all but the L most powerful spectral coefficients to 0. Smaller values of L are preferred to help reduce overfitting and reduce cognitive load. Quantifying exactly how many to keep is outside the scope of this chapter.

For validation purposes, when the truncated spectrum $\hat{R}$ is found by taking the top K frequencies, we can then reconstruct an approximate residual representation:

$$\hat{r}(x) = \int_{\omega} \hat{R}(\vec{f})\, e^{2\pi i\, \vec{f}\cdot\vec{x}} d\vec{f} \quad [36]$$

4.2.2.2 Residual Analysis via Legendre Polynomials

As we know that the Pareto frontier gets reduced to a set of d-1 independent coordinates that represent the spanning of the hyperplane, it remains to find the actual locations on the Pareto frontier.

There is likely not a "correct" answer for this step. Any analysis will likely depend on the complexity of the Pareto frontier and also the capacity of the engineer to understand the complexity of the model chosen in this step. Any $\Re^{d-1} \rightarrow \Re$ approximation method should be hypothetically useable depending on the Pareto frontier in question but we chose the Legendre Polynomial expansion as a potential model for a couple reasons:

1) Simplicity: Legendre polynomials are really just polynomials and have the quadratic/cubic/quartic shape of most polynomials
2) Ortho-normality of the basis: the Legendre Polynomials are orthonormal to each other under integration and as such average to zero. This allows for the effects of each component to be relatively isolated.
3) Completeness: as the Legendre basis is orthogonal, it follows that the Legendre polynomials are (theoretically) capable of approximately representing any residual function given sufficiently high numbers of parameters and sample points to an arbitrary degree.

To show point 3, we refer to the completeness of polynomials from functional analysis (Zeidler 1995b) and stick with L^2 as our norm for the criteria space and the standard integrated function inner product (multiplication and integration).

Hence, let $L^i(x)$ be the legendre polynomial in x of order i. For reference a few of the Legendre polynomials are

$$L^2(x) = \frac{3x^2-1}{2}$$
$$L^3(x) = \frac{(5x^3-3x)}{2} \quad [37]$$
$$L^4(x) = \frac{(63x^5-70x^3+15x)}{8}$$

Given mutual orthogonality, we define the multi-dimensional Legendre polynomial as:

$$L^{\vec{\imath}}(\vec{z}) = \prod_j L^{i_j}(\vec{z}_j)$$

Where $\vec{\imath}$ is the index vector of the order of each univariate Legendre polynomial (i.e. $\vec{\imath} = (1,1,2)$ would correspond to a term $L^{\vec{\imath}}(x,y,z) = L^1(x)L^1(y)L^2(z)$), j is indexing the index vector and $\vec{z}$ is the parameterization for the residual as defined earlier.

That is, we multiply multiple single-dimensional Legendre polynomials together. This leads to the immediate model for the residual:

$$r(z) = \sum_{\vec{\imath}} \alpha_{\vec{\imath}} L^{\vec{\imath}}(z) \quad [38]$$

Where $\alpha_{\vec{\imath}}$ is the weight associated with each multi-dimensional polynomial.

We can similarly calculate the pseudo-Vandermonde Matrix (the matrix of terms of the polynomial expansion down rows with evaluated at sample points across columns):

$$L = \begin{bmatrix} L^{000}(z_1) & \cdots & L^{000}(z_n) \\ \vdots & \ddots & \vdots \\ L^{\vec{\imath}}(z_1) & \cdots & L\hat{}\vec{\imath}(z_n) \end{bmatrix} \quad [39]$$

In practice, the pseudo-Vandermonde matrix is often ill-conditioned and the interpolation can suffer from Runge's phenomenon when using the Vandermonde matrix to find the coefficients via pseudo-inversion. Hence, we actually follow an approach inspired heavily by (Jung and Stefan 2011). Let us take the SVD of the pseudo-vandermonde matrix:

$$L = U_L \Sigma_L V_L^T \quad [40]$$

As done by Jung and Stefan, let the ill-conditioning threshold be then defined:

$$c_c = \varepsilon_m \Sigma_{L,(0,0)} \prod \vec{o} \quad [41]$$

Where ε_m is the machine epsilon and $\prod \vec{o}$ is the product of the maximum degree for each dimension. For this chapter we chose

$$o = \left\lfloor 2\sqrt[2d]{N-1} \right\rfloor \quad [42]$$

for each dimension. This was done largely arbitrarily and there's no particular reason a different method of choosing the orders of the polynomials in each dimension should follow this scheme. Furthermore let the Runge threshold be given by:

$$c_R = \max_{\text{rows}}\left(columnIndexOfMax(V_L^T)\right) \quad [43]$$

And then we truncate the eigenvalues of L:

$$\Sigma'_{L,(i,j)} = \begin{cases} \Sigma_{\mathrm{L}(i,j)} \; if \; \Sigma_{\mathrm{L}(i,j)} > \max(c_R, c_c) \\ 0 \text{ otherwise} \end{cases} \quad [44]$$

And recalculate the truncated pseudo-Vandermonde matrix as

$$L' = U_L \Sigma'_L V_L^T \quad [45]$$

Finally we use the pseudoinverse to evaluate

$$\vec{\alpha} = (L')^{\dagger}\vec{r} \quad [46]$$

To simplify interpretation, we want to additionally focus on only the most significant effects. For this, we sort the values of the components of $\vec{\alpha}$ and look at the largest values.

4.2.2.3 Radial Basis Function Neural Network (RBFNN)

To contrast with the global methods of Legendre polynomials and Fourier transforms, we also include a basic radial basis function neural network. The global methods attempt to find parameters which fit the entire Pareto surface well and getting a complicated fit in one area often comes at the cost of accuracy in another. For comparison the RBFNN uses individual locations with a bandwidth and value to form a kind of localized average.

The model for a radial basis function is simply

$$r(z) = B + \sum_j \alpha_j \phi\left(\frac{|z_j - l_j|}{h_j}\right) \quad [47]$$

Where B is a constant bias term (observe that RBFNN is not centered by default like the polynomials or Fourier series), α are weights, h is a bandwidth parameter and l is the location of the center of the radial basis neuron. For this chapter, we chose a Gaussian as the radial basis function to use:

$$\phi(\beta) = e^{-\beta^2} \quad [48]$$

Where β is a 1d dummy variable. While the model is simple, the parameter estimation is much more complicated. For this chapter, we used a 2 step optimization using Scipy optimization libraries (Community 2019). The lowest level would optimize and L1 penalized version of the network with the evaluation function:

$$Q_1 = \left| r_{train} - \left(B + \sum_j \alpha_j \phi \left(\frac{|z_j - l_j|}{h_j} \right) \right) \right|^2 + \zeta \left| \sum_j \alpha_j \right|_1 \quad [49]$$

Where $|\cdot|_1$ is the L1 norm and ζ is a penalty chosen by the a 2nd phase optimization.

The higher level optimizer attempted to estimate the actual loss (instead of just penalized training loss) by use of k-fold cross-validation and optimize for a good penalty.

The high level optimizer would attempt to optimize ζ over the evaluation function

$$Q_2 = \sum_{i \in K_f^V} \left| r_i - \left(B^*(\zeta) + \sum_j \alpha_j^*(\zeta) \, \phi \left(\frac{|z_j - l_j|}{h_j} \right) \right) \right|^2 \quad [50]$$

Where K_f^V is the validation split of the fth k-fold in the cross validation. Thus, internally, the evaluation function would evaluate:

$$\alpha^*, B^* = \underset{\alpha, B}{\operatorname{argmin}} \, Q_1(\zeta, r_{train} \in K_f^T) \quad [51]$$

Which means that the outer optimization

$$\alpha^{**}, B^{**} = \underset{\zeta}{\operatorname{argmin}} \, Q_2(\zeta) \quad [52]$$

Would pass in a ζ, and the outer evaluation function would perform k-fold cross validation before passing the k-fold training set to the inner optimization to get potentially optimal sized parameters and networks. The penalty term itself is stripped out of the outer optimization to combat regularization bias.

As with the polynomials and Fourier series, after fitting the data, all but the most important terms are dropped to help combat overfitting and provide a simpler picture to a potential user.

4.2.3 Effects of Centering

As described in section 4.2.1.1, $\left(T = \{b_i\}_{i=1}^{d-1}\right) \cup \{n\}$ defines a basis. Therefore, we just break up the dataset into two parts: the coordinates:

$$z = T^T Y \quad [53]$$

and the residuals

$$\overrightarrow{r_{train}} = \hat{Y}\vec{n} \quad [54]$$

where $\hat{Y}$ is just the centered points in the sample and $\overrightarrow{r_{train}}$ are the corresponding residuals. Observe that this gives us sufficient information to create a model of r using one of the methods in section 4.2.2. Notice that because $\hat{Y}$ is centered, $\vec{1} \cdot \overrightarrow{r_{train}} = \vec{1} \cdot \left(Y - \vec{1}\frac{\vec{1}Y}{N}\right)\vec{n} = \left(\vec{1}Y - N\frac{\vec{1}Y}{N}\right)\vec{n} = 0$

Geometrically, this looks like the mean plane passing through the center (coordinate average) of the Pareto frontier and the residuals being balanced between positive and negative. This is a useful fact as it means that spectral methods like Fourier analysis or Legendre Polynomials will necessarily have zero constant component.

Observe that because $r(z), \mathcal{Z}$ parameterizes the Pareto surface, we can generate ("predict") Pareto surface points by taking elements from the parameterization space $\Re^{d-1}$ and then apply an empirical model of $r(z)$ to find the anticipated location on the Pareto surface.

4.3 Calculation of Useful Metrics

One major use of the model is to perform direct calculation of important quantities for engineering design. Zero-residual tradeoffs provide a potential interpretation of the mean plane as yielding the tradeoffs of the model in the absence of the surrogate. Calculation of the tradeoff rates gives an estimate of the marginal rate of substitution at any design in the tradespace, which can yield insights into what is being "given up" by choosing one design over another. Finally, a common question in engineering design is the use of "knee points" for selecting designs; such knee points are points where the tradeoff ratio switches sign; in differentiable spaces, this corresponds to a positive definite Hessian with zero gradient; we provide a calculation of the Hessian for our interpolation model.

4.3.1 Zero-Residual Tradeoffs

If we take the simplest model for the MAPSA equation and only use the mean plane (that is, $r(\vec{z}) = 0\ \forall \vec{z}$) it is easy to obtain the derivative as $\frac{\partial}{\partial t}\vec{y}\left(\vec{z}(t)\right) = T\vec{z}$ where $\vec{z}(t)$ denotes a path with *t* as the parameter of the path. Now we would rather simplify this expression in terms of the normal vector. We require that when computing tradeoffs, we go only in the directions of interest and look at objectives in pairs. So let i, j indicate the objectives of interest. Now let T_{ij} denote the submatrix with rows ij and let $T_{\neq ij}$ be the submatrix with the other rows. Now recall othornormality of the basis and normal vector yields:

$$\begin{bmatrix} T_{\neq ij} & n_{\neq ij} \\ T_{ij} & n_{ij} \end{bmatrix}\begin{bmatrix} T_{\neq ij} & n_{\neq ij} \\ T_{ij} & n_{ij} \end{bmatrix}^T = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad [55]$$

Hence we can apply block matrix multiply and find:

$$T_{\neq ij}T_{ij}^T + \vec{n}_{\neq ij}\vec{n}_{ij} = 0 \Rightarrow T_{\neq ij}T_{ij}^T = -\vec{n}_{\neq ij}\vec{n}_{ij} \quad [56]$$

To look at only two objectives means the path $\vec{z}(t)$ must travel in only the directions that would describe directions *i* and *j*. This gives a null constraint on the derivative at the point of interest $T_{\neq ij}z' = 0$, to which we guess:

$$z' \text{proportionalTo } T_{ij}^T(n_i\vec{e}_j - n_j\vec{e}_i\,)) \quad [57]$$

Which we then substitute in

$$T_{\neq ij}z' = T_{\neq ij}T_{ij}^T\begin{bmatrix} -n_j \\ n_i \end{bmatrix} = -\vec{n}_{\neq ij}\vec{n}_{ij}^T(n_i\vec{e}_j - n_j\vec{e}_i\,) = \vec{0} \quad [58]$$

Verifying our guess satisfied the null constraint, returning to the block matrix multiply we find:

$$T_{ij}T_{ij}^T + n_{ij}n_{ij}^T = I \Rightarrow T_{ij}T_{ij}^T = I - n_{ij}n_{ij}^T \quad [59]$$

And hence,

$$\vec{e}_i \cdot T_{ij}z' = \vec{e}_i \cdot T_{ij}T_{ij}^T\left(n_i\vec{e}_j - n_j\vec{e}_i\,\right) = \vec{e}_i \cdot \left(n_i\vec{e}_j - n_j\vec{e}_i\,\right) \quad [60]$$

Giving the simplified tradeoff of:

$$M = \frac{\partial y_i}{\partial y_j} = \frac{\partial y_i}{\partial t}\left(\frac{\partial y_i}{\partial t}\right)^{-1} = \vec{e}_i \cdot T_{ij}z'\left(\vec{e}_j \cdot T_{ij}z'\right)^{-1} = -\frac{n_j}{n_i} \quad [61]$$

In 2d this can be seen by examining similar triangles geometrically in Figure 39

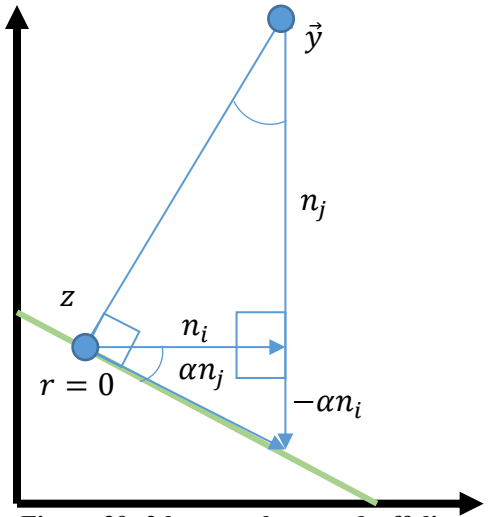


Figure 39: 2d mean plane tradeoff diagram

4.3.2 Tradeoff Calculation

Since we have modelled the Pareto front with differentiable functions, we can calculate the tradeoffs. The tradeoff between objectives is useful because it informs us what we give up in one objective for an improvement in another at a given point on the Pareto frontier. In economics, this is the marginal rate of substitution. In other contexts, this is closer to sensitivity in that it describes how much change in a given objective could be had for a change in another objective. In any case, it will be a kind of first derivative at a given point on the Pareto frontier.

We can think of the Pareto front as being approximately described by the set of points defined by having a location on the mean plane such that after moving back into the original $\Re^d$ space we can multiply the residual to the man plane's normal vector. That is:

$$\hat{P} = \{\vec{y} \mid \exists \vec{z} : \vec{y} = T^T \vec{z} + \vec{n}\hat{r}(\vec{z})\} \quad [62]$$

We now imagine a trajectory that produces locations on the mean plane and corresponding locations on the Pareto front:

$$\vec{y}(\vec{z}(t)) \in \hat{P} \quad [63]$$

Because of the parameterization of the path $\vec{z}(t)$ we will require that the trajectory be confined to the Pareto frontier and will produce tradeoff ratios analogous to the rate at which pairs of objectives will trade off with each other. Intuitively, this represents an engineer looking

at tradeoff rates to find a design suitable tradeoff rate. Alternatively, one can imagine an engineer repeatedly calculating the tradeoff rates at a given design and evaluating the next design to consider by following the preferred tradeoff rate while keeping the other variables as close to the same as possible—essentially implementing coordinate descent by hand. In any case, we can compute the derivative of the trajectory:

$$\frac{\partial}{\partial t}\vec{y}\left(\vec{z}(t)\right) = \left(T + \vec{n}\nabla_{\vec{z}}\hat{r}(\vec{z})\right)\frac{\partial}{\partial \mathrm{t}}\vec{z}(t) \quad [64]$$

Where the $\vec{n}\nabla\hat{r}(\vec{x})$ should be seen as an outer product (nabla denote the Jacobian matrix) of the residual model. This is found by simply taking partial derivatives of each component of $\vec{y}\left(\vec{z}(t)\right)$.

For separable basis (Legendre polynomials or Fourier Transform):

$$\frac{\partial}{\partial z_j}L^{\vec{\imath}}(\vec{z}) = \frac{\partial}{\partial z_j}\prod_j L^{i_j}(\vec{z}_j) = \frac{L^{\vec{\imath}}(z)}{L^j(z_j)}\frac{\partial L^j(z_j)}{\partial z_j} \quad [65]$$

Which can then be substituted into:

$$\frac{\partial}{\partial z_j}r(z) = \frac{\partial}{\partial z_j}\sum_{\vec{\imath}}\alpha_{\vec{\imath}}L^{\vec{\imath}}(z) = \sum_{\vec{\imath}}\alpha_{\vec{\imath}}\frac{\partial}{\partial z_j}L^{\vec{\imath}}(z) \quad [66]$$

And the partial derivatives can be stacked to get the relevant gradient/Jacobian matrix of $r(z)$.

This leaves open the question of how to choose $z(t)$.

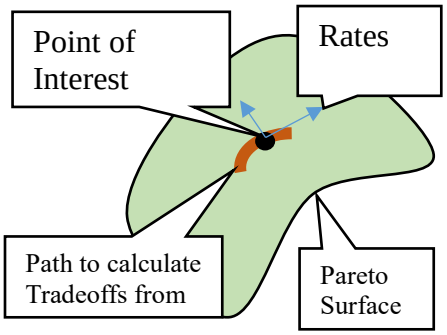


Figure 40: Diagram of Path for Tradeoffs and Curvature

When computing the tradeoff between objectives *i* and *j*, we will require that the trajectory only changes in these two directions, that is:

$$\vec{y}\left(\vec{z}(t)\right)_{k\neq i,j} \; is\; constant \quad [67]$$

This immediately leads to a constraint on the gradient.

$$\vec{e}_k \cdot \left(\frac{\partial}{\partial t}\vec{y}(\vec{z}(t))\right) = 0 \; \forall k \neq i, j \quad [68]$$

Where $\vec{e}_k$ is the elementary basis (0 for all components except *k*, which is 1). Noting then there are $d-1$ components to $\nabla\vec{z}(t)$ and $d-2$ such constraints, we can without loss require $|\nabla\vec{z}(t)| = 1$ and just compute an appropriate $\nabla\vec{z}(t)$ by computing the nullspace of the constraints. That is:

$$\nabla\vec{z}(t) = \frac{\vec{v}}{|\vec{v}|} : \left([T + \vec{n}\nabla_{\vec{z}}\hat{r}(\vec{z})]_{rows,except\ i,j}\right)\vec{v} = 0, v \in \Re^{d-1} \quad [69]$$

As in the section 4.3.1, the tradeoffs (M, for marginal utility) are then given computed:

$$M_{ij}(\vec{z}_0) = -\frac{e_i \cdot \nabla\vec{y}(\vec{z}(t))}{e_j \cdot \nabla\vec{y}(\vec{z}(t))} \quad [70]$$

Or, to substitute in expressions for the gradient from earlier:

$$M_{ij}(\vec{z}_0) = -\frac{e_i \cdot \left(T + \vec{n}\nabla_{\vec{z}}\hat{r}(\vec{z})\right)\vec{v}}{e_j \cdot \left(T + \vec{n}\nabla_{\vec{z}}\hat{r}(\vec{z})\right)\vec{v}} \quad [71]$$

Where the negative is because we expect the objectives to trade relative to each other (one increases as the other decreases) and the $\vec{z}_0$ is to remind us that we will need to specify a location in the space to compute the tradeoff. Because of the parameterization, we can extract such $\overrightarrow{z_0}$ for any point $\vec{y}$ of interest by simply taking $\overrightarrow{z_0} = \mathrm{T}^{\mathrm{T}}(\overrightarrow{y_0} - \bar{y})$

4.3.3 Curvature Calculation

We can, however, go one step further and look to calculate something analogous to a curvature. Specifically, we can get a 2nd derivative (or higher) of the Pareto surface as modelled by our mean plane and residual.

We continue with the idea of taking a curve and following the Pareto surface, but this time we want to know if the surface is bending towards or away from the optimal point at the origin. The 2nd derivative will correspond to the behavior that Unal would call “convexity” or more commonly would be referred to as curvature (Unal, Warn and Simpson 2016). Taking another derivative we do some calculations to get:

$$\frac{\partial^2}{\partial t^2}\vec{y}\left(\vec{z}(t)\right) = \left(T + \vec{n}\nabla_{\vec{z}}\hat{r}(\vec{z})\right)\frac{\partial^2}{\partial t^2}\vec{z}(t) + \vec{n}\left(\frac{\partial}{\partial t}\vec{z}(t)\cdot\nabla_z^2\hat{r}(\vec{z})\frac{\partial}{\partial t}\vec{z}(t)\right) \quad [72]$$

Where the $\nabla^2\hat{r}\left(\vec{z}(t)\right)$ is the Hessian matrix $\hat{r}$ in terms of the variables z. The first term is clearly just the acceleration due to the path itself ($\frac{\partial^2}{\partial t^2}\vec{z}(t)$) and the second term is the acceleration due to the curve geometry ($\nabla_z^2\hat{r}(\vec{z})$).

To clean things up a bit, remember that we have control of the path chosen $\vec{z}(t)$ hence we can choose:

$$\vec{z}(t) = \vec{v}_d t + \vec{v}_c \quad [73]$$

Where v_d, v_c are chosen completely arbitrarily. Thus the curvature equation reduces down to:

$$\frac{\partial^2}{\partial t^2}\vec{y}\left(\vec{z}(t)\right) = \vec{n}(\vec{v}_d\cdot\nabla_z^2\hat{r}(\vec{z})\vec{v}_d) \quad [74]$$

Which implies that the important quantity will be the Hessian matrix inside. In particular, if $\nabla_z^2\hat{r}(\vec{z})$ is positive-definite, then because $\vec{n} > 0$ we will always have positive 2nd derivative, meaning that we are accelerating away from the origin in all variables.

As a quick demonstration of what this looks like, consider the d=2 case so that $\hat{r}(z), \vec{z}(t)$ are single scalar functions of a single variable. The simplified curvature equation is then:

$$\frac{\partial}{\partial t^2}\vec{y}\left(z(t)\right) = \vec{n}\left(v_d^2\frac{\partial\hat{r}(z)}{\partial z^2}\right) \text{proportionalTo } \frac{\partial\hat{r}(z)}{\partial z^2}\vec{n} \quad [75]$$

So the derivative would be dictated by the sign of the 2nd derivative.

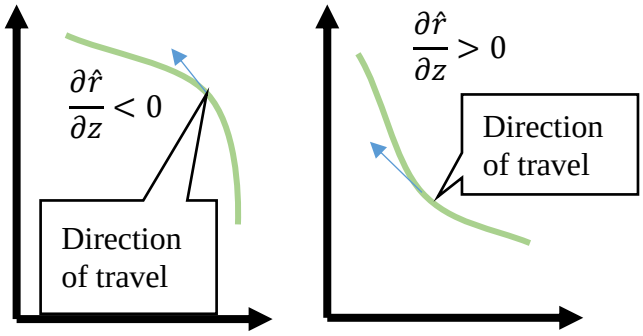


Figure 41: Curvature in 2d diagram

Observe that curvature is a local property and we must fix the path to intersect a point of interest. The sign of the curvature can change with location on the Pareto frontier (e.g. "wavy"

Pareto frontiers). Higher dimensional Pareto frontiers can have saddle points and other more complicated behaviors.

4.4 Visualization

4.4.1 Considerations for Dimension When Making Graphics

In very low dimension visualization is easy as we can directly represent the Pareto frontier. The advantage of MAPSA is that it provides a means to numerically quantify the shape of the Pareto front and potentially automate sifting through Pareto frontiers.

In three dimensions, it is still possible to directly visualize the Pareto surface but interpretation becomes more challenging. In this case, being able to quantify the shape of the Pareto front provides very helpful information regarding the behavior of the Pareto front and graphics to represent the power and possibly phase of specific terms in the residual models become very feasible as these graphics are inherently 2-dimensional.

In very high dimensional cases (≥4) direct visualization becomes impossible. However, it is still possible to represent the Pareto frontier numerically and "visualize" by reporting on key numbers and summarizing basic understanding of the shape.

4.4.2 Tradeoff and Curvature Plots

In principle, it should be simple to compute the tradeoffs for each pair of objectives, however the tradeoff can be a function of the location on the Pareto front and as such we will attempt to represent this dependence in arbitrary dimension on a 2d computer monitor.

We start by noting that if we only use the mean plane (that is, pretend the residual $\hat{r}$ is 0 and let the mean plane alone represent the frontier), Eqn. 70 becomes independent of the position and reduces to the much simpler:

$$M'_{ij} = -\frac{n_i}{n_j} \qquad [76]$$

Since the tradeoffs are calculated for a pair of objectives, we can think of a table collecting values of $M_{ij}(\vec{z}_0)$. As such, we will represent the tradeoffs as a color-coded table. The result will be as seen in Figure 52, Figure 60 and Figure 67. To draw such figures we perform the following:

1. Resample at random from the original Pareto frontier
2. Take the PCA of the resample and reduce to 2 dimensions to compress high-dimensional Pareto frontiers parameterizations to a computer monitor
3. Add a small amount of noise to the reduced locations. Center by subtracting means.
4. Form a uniform 2d grid of locations spanning the values of the reduced locations of the resample
5. Compute all distances between the centered uniform 2d grid and the noisy 2d resample locations
6. Compute the bipartite partial matching between the 2d grid and the 2d resample locations
7. Color elements of the grid by $M_{ij}(\vec{x})$ where $\vec{x}$ is the location the grid point was matched to. For unmatched grid elements use M'_{ij}

The resampling is performed to control the number of locations that is represented. PCA and the matching procedure is used to approximately capture spatial relationships between elements of the sample in the resulting image and the noise is added to break up locations that are reduced to the same location in the PCA. The partial matching is the solution to the problem

$$\min_{C:S\to G} \sum_{\vec{z}^k\in S} \left|\left|\vec{z}^k - C(\vec{z}^k)\right|\right|_2^2 \qquad [77]$$

Where $\vec{z}$ is a 2d noised location in the resample, and C is a function that matches elements of the resample to the grid. Essentially, we require every sample point be represented by one grid point and perform the matching to minimize total movement of points. Partial matching can be solved with the Hungarian algorithm (Kuhn 1955).

The color procedure causes the resample points to appear as tradeoffs over a background of the mean plane tradeoffs allowing a comparison of the overall tradeoff to the realized tradeoff values.

4.5 Comparative Study

4.5.1 Pareto Front with Multiple Square Discontinuities

The Pareto Fron dominance condition is axis-aligned in the sense that any point is possible to draw an axis-aligned hypercube such that no point falls in another point's region of dominance. Similarly, there is a set of hyper-cubes such that the given point is not dominated by a hypercube induced at the corner of each hypercube in the set. Ultimately, this means that in 2 dimensions the "limiting" Pareto frontier is an axis-aligned sawtooth wave as seen in Figure 42.

As a test of the ability of the method to represent this "extreme" Pareto frontier, we compare the ability of each modelling method to reproduce this curve and characterize it's multiple dips.

4.5.1.1 Fourier Decomposition

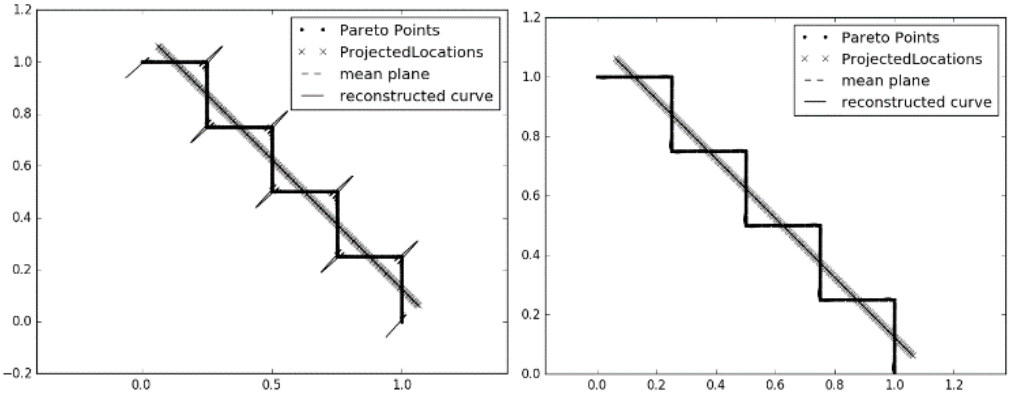


Figure 42: Multiple Square Discontinuities with Fourier Decomposition Example Mean Plane, Sampled Curve and (left) Reconstructed Curve with full spectral information (right) Reconstructed curve with only top 2 components

When applying Fourier Decomposition we can clearly observe Gibb's phenomenon at the discontinuities in the residual signal derivative (cf. Figure 42, left). After truncating signals the

reconstructed signal is qualitatively better at representing the Pareto frontier and thus providing accurate information on the spectral decomposition (cf Figure 42, right)

We can still examine the spectral power to judge the overall shape of the frontier:

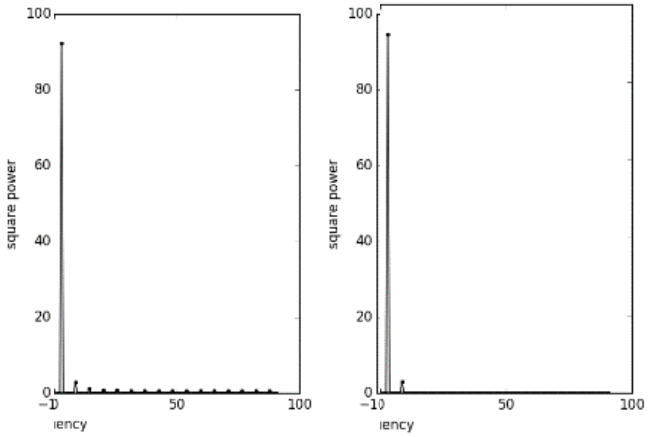


Figure 43: Multiple Square Discontinuities Example (left) full spectral power (right) truncated spectrum with only top 2 components

Figure 43 compares the importance of spectral components before and after truncation. Of note is that we see the clear sinusoidal shape of the Pareto frontier (the residual signal is a sawtooth wave). This is a reason to truncate the signal—we can automatically detect important signals beyond just the first couple of frequencies indicating curvature and unevenness.

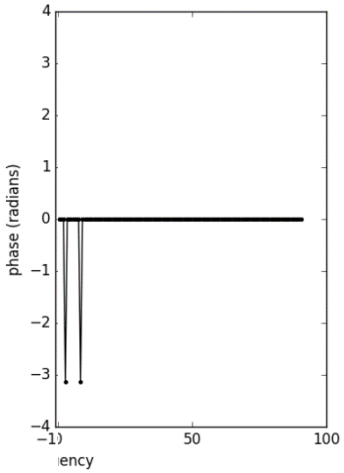


Figure 44: Multiple Square Discontinuities Example Phase

Looking at the phase, we see that both frequencies are essentially -π, which means that the humps bend away from the origin for both components, and add directly to each other, sharpening the curve. Additionally, there is no left/right bias to the components.

4.5.1.2 Legendre Polynomial Expansion

We see that the polynomial expansion follows a similar behavior to the Fourier case. However, the peak order bleeds into adjacent terms. Unsurprisingly, the peak terms occur when there are 7 inflection points (orders 10 and 12)

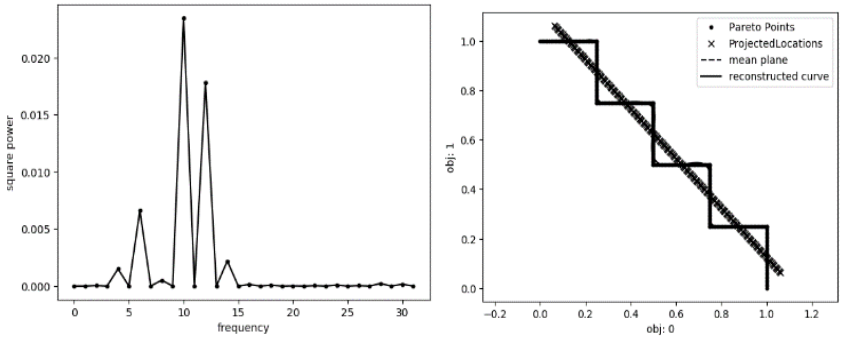


Figure 45 Multiple Square Discontinuities with Fourier Decomposition (left) Coefficient power by polynomial term order (right) reconstructed curve

Plotting these two terms together illustrates why the decomposition would focus on these two terms, as the number of peaks resembles the more significant term.

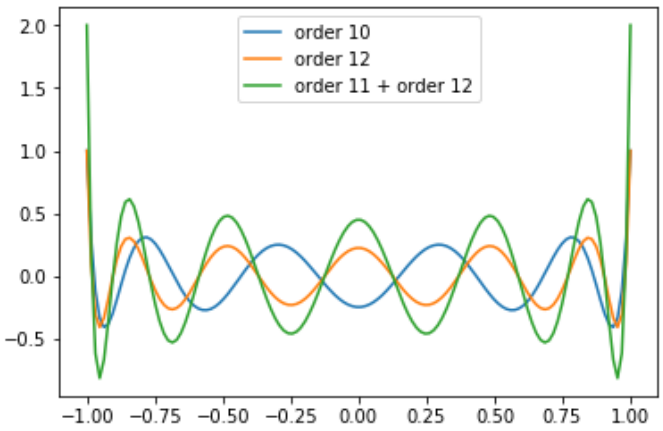


Figure 46: Illustration of terms from Lagrange Decomposition

Compared to the Fourier series, we see that the reconstruction is more complicated for the Legendre polynomials. This likely is due to the fact that the Legendre polynomials become "sharper" near the extremities when compared to the Fourier series.

4.5.1.3 Radial Basis Function Expansion

The Radial Basis Functions tend towards a very different model than the others. Evaluating parsimony is not nearly as clear as most terms have some effect on the reconstruction of the curve. This makes sense as there is not a way to represent the multi-modal curve with more than the sum of a few Gaussians.

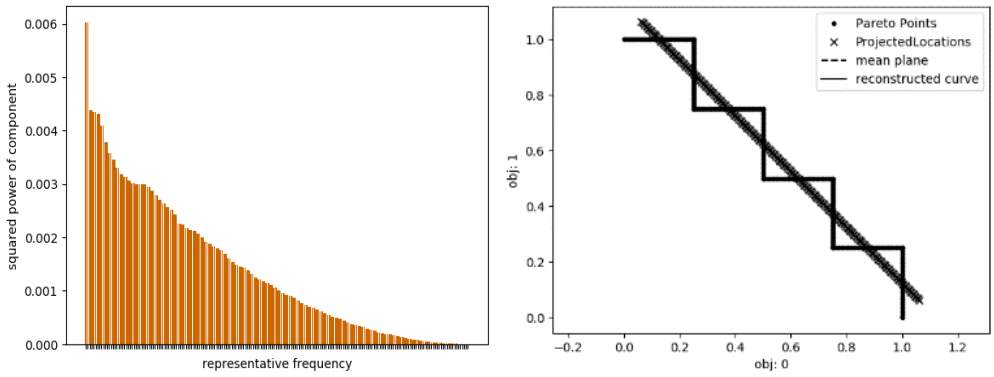


Figure 47: Multiple Square Discontinuities with Fourier Decomposition (Left) Ordered power of each term. (Right) reconstructed curve

4.5.2 Hypersphere 3-objective Pareto Front

To understand the technique in higher dimensions, consider the positive orthant sphere. This is a chosen as a Pareto frontier that is simple for the user to relate to (it represents just 1/8 of the surface of a globe). It also represents the other extreme from the sawtooth Pareto Frontier in 2 dimensions because there are no inflection points and the curvature is consistent throughout

4.5.2.1 Fourier Decomposition

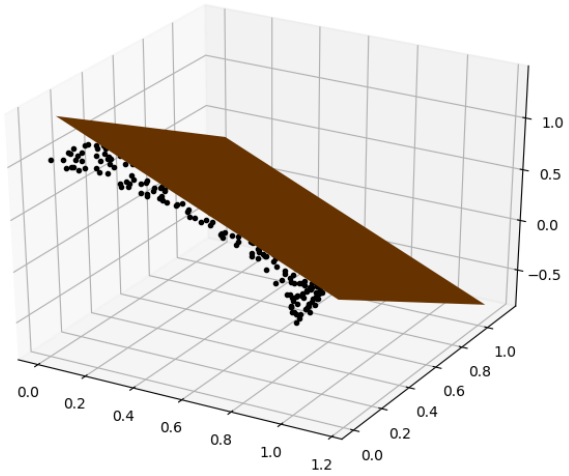


Figure 48: Spherical Pareto Front with mean plane

Figure 48 depicts the random sample of the sphere with the mean plane. We can already see the relative difficulty of visualizing the surface compared to the 2d case.

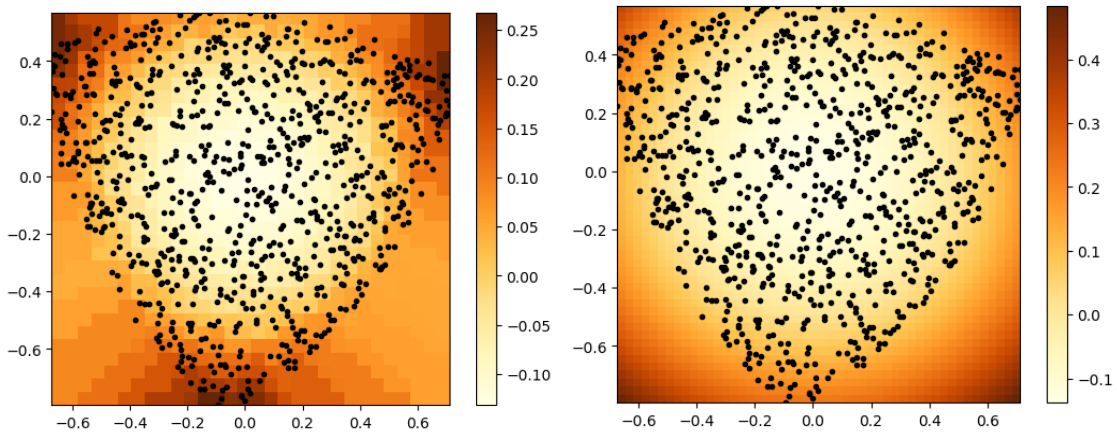


Figure 49: Sphere Example Residual Plots black dots are sample locations projected into the plane (left) Original Residual (Right) Reconstruction.

We can of course think of calculating the tradeoffs with the derivative information of the residual. Figure 49 depicts the residuals of the mean plane to the actual surface interpolated using the residual value of the nearest point at any drawn location. Be aware corners are irrelevant as they are distant from sample points and are represent extrapolations that likely don't exist in reality. We can think of the Fourier step as attempting to compress the left image with a jpeg-

style algorithm and yield the right image with as few components (and as little cognitive load) as possible.

As is clear, we can get the general shape, such as the dip in the middle though there is a certain degree of residual in the bottom-center and a general wavy shape to the reconstruction. Regardless, we are able to represent the shape to some degree and capture features like the general dip in the middle with the components we retain.

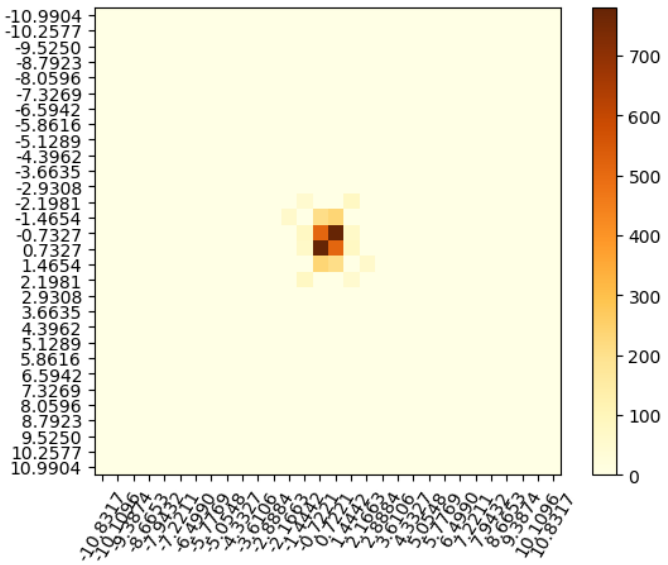


Figure 50: Spectral Plot of the Sphere Example

Figure 50 draws the spectral power of the residuals of the spherical Pareto front (recall that, in a multidimensional Fourier transform, frequency is a vector with a magnitude and direction). As can be seen, the most important components close to the origin, which corresponds to only a single hump in the data. As an alternative representation that will become more useful in higher dimensions, we can also draw the spectral power as a sorted bar graph

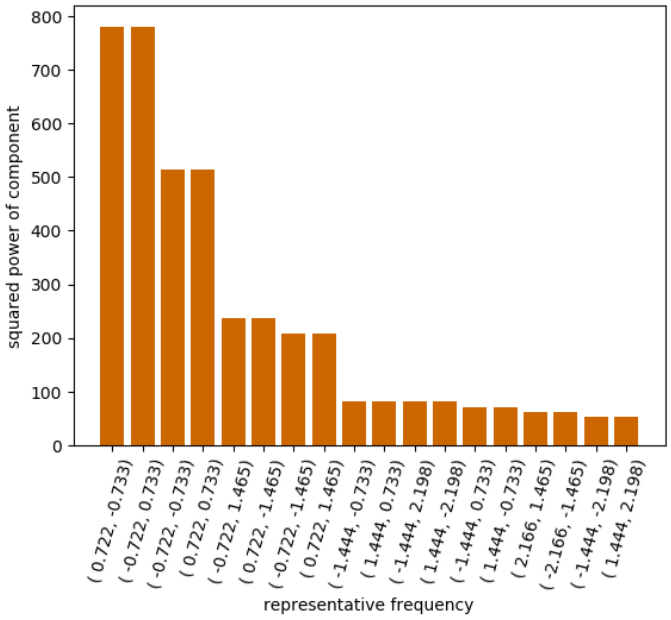


Figure 51: Sorted bar graph of Spectral Power for Spherical Example

Figure 51 reports the importance of different components to the reconstruction of the residual signal. As is clear, there is a sharp drop off with the majority of important signals being at the frequencies near the origin. Another step to produce understanding of high-dimensional spaces is to summarize the important frequencies in an automatically generated table, which is made significantly more tractable by discarding the vast majority of frequencies.

frequency		number of humps		Phase	Power
0.722111	-0.73269	1	-1	-30.5681	781.0367
0.722111	0.732694	1	1	27.5537	514.9046
0.722111	-1.46539	1	-2	-179.139	236.7183
0.722111	1.465389	1	2	159.298	208.4396
1.444223	0.732694	2	1	-106.533	81.61531
1.444223	-2.19808	2	-3	-32.8815	81.15143
1.444223	-0.73269	2	-1	95.63878	69.29516
2.166334	1.465389	3	2	82.91558	62.22246
1.444223	2.198083	2	3	-20.1206	51.90347

Table 8: Automatic Report for Sphere Example

Table 8 reports that the most important components have humps in opposite directions for the 2nd component of the frequency, so there is some kind of asymmetry in the problem (as is evident in the residual plot, this asymmetry is due to the triangular shape of the Pareto surface

projected in the plane). We can see that the majority of the surface is a kind of bowed shape bowing away from the origin.

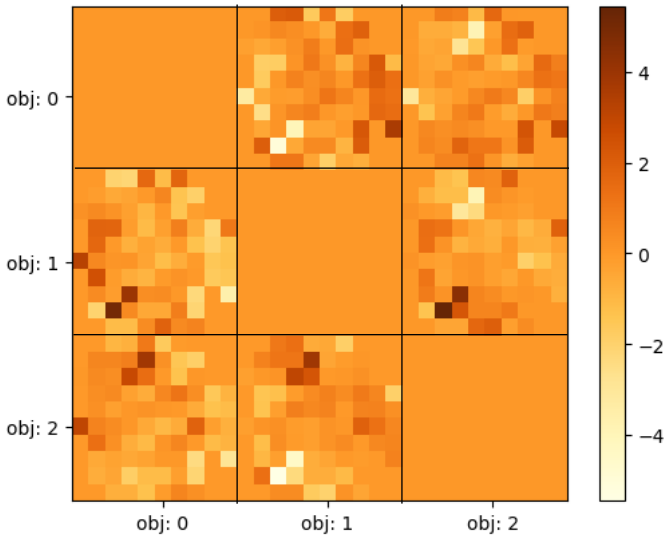


Figure 52: Logarithmic Mean Plane Tradeoff Image

Figure 52 shows the tradeoffs of the sphere. The image is noisy because of reducing the dimension and from errors in the approximation. However, we can see that there is a range of tradeoff values from very high to very low with the largest changes in the tradeoffs around the edges where the corresponding pixels are brighter or darker.

4.5.2.2 Legendre Polynomial Expansion

Given that the Fourier Transform is successful at reducing the hyperplane to a couple key central components, we consider the case of the Legendre Expansion as an alternative model.

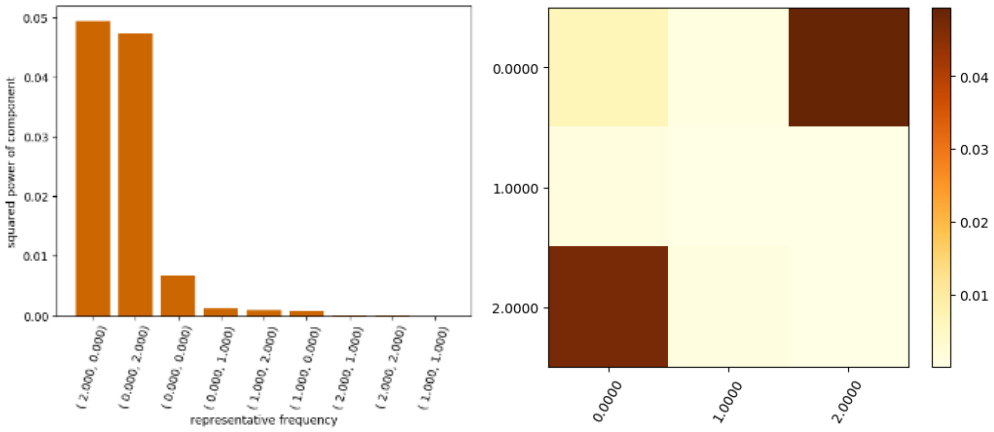


Figure 53: Power of Coefficients for semi-sphere with Legendre Residual (left) spectral power as a bar plot (right) spectral power as an image plot

First, we observe that the Legendre decomposition is much cleaner than the Fourier decomposition. This is likely because the Legendre Polynomials are essentially ideal for modelling the hypersphere—the quadratic terms are necessarily centered and so there are no estimation errors from trying to properly center the relevant terms. Additionally, whereas the Fourier decomposition tends to level out at the extremes, the Polynomials expand to infinity. This means that the quadratic terms are more than sufficient to capture the complexity of the semi-sphere. Logically, this makes sense, a semi-sphere is almost entirely a single concave curved surface like a cosine or a quadratic. Importantly, the quadratic terms oppose each other, there is one term that represents the curvature in one direction and an almost equal and opposing curve in the opposite direction.

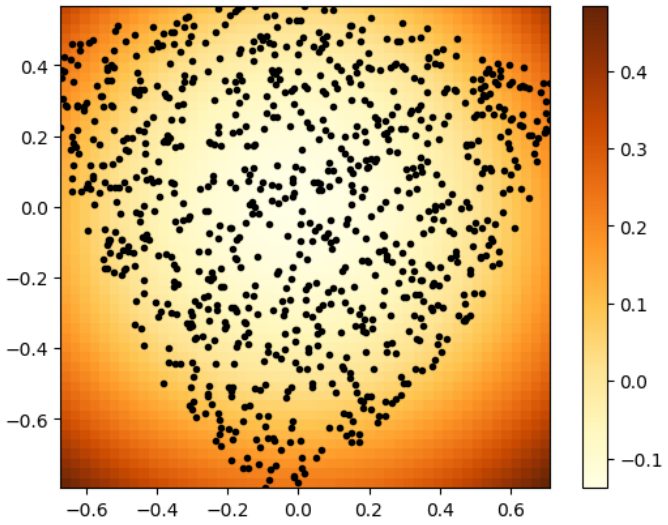


Figure 54: Residual reconstruction of the polynomial expansion for the semi-sphere

Looking at the reconstruction of the residual we see the curvature of the symmetries and cleanliness of the representation quite clearly. The ‘bowed’ coloration represents the curvature of the semi-sphere quite well.

4.5.2.3 Radial Basis Function Expansion

As the radial basis function network works differently, we expect a different kind of result. Indeed, we obtain different results because the Radial Basis Function treats the extremities as zero.

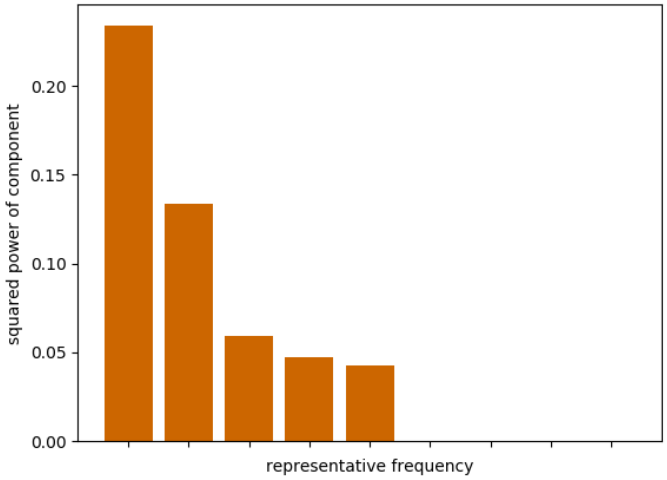


Figure 55: Radial Basis Function Coefficients for semi-sphere

We see a spectrum of terms for the radial basis function expansion. This is because there is a dominant term that models the overall falloff and a series of later terms that then fit the extreme points on the residual.

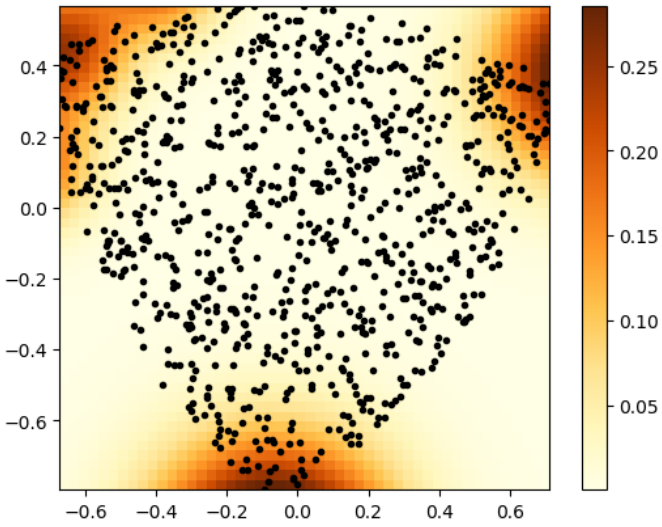


Figure 56: Residual reconstruction of the Radial Basis Function expansion for the semi-sphere

Looking at the reconstruction plot, we see the curvature is captured, but a great deal of attention is paid to the extreme corner elements of the Pareto surface. This is in contrast to the other residual reconstruction methods as they are global and attempt (and succeed) in fitting the curve with relatively few high curvature terms.

4.5.3 GNC Problem

To study how the analysis method fares in the case of a more complex real-world Pareto frontier in low dimension (for understanding) we use the study of historical NASA spacecraft by Dominguez-Garcia et al. and used in sections 2.7.4 and 3.3.4 (Dominguez-Garcia et al. 2007).

4.5.3.1 Fourier Decomposition

We start with a Fourier decomposition of the problem.

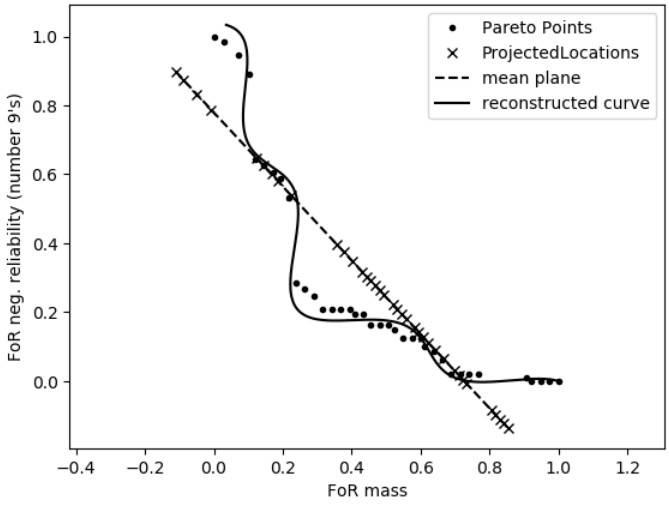


Figure 57: Mean Plane, Sampled Curve and Reconstructed Curve for GNC Problem

Figure 57 is the plot of the Pareto frontier as calculated by full factorial enumeration and filtering of the Pareto optimal points. We can observe that we are able to model the surface quite effectively with the curve reconstructed from only a few components.

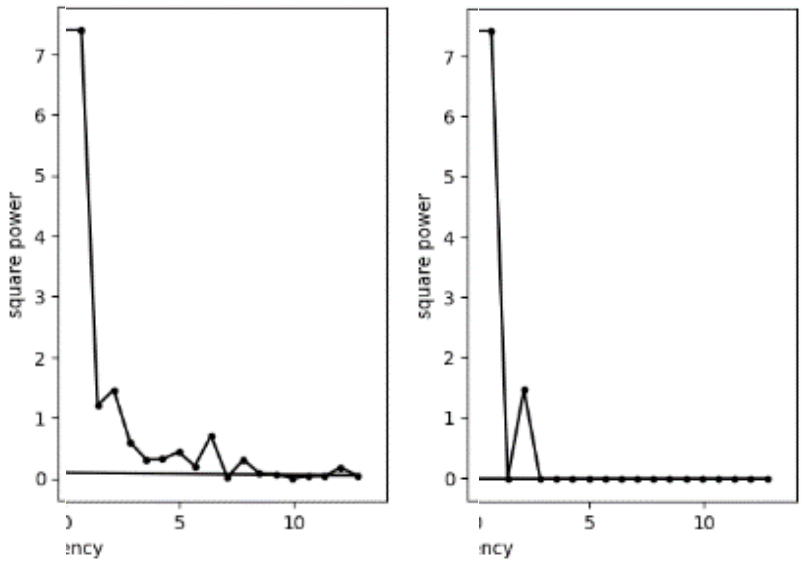


Figure 58: Full spectral power for the GNC problem

Figure 58 shows the spectrum before and after filtering. As is clear, taking 2 components is ample to represent the overall bowed in shape.

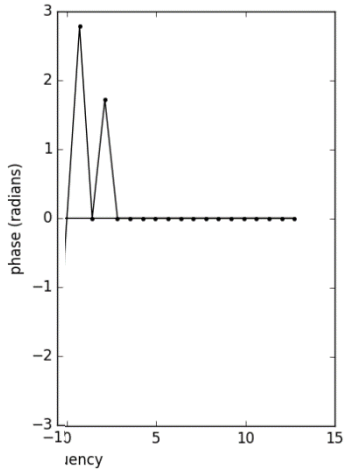


Figure 59: Spectral Phase for the GNC Problem

Figure 59 shows the smaller frequency has a large positive phase, which indicates the overall shape of the frontier bends toward the ideal point and is biased to the lower values. The second used frequency is also positive indicating it sharpens the curve and is roughly $\pi/2$ indicating it resembles the asymmetry of sine. Notice the 2nd used frequency has 3 humps corresponding to the 3 levels in Figure 57.

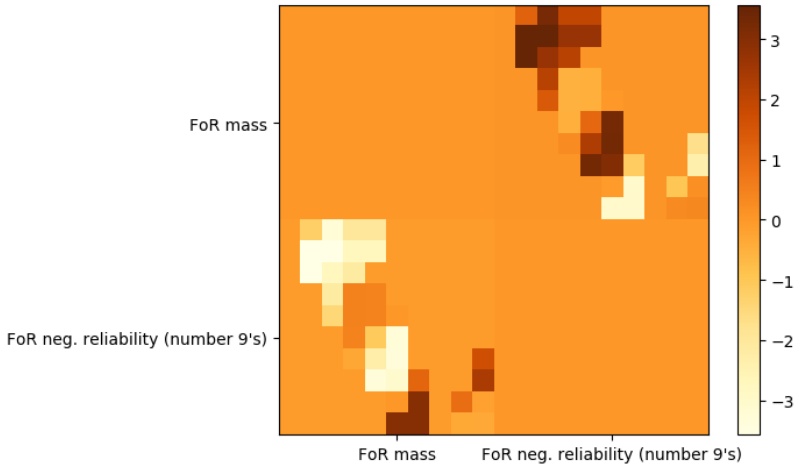


Figure 60: Tradeoff Chart for the GNC problem

For this simple 2d problem, the tradeoff plot roughly represents the shape of the Pareto frontier in the pixels of Figure 60. We can then see that the color of the half-circle appears banded in Figure 60 corresponding to alternating between the nearly vertical and nearly horizontal parts of Figure 57

4.5.3.2 Legendre Polynomial Expansion

As the polynomial expansion lacks the ability to translate the maximum of its basis functions, we expect the reconstruction and modelling to be less flexible, than the Fourier Decomposition. However, the simplicity also is advantageous as the representation will tend towards cleanliness and be less prone to overfit.

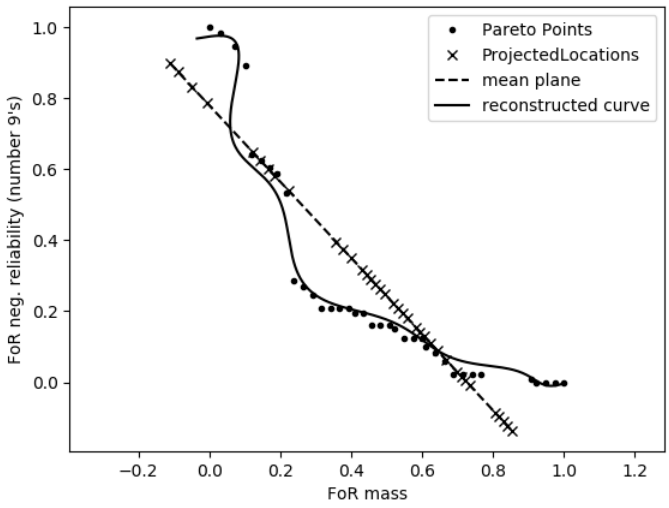


Figure 61: Legendre Reconstruction of GNC Problem

Looking at the reconstruction, we can immediately see that the representation models all points on the Pareto Frontier, though there is a degree of bias in the representation. Interestingly, the edge effects at the top left appear to be largely incorrect, but a few points on the edge are difficult to model overall, so this is generally ignored by the fit. This is an interesting limitation as there are no real guarantees of the shape of a Pareto-Frontier. For the purposes of optimization it is usually assumed that the optimization region is bounded for tractability and for multi-objective optimization it then is often assumed that the objectives will tend towards becoming axis-aligned or bounded to stay in one orthant, but strictly speaking this need not be the case. Clearly, the old rule of surrogate modelling should hold that one should be far more skeptical of extrapolations than interpolations.

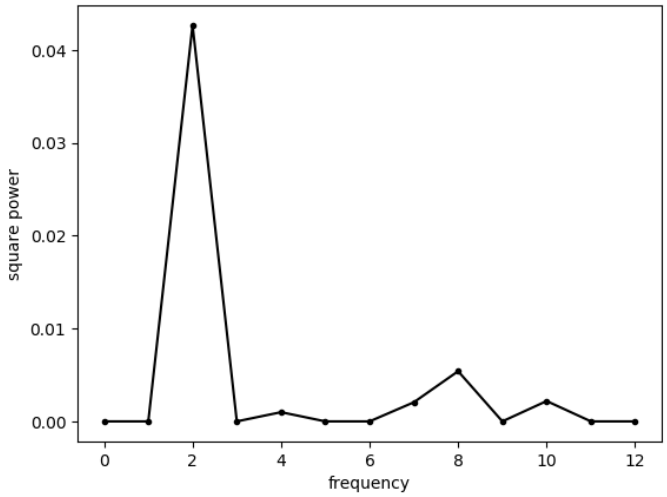


Figure 62: Coefficient Power versus term order for Legendre Polynomial of GNC problem.

Much like the Fourier decomposition, the Legendre expansion is capable of efficiently representing the strong overall bend in the Pareto front. This is well captured in the very strong signal for the quadratic term. The order 8 term then captures the 6 inflection points or 3 distinct humps corresponding to each group.

4.5.3.3 Radial Basis Function Expansion

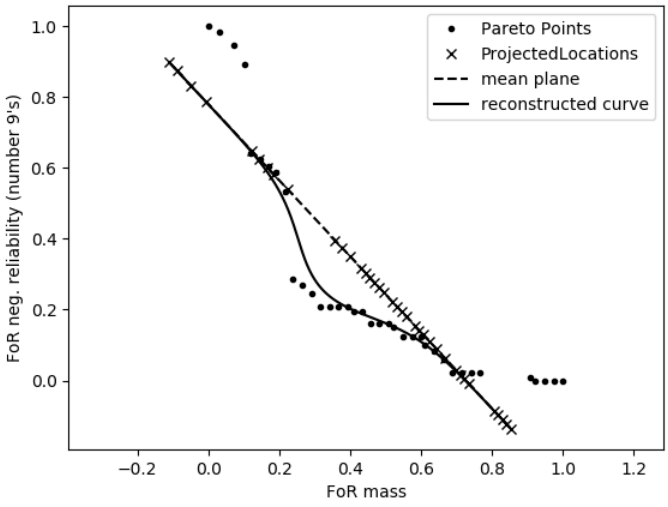


Figure 63: RBFN Reconstruction of GNC problem

In contrast to the other methods, the radial basis function network appears anemic. It heavily fits the overall curve but fails to capture much other signal.

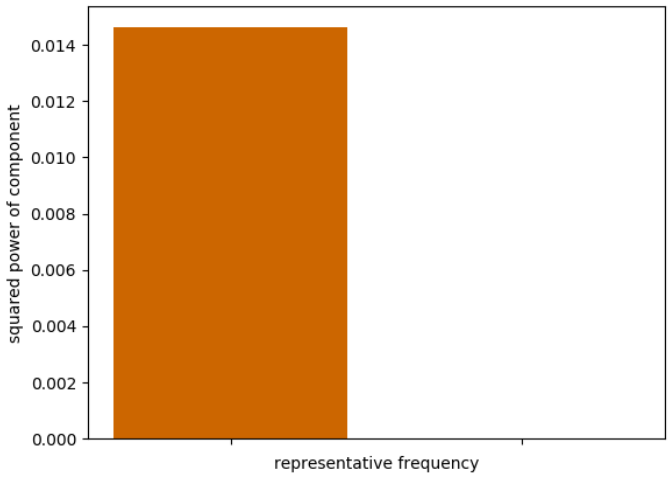


Figure 64: Coefficient power of terms for RBFN

We see in the evaluation of the power of the network terms that the RBFN only ever uses a single term for approximating the residual.

4.5.3.4 Comparison of Performance versus Number of Terms (Complexity)

An interesting question for the methods is how the performance varies with the number of terms used in the representation. This is important for a couple reasons; first, for providing guidance and rules of thumb for designers, it is sensible to expect to be able to provide a scale of models from simple and understandable to accurate; second, intuitively, the fewer the number of terms required the less effort is needed to understand the model and the Pareto front by extension; third, by mathematical theory we expect that the models should be able to fit the training set in general and should test this hypothesis. Observe that for all previous discussion the models were chosen to generally perform best in reconstruction mean square error.

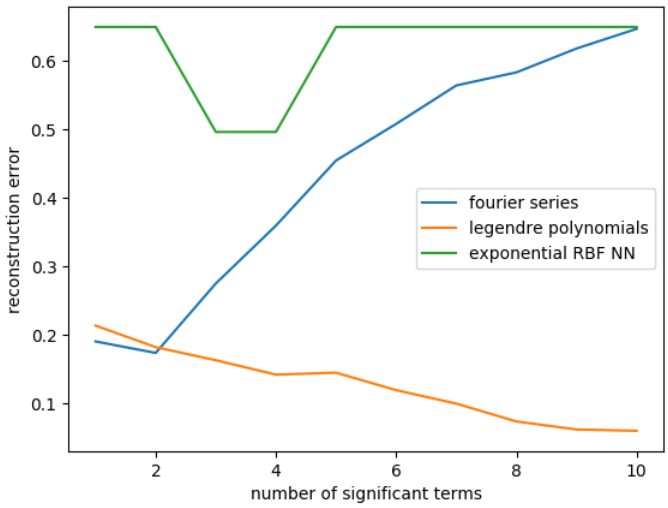


Figure 65: Reconstruction performance versus number of terms used in the model

We see some immediate trends of interest. First, the RBFN generally doesn't like to fit the model much. This is almost certainly due to the regularization penalizing the model from choosing unneeded terms and so if the number of terms doesn't easily fit the number of elements in the dataset it will generally default to not doing anything at all. We also see that the Legendre Polynomials are able to hone in on the overall effect and then quickly tune down to the finer facets of the dataset. For contrast, the Fourier Series is able to quickly capture the primary effects but becomes "distracted" as it is allowed to use more terms and tends to rapidly over-fit.

4.5.4 EOSS Down Selection

As another real-world example in a higher dimensional tradespace, we consider the results of the 2007 Earth Science Decadal Survey study introduced earlier. In the study, a subset of candidate instruments to fly are selected and the resulting satisfaction of science requirements met, lifecycle cost of the program, programmatic risk and fairness across scientific communities are evaluated (Selva, Cameron, and Crawley 2014).

4.5.4.1 Fourier Decomposition

As with other methods, we apply the same mean plane and residual decomposition as before. However the much larger number of criteria constrains our options for visualizing the shape outright. Hence we skip to the purpose of the method: the component power:

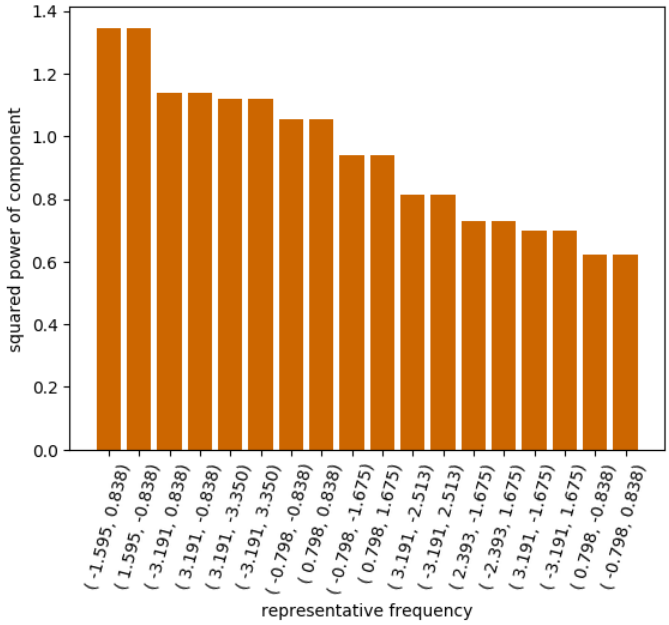


Figure 66: Spectral power of the residual signal for the EOSS Down Selection problem

We can then present the automatically generated report:

frequency			number of humps			Phase	Power
0.640843	0.818716	-3.80747	1	1	-1	30.08487	0.544898
1.281686	0.818716	-3.80747	2	1	-1	-121.941	0.510376
0.640843	-0.81872	7.614931	1	-1	2	47.98344	0.471746
1.281686	1.637432	7.614931	2	2	2	-99.8385	0.342166
0.640843	-0.81872	3.807465	1	-1	1	71.70231	0.338048
1.281686	-0.81872	3.807465	2	-1	1	6.41099	0.286471
1.281686	0.818716	-7.61493	2	1	-2	-38.8585	0.23565
1.281686	-0.81872	7.614931	2	-1	2	-92.5993	0.228637
1.281686	-0.81872	-3.80747	2	-1	-1	101.5546	0.227492
0.640843	1.637432	7.614931	1	2	2	34.58251	0.155259
0.640843	0.818716	7.614931	1	1	2	-101.063	0.150841
1.281686	1.637432	3.807465	2	2	1	-34.1117	0.142024
0.640843	-1.63743	7.614931	1	-2	2	2.762451	0.133095
1.281686	0.818716	3.807465	2	1	1	-168.612	0.115264
0.640843	-1.63743	-3.80747	1	-2	-1	49.63738	0.100894
1.281686	-1.63743	7.614931	2	-2	2	148.4087	0.096894

Table 9: EOSS Down Selection Report

Table 9 indicates that there are primarily sinusoids and have some degree of asymmetry in the Pareto frontier. Unfortunately, the decline in power for this particular problem is slow,

which makes understanding more difficult. However, the overall power is small, so the linear representation is somewhat accurate. The extreme colors in Figure 67 shows that on the average science score trades heavily with other objectives: costs, programmatic risk and fairness among scientific communities. There is a similar set of trades if we were to prioritize fairness. We also see a range of other tradeoff values. We also see that most tradeoffs tend to be 1-1 for most of the space.

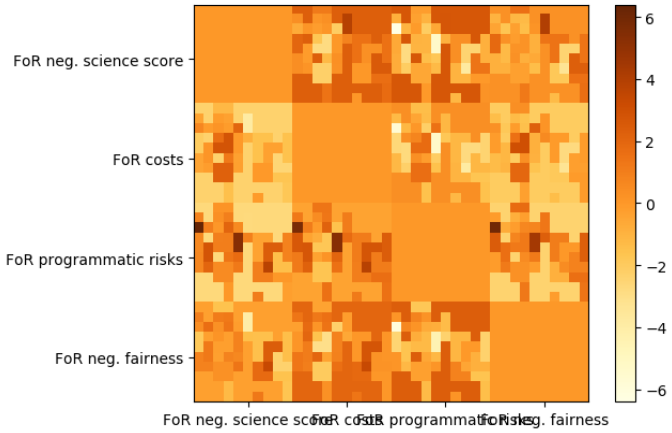


Figure 67: Mean Plane Log Tradeoffs of the EOSS Down- Selection Problem

4.5.4.2 Legendre Polynomial Expansion

As has been seen, we expect that the simpler Legendre model will tend to resemble the Fourier Series expansion, but with a stronger filter for noise but preference for more terms

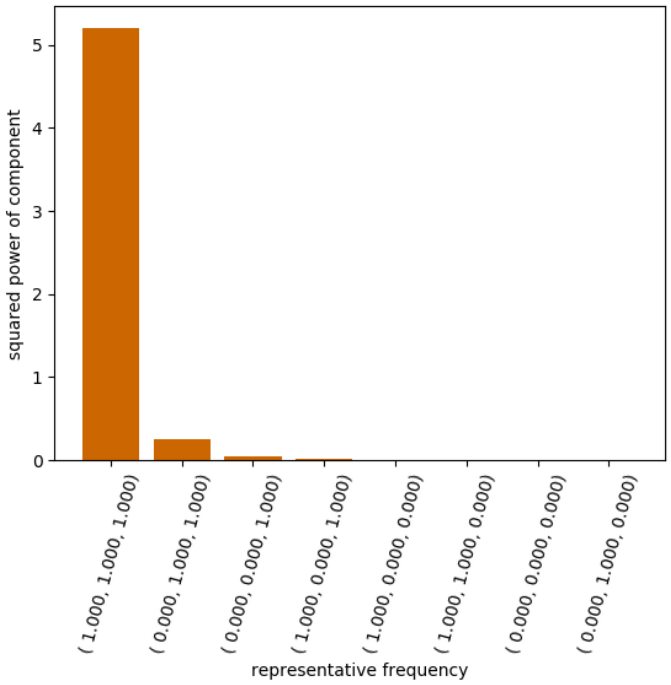


Figure 68: Power Decline plot for Legendre Polynomial Expansion

This is largely carried out by the results. We see that the residual is very well modelled by a single term of a Legendre Polynomial expansion, indicating it is mostly quadratic in the very high dimensional space. The Pareto Frontier closely resembles the 3d version of a paraboloid.

Terms			Term Power	number of humps dim 0
1	0	1	0.076972113	1.242205269
1	1	1	0.062889871	1.242205269
1	0	0	0.001610504	1.242205269
1	1	0	0.000149065	1.242205269

Table 10: Legendre Expansion Terms for EOSS Down-Selection Problem

4.5.4.3 Radial Basis Function

As has been seen in the other problems, the RBFN tends towards not using terms of the expansion if it can be avoided. Indeed, that appears to be the case here.

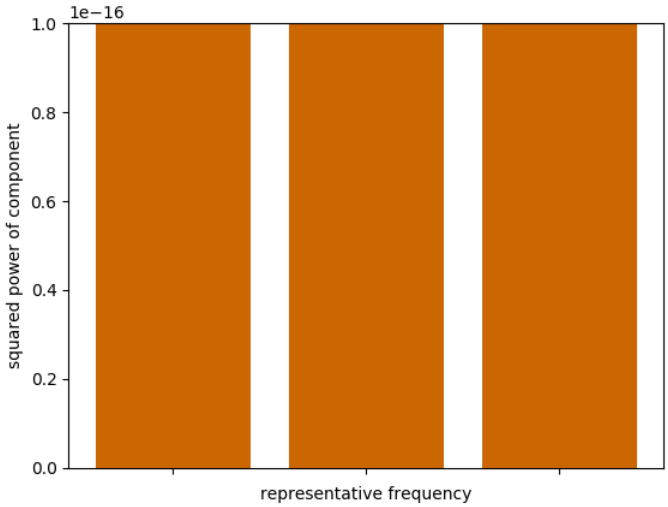


Table 11: Power of Terms for RBFN in EOSS Problem

All terms tend towards zero (or $1x10^{-16}$ to be more precise).

Center Location			spectral power
0.61399	-0.49661	0.053986	1.00E-16
0.403561	0.179326	-0.00066	1.00E-16
-0.34816	-0.03448	-0.00802	1.00E-16

Table 12: Terms power for RBFN

4.5.4.4 Comparison of Performance versus Number of Terms (complexity)

As before, we can compare the performance of the different residual models:

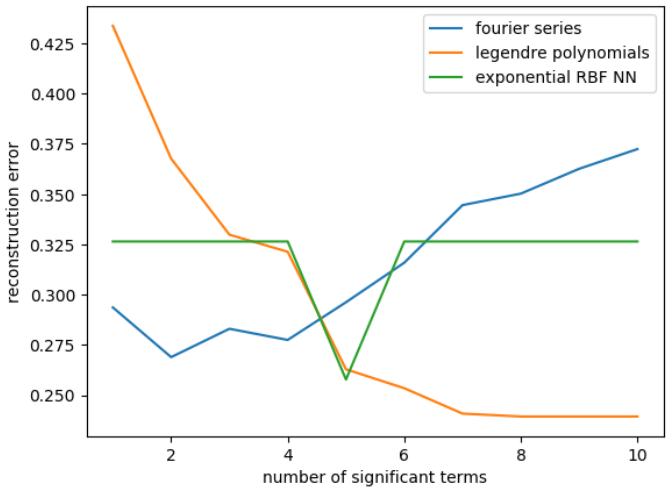


Table 13: Performance of Residual Reconstructions versus Number of Terms

We see similar results to the GNC problem when comparing the number of terms versus the performance of the different methods. As before, as the number of terms increases for the

Fourier Series, it tends strongly towards overfitting. Simultaneously, the Legendre Polynomials are able to quickly find the most dominant factors and get more accurate with small adjustments. The RBFN tends to not like mismatched terms, but can sometimes get just the correct number of terms to fit accurately.

4.6 Conclusions

We have proposed a three-step method to understand the shape of a Pareto frontier by using a mean plane to parameterize it and a surrogate model to approximate the residuals that represent it. First, a mean plane is fit to minimize the distance to the points on a sample of the Pareto frontier. Second, a multidimensional decomposition is computed on the residual to the plane with the projection of the Pareto frontier to the plane as coordinates. The Fourier decomposition is then truncated to retain the most significant components, which are reported to a user to understand the space. Third, the coefficients of the residual model are presented to a designer and the tradeoff ratios are calculated and plotted.

The mean plane can be used to gain a global understanding of the tradeoffs between objectives and provides a means to describe points on the Pareto frontier in the proper number of dimensions.

The spectral decomposition can be used to find the number of "humps" in the Pareto frontier and generate an interpretable surrogate model of the Pareto frontier. It also quantifies the nonlinearities in the surface and provides a rough human-understandable image of the Pareto frontier. For instance, the magnitude of the coefficient on the 1^{st} fundamental frequency will correspond to how generally bowed the Pareto frontier is and the phase will indicate asymmetry and whether the Pareto frontier bows towards or away from the ideal point. As the

decomposition is describing variations about a mean, a spectral transform is a reasonably effective compression and surrogate for the surface shape.

One related question for future work is what kind knowledge can be effectively gained by looking at the truncated Fourier spectrum. What kinds of visual representations might facilitate knowledge discovery form the spectra? Can we include this curvature information from the Fourier decomposition in the mean plane tradeoff plots? Can we make the reporting of Fourier components more intuitive to understand?

Finally, we reported the sparse spectral information directly. Summarizing (e.g. the Fourier decomposition is radially symmetric) may allow for even better reconstruction of the surface with easier to understand pieces.

5 FINAL CONCLUSIONS

5.1 Executive Summary

We have developed novel methods that are specialized for problems of understanding the trade and decision space of multi-criteria engineering design problems. These methods are able to use visual representations of the tradespace to allow decision makers to gain an understanding of the main drivers of the design problem. These methods generally improve upon pre-existing methods by either specializing the method to problems of multi-criteria decision making, possessing intuitive explanations and interpretations or representing more or more useful information than has been traditionally available. All methods presented here are capable of representing nonlinear Pareto frontiers and actually thrive under such circumstances.

Cityplot is the first method in the engineering community that is designed for and capable of representing the decision and tradespace information simultaneously as a landscape. It is capable of representing and encouraging understanding of the relationships between design alternatives, such as sensitivity and the presence of design families. Finally, it presents a solution to reasonably represent large numbers of objectives without the crowding of a parallel axis plot and while using the structure of the decisions to respecting the features of the tradespace and represent the criteria space. Human experimentation has confirmed the ability of the plots to be learned and navigated quickly, effectively and intuitively.

MOMS-CFCs has presented an interpretation of design spaces in the objective space as being a representation of changes in performance between designs. By mapping these changes back down to the decisions, we are able to find a method to search for groups of decisions that act in concert to drive performance changes. These groups of decisions naturally include the

performance couplings of different subsystems. The method is nonparametric and nonlinear and based on intuitively matching points in the tradespace based on distances from each other.

MAPSA presents the idea of using a hyperplane and normal projection to parameterize a Pareto Frontier and reduce the quantification of the shape to a problem of approximating an $\Re^{d-1} \rightarrow \Re$ function. We have shown that this parameterization is well-founded and reasonably general in practice. This allows for the quantification of features of the tradeoff surface and also the general creation of surrogates. We have applied the quantification as a means to understand the shape of high dimensional surfaces that cannot be easily visualized and shown how different surrogate models can provide different understanding of the surface. We have also used the surrogate models to demonstrate how information provided on tradeoff rates and curvature can be used to ease decision making.

5.2 Limitations and Future Work

The various methods presented in this thesis have their own limitations. Currently, the Cityplot renders all the objects in the space, improvements in reducing the number of necessary objects would speed up the program and enable use on larger problems. There is a related open question of whether Cityplot can reduce crowding by grouping up close designs to allow for more succinct understanding of regions of the space. Finally, most of our work on distance functions and clustering into design families is ad hoc, a more formal study of distance functions and precise definitions for clustering would aid in applicability and provide better guidance to engineers in practice as there are no universal distances for all problems. Additionally, alternative distance functions with clear engineering interpretations have interesting possibilities: suppose we used the cost to change between designs as the distance, how well does this correlate with subjective measures of "similarity"? Suppose we built trajectories of potential deployments,

could the distances be inferred from the graph of changes in a manner that would respect intuitive notions of separation?

MOMS-CFCs suffers strongly from requiring very large sample sizes and accurate measurements. This is in part due to the "bottom up" nature of the linkage method used. MOMS-CFCs makes very few, if any, assumptions on the behaviors of the tradespace. As such, it is unable to easily distinguish between signal and noise and lacks the power of methods that make stronger inferences on the dataset. Future development should find ways to loosen the deterministic nature of the method to provide more probabilistic interpretations of the change sets. In doing so, it will likely be necessary to incorporate assumptions on the nature of tradeoff and decision spaces, so basic research on what is "typical" behavior is necessary.\

Concerning MAPSA, one big issue is that the method is lossy by nature, so it is not possible to know for sure what kind of behavior underlies the Pareto Frontier even with a full set of parameters (that is, the inverse problem from model to original frontier is not well defined). It also assumes a high degree of parsimony in the surrogate for the explanations to be useful. As such, generating an automated meta-surrogate model to find strongly parsimonious surrogates will be a necessity to ease use of the method. Finally, human validation of the approach will be valued in the current state of the engineering design and systems engineering community.

Future research should broadly seek to better integrate optimization and more abstract information such as known equations or physical models. The methods presented in this research generally assume the creation of a dataset and seeks to interpolate and interpret that dataset. In practice, engineers have a rich understanding of components of the models and systems at work. While this information can be incorporated into the creation of the datasets, richer integration of such information remains an open question and relates the methods. One way to think of this

problem is one of debugging—as mathematical programs are often debugged with ad hoc plotting and desk-checking—we have the plotting but not the desk-checking; richer integration in the calculation procedure would allow for better tracing of the relevant factors in how a design is placed in the space. Alternatively, richer knowledge could be transformed into assumptions for model creation as in MOMS-CFCs to act as a baseline for the model to attempt to verify.

6 APPENDIX

TUTORIAL SCRIPT: CITYPLOT-VR

The following was displayed on the command prompt and each line was advanced by pressing any key. This tutorial was run before any tests were performed with the Cityplot tool.

Type 'y' when a window has opened with a number of boxes and lines connecting some of the boxes.

Welcome to the Cityplot-VR tutorial

You can use either the keyboard or the provided video game controller to interact with this tool.

You may pick which either control method feels more comfortable. Both controllers have the same capabilities.

REMEMBER: regardless of the controller you pick, you will have to click between this window and the window with the buildings to interact

the rectangles are 'buildings' and the colored lines are 'roads'. Groups of buildings are 'cities'

Buildings are color coded according to which objective they represent. Use the legend to check which color each building represents

cities are placed in space to represent how much the design has to change to change one design (city) into another and the placement of cities in the space roughly approximates the true number of changes to change designs in the high (20+) dimensional space where decisions are made

the roads are then the true number of changes (distance) between the designs.

Press t on the keyboard, y on the controller to see a legend that tells what the colors mean.

The left side legend labels how many changes a given road represents. The right side legend labels which objective is represented by each building.

Press t on the keyboard, y on the controller again to hide the legend. Pressing after hiding the legend will respawn it in front of you.

Now try this question.

what is the true distance represented by the yellow roads?

[while given incorrect answer:] incorrect. Try Again [and repeat]

what is the name of the objective (building) represented by the blue objective and listed in the building legend as blue?"

excellent

now for movement.

movement in Cityplot-VR is key to using the tool and is quite intuitive once you get the hang of it.

Put simply, your lefthand on the keyboard, thumb on the controller controls your movement parallel to the ground and your right hand on the keyboard, thumb on the controller controls the direction you are facing.

on the shoulders of the controller, you'll see the left and right shoulder buttons (at the bottom if there are two sets)

The e on the keyboard, R shoulder on the controller causes you to move up and the q on the keyboard, L shoulder on the controller causes you to move down.

in principle, you use your left hand fingers on the keyboard, index fingers on the controller to press these buttons and leave these fingers over these buttons at all times

if the right hand on the keyboard, thumb on the controller movement is awkward, the vertical rotation can be inverted by simply pressing i on the keyboard, start on the controller

"put longer:"

to strafe left

to strafe right

to move forward

to move backward

to move up

to move down

to look left

to look right

to look up

to look down

a on the keyboard, left stick left on the controller

d on the keyboard, left stick right on the controller

w on the keyboard, left stick up on the controller

s on the keyboard, left stick down on the controller

e on the keyboard, R shoulder on the controller

q on the keyboard, L shoulder on the controller

left arrow on the keyboard, right stick left on the controller

right arrow on the keyboard, right stick right on the controller

up arrow on the keyboard, right stick up on the controller

down arrow on the keyboard, right stick down on the controller

take a moment to move in each direction. Getting comfortable is very important to using this tool, so please play around until you feel comfortable moving around

During this tutorial, we aren't timing the time it takes to move through the space but we will be after this tutorial is over

Type 'y' when you are ready to continue.

by now you've noticed a little symbol in front of you (default is a green name tag).

The symbol is the gaze pointer and will stay centered in your vision at all times.

the gaze pointer is used to select items to interact with. You may have noticed it bounces around to attach to surfaces.

when it is over a road or a city, it selects that city.

You would need to select cities to use tools. Which we will now start describing. Notice that the gaze pointer changes every time you select a tool

press Lshift on the keyboard, x on the controller to open the tool menu and Lshift on the keyboard, x on the controller to close it.

with the tool menu open, use "+currentMap.lup+" and "+currentMap.ldown+" to move the gaze pointer

to the tool menu drop down and then press Rshift on the keyboard, A on the controller

to open the tool list. Press Rshift on the keyboard, A on the controller over a tool item to select it.

For now, select the 'name' tool

Then move it over a city and press Rshift on the keyboard, A on the controller. A box should appear with some text on it.

"this text is the design name (number). It represents the design and is what you should enter when asked to enter a design.

pressing Rshift on the keyboard, A on the controller again will hide the box and text.

q.askUntilValidArch("Input the design that you selected.

The 'name' tool will be very important for this experiment as it gives a unique identifier for the design.

another use for the name tool is to get the design names connected by a road.

try moving the gaze pointer cursor over a road (you will likely need to get close and look down) and then press "+currentMap.select)

you should see a box with text appear that will have three lines. The middle line will say 'is connected to'

the other two are names of designs which are connected by the road. These names are the same you would get by using the name tool on the cities.

q.askUntilTwoValidArch("input the (1) the top line and (2) the bottom line. Press Ctrl+C to go back

You may have noticed it can be slightly challenging to see precisely how the objectives change. The next two tools will help get more information

"about the objectives if you think that the perspective is making it hard to compare objective values.

bring up the tool menu and select the 'Height' Tool

Now move it over a city and press Rshift on the keyboard, A on the controller. You should see a label with colored boxes and numbers. The boxes correspond to objectives

"and the numbers are the actual values of the corresponding objectives.

this tool can be useful to verify if the objectives of a design satisfy a given set of requirements

let's try this. Use the 'name' tool to find the name of a design (doesn't matter which one)

q.refArch=q.askUntilValidArch("which design did you select?

obj=q.askUntilValidNum("what is the value of the objective 'blue3' for the design you just picked?

while not (np.abs((obj-q.refMatrics(objNames='blue3'))/obj)<1e-2):

incorrect, try again. Remember you used design "+str(q.refArch))

obj=q.askUntilValidNum("what is the value of the objective 'blue3' for the design you just picked?

Name and Height are very important tools for answering questions as they tell you precise information on the design(s)

q.cls()

the remaining tools are used to compare designs

open the tool menu and select the 'Compare' tool

the compare tool is used to compare the values of objectives visually.

move the gaze pointer over a design and press Rshift on the keyboard, A on the controller.

now if you look at the other cities, you should see black bars above and below the buildings.

the compare tool takes the heights of the buildings for the city you selected and draws the heights as rectangles

so you can see if the objectives are greater than, less than or equal to the values of another design.

the vertical posts are so that you can see which objective a bar corresponds to if the originally selected design has a greater value in

in that particular objective.

do you see bars for every design except the one you selected?"

[if 'yes':] As you can see. Compare compares all the cities except the one selected

[if 'no':] Try again. Compare compares all the cities except the one selected

now for the final tool

from the tool menu, select the 'link' tool

You may have noticed that the road network can be a bit of a mess and it can be hard to distinguish the values of roads.

the link tool is used to raise and lower cities and roads to better distinguish between roads and cities of interest.

With the 'link' tool selected move vertically up a bit and then, select a city

When selecting a city with the link tool, the roads connected to the city are raised to your current height and out of the birds nest that is the ground plane.

selecting the city again will cause the roads to reset to the original heights.

dontCare=q.askUntilValidForInt("how many roads were raised for the city you selected?

Now try using the link tool over a road that is still on the ground.

Using link on a road causes the connected cities and connecting road to rise to your current height.

what are the two designs that you raised? (don't forget the name tool!

[if incorrect:]that's not right. The roads should be drawn distance 2 so if you correctly read the cities, they should be distance 2 or smaller

now odds are you might end up with a large number of raised cities and roads. To bring everything back to the original height at the ground plane press r on the keyboard, z on the controller

very good. This is all the tools in Cityplot-VR

As a tip, you can also change the tools with the d-pad on the controller or the spacebar on the keyboard.

It might be easier than opening the tool menu.

very good. Now for the real experiment:

don't forget, roads are number of decision changes and building heights are objective values

TUTORIAL SCRIPT: RAVE

The following was displayed on the command prompt and each line was advanced by pressing any key. This tutorial was run before any tests were performed with the Cityplot tool.

You should see a number of windows appear. Type 'y' when you see one open with a grid pattern and a couple graphs

Welcome to the RAVE tutorial

RAVE is a tool that is intended to run and visualize design analysis.

It visualizes designs through a set of fairly standard plots

in the top left you should see a scatterplot--a set of dots positioned along 2 axes

scatterplots are the most basic visualization. Each dot is a design and each dot is placed according to the values of the design in the two selected axes

note that scatterplot axes can be decisions or objectives but you only get two at a time.

decisions are labeled with 'decision#' and objectives are given the form 'color#' for simplicity in this experiment

The default axes tend not to be terribly useful.

click around the edge of the scatterplot to select it.

if you look to the far left there should be a box with tabs out the left side. Click the 'view' tab.

you should now see a 'Data Series' group. Click the dropdown box for 'X-axis variable' and select cyan1

now click the dropdown for 'Y-axis variable' and select green2

you should see the dots move position as you change the variables.

Now try this question.

the spread of values for the green2 objective is higher for larger values of cyan1 than for smaller values of cyan1? (y/n)

[if incorreect (n):] incorrect. towards the lower values of cyan1, the dots have less vertical extent than for the higher values of cyan1--in other words there are more possible values of green2 for higher values of cyan1

Now for an important interaction. You can get the design name (number) by hovering your mouse over a data point. This will be what you key in when asked to pick or identify a design

Now try this question.

what is the name of the design with the smallest value of cyan1?

[if answered with invalid architecture:] try again, it's important to understand how to do this. If you are absolutely stuck, ask for help from teh experiment supervisor

[if answered with incorrect architecture:] try again

moving to the bottommost plot we get histograms.

histograms simply count the number of times a value (or close values) appears.

select the histogram by clicking around the edge of the plot

if you look back at the view tab that we used earlier you can change the variable selected

You can also draw multiple histograms by using ctrl+click or shift+click to select multiple variables at once

For example, try this question.

how many designs have a value of 'cyan1' near 0.5?

[if given incorrect answer:] incorrect. Try again. Be sure to use the histogram with the 'cyan1' variable selected.

finally, we have the parallel axis plot

the parallel axis plot is a relatively uncommon plot that lets you visualize arbitrarily high numbers of objectives.

select the parallel axis plot by clicking around the edge.

now on the 'view' tab use ctrl+click or shift+click to select more than one quantity to plot from choose variables.

you should see a number of vertical bars and lines crossing the vertical bars.

Each vertical bar is a variable that was selected and each line is a design. A given design (line) crosses a vertical bar at the value of the variable for that design.

for example, if a design has a value of 0.5 for cyan1 it will cross the cyan1 bar at 0.5 (which happens to be near the largest values of cyan1 in this example)

by tracing the line across the plot, you can view the values of multiple objectives or decisions for a given design

as with the scatterplot, you can hover over a line to get the design name (number) for the line and the line will highlight purple.

try playing with the plot a bit. try plotting all four objectives (cyan1, green2, blue3, red4) at once

Press 'y' when you are ready to continue.

now, the strength of RAVE lies in linked plots and filtering.

if you click on a design (a dot in the scatterplot or a line in the parallel axis plot) it will turn purple in all plots where that design is represented

you can also do this with multiple designs at once by holding ctrl or shift while you click or by clicking and dragging on a plot

try playing around with linked designs for a bit. Press 'y' when ready to continue.

finally for filtering

you may have noticed the small yellow triangles in the parallel axis plot or the histogram(s)

simply click and drag these arrows. The designs which do not have objective values between all the arrows will be greyed out.

this is useful for finding designs which are not close in particular quantities or finding designs which meet particular requirements.

WARNING: if you set a filter and then change variables in the parallel axis plot or histogram, you will have to re-display the variable to move the filter. You also cannot get design information from filtered designs.

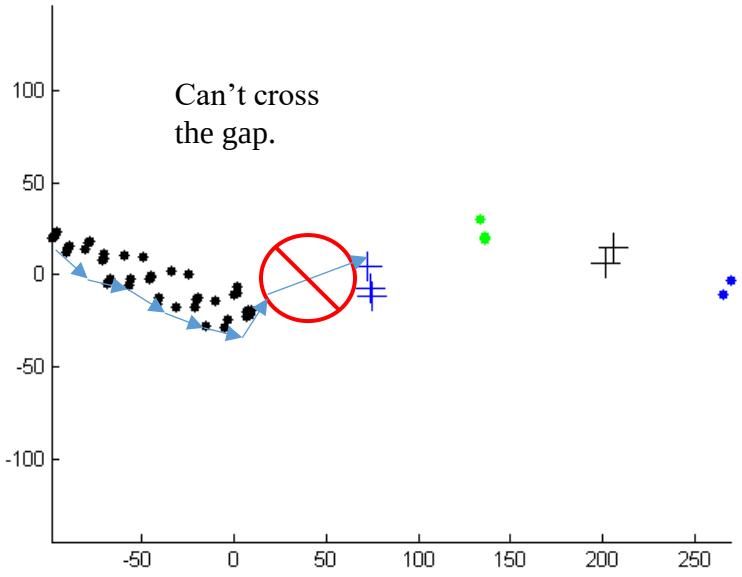
very good, now try the real experiment

don't forget decisions are labeled as decisions and objectives are given as a color and a number

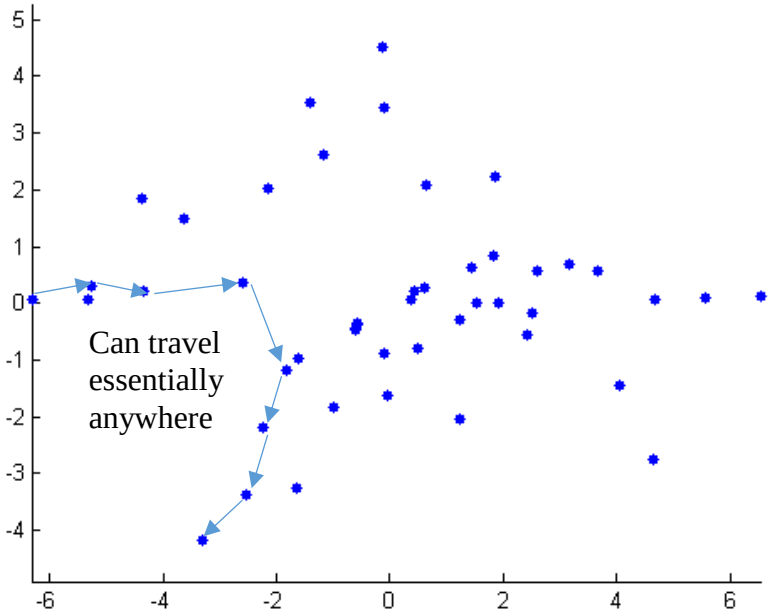
therefore, changes to decisions should be interpreted as decision variables being changed together.

DESIGN GROUPING DEFINITION PRINTOUT

A set of grouped points. Groups are indicated by different markers or colors. Observe that while the differences (corresponding to changes in design decisions or criteria) between adjacent designs are small within groups; there are big gaps between groups.



For contrast, this is a not-grouped set of designs. The changes needed to go from one design to another are roughly always even and it is possible to change from design to design to get between any pair of designs.



7 REFERENCES

Bibliography

(Luke) Zhang, Xiaolong, Tim Simpson, Mary Frecker, and George Lesieutre. 2012. “Supporting Knowledge Exploration and Discovery in Multi-Dimensional Data with Interactive Multiscale Visualisation.” *Journal of Engineering Design* 23 (1): 23–47. https://doi.org/10.1080/09544828.2010.487260.

Agarwal, Pankaj, and Kasturi Varadarajan. 2004. “A Near-Linear Constant-Factor Approximation for Euclidean Bipartite Matching ? ∗.” In *Symposium on Computational Geometry*. Brooklyn, NY: ACM. https://doi.org/10.1.1.72.3542.

Agrawal, Gautam, Sumeet Parashar, and Christina Bloebaum. 2006. “Intuitive Visualization of Hyperspace Pareto Frontier for Robustness in Multi-Attribute Decision-Making.” In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 1–14. Reston, Virigina: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2006-6962.

Allaire, Douglas L., and Karen E. Willcox. 2012. “A Variance-Based Sensitivity Index Function for Factor Prioritization.” *Reliability Engineering and System Safety* 107: 107–14. https://doi.org/10.1016/j.ress.2011.08.007.

Almeida, Adiel T. De, Cristiano A. V. Cavalcante, Marcelo H. Alencar, Rodrigo J. P. Ferreira, Adiel T. De Almeida-Filho, and Thalles V. Garcez. 2015. *Multicriteria and Multiobjective Models for Risk , Reliability and Maintenance Decision Analysis*. https://doi.org/10.1007/978-3-319-17969-8.

Andriani, P, and B McKelvey. 2007. “Beyond Gaussian Averages : Redirecting International Business and Management Research toward Extreme Events and Power Laws.” *Journal of International Business Studies* 38 (7): 1212–30. https://doi.org/10.1057/palgrave.jibs.8400324.

Avigad, Gideon, and Amiram Moshaiov. 2009. “Interactive Evolutionary Multiobjective Search and Optimization of Set-Based Concepts.” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 39 (4): 1013–27. https://doi.org/10.1109/TSMCB.2008.2011565.

Beltrame, Giovanni, and Gabriela Nicolescu. 2011. “A Multi-Objective Decision-Theoretic Exploration Algorithm for Platform-Based Design,” 4–7.

Benedikt, Michael. 1991. “Cyberspace: Some Proposals.” In *Cyberspace*, 119–224. Cambridge, MA: MIT Press.

Beyer, H, H Beyer, B Sendhoff, and B Sendhoff. 2007. “Robust Optimization – A Comprehensive Survey.” *Computer Methods in Applied Mechanics and Engineering* 196 (33–34): 3190–3218. https://doi.org/10.1016/j.cma.2007.03.003.

Borg, Ingwer, and Patrick J. F. Groenen. 2005. *Modern Multidimensional Scaling*. 2nd ed.

Springer Series in Statistics. New York, NY: Springer New York. https://doi.org/10.1007/0-387-28981-X.

Brooke, John. 1996. “SUS - A Quick and Dirty Usability Scale.” *Usability Evaluation in Industry*, 4–7. https://doi.org/10.1002/hbm.20701.

Cariboni, J., D. Gatelli, R. Liska, and A. Saltelli. 2007. “The Role of Sensitivity Analysis in Ecological Modelling.” *Ecological Modelling* 203 (1–2): 167–82. https://doi.org/10.1016/j.ecolmodel.2005.10.045.

Carterette, Ben, Paul N Bennett, David Maxwell Chickering, and T Susan. 2008. “Here or There Preference Judgments for Relevance.”

Chen, Wei, Ruichen Jin, and Agus Sudjianto. 2004. “Analytical Variance-Based Global Sensitivity Analysis in Simluation-Based Design under Uncertainty.” *Proceedings of DETC’04 ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* 127 (September 2005): 1–10. https://doi.org/10.1115/1.1904642.

Chiu, P-W., A.M. Naim, K. E. Lewis, and C. L. Bloebaum. 2009. “The Hyper-Radial Visualisation Method for Multi-Attribute Decision-Making under Uncertainty.” *International Journal of Product Development* 9: 4–31. https://doi.org/10.1504/IJPD.2009.026172.

Community, The SciPy. 2019. “Scipy.Optimize.” https://docs.scipy.org/doc/scipy/reference/optimize.html.

Crawley, Edward, Bruce Cameron, and Daniel Selva. 2016. *System Architecture: Strategy and Product Development for Complex Systems*. Hoboken, NJ: Prentice Hall.

Daskilewicz, Matt, and Brian German. n.d. “RAVE: An Open Source Mouse-Driven MATLAB Toolbox to Visualize, Explore, Sample, Optimize, Analyze and Model Data and Data-Generating Functions.” Oct. 29, 2015. Accessed November 17, 2015. http://www.rave.gatech.edu/index.shtml.

Decision Vis. n.d. “DiscoveryDV.” Accessed June 22, 2015. https://www.decisionvis.com/explorerdv/.

Denœud, Lucile. 2008. “Transfer Distance between Partitions.” *Advances in Data Analysis and Classification* 2 (3): 279–94. https://doi.org/10.1007/s11634-008-0029-0.

Dominguez-Garcia, Alejandro, Gregor Hanuschak, Steven Hall, and Edward Crawley. 2007. “A Comparison of GNC Architectural Approaches for Robotic and Human-Rated Spacecraft.” In *AIAA Guidance, Navigation and Control Conference and Exhibit*. Hilton Head, South Carolina: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2007-6338.

Drake, Doratha E, and Stefan Hougardy. 2003. “Improved Linear Time Approximation Algorithms for Weighted Matchings.” *Approximation, Randomization and Combinatorial Optimization, Lecture Notes in Computer Science* 2764: 14–23. https://doi.org/10.1007/978-3-540-45198-3_2.

Duan, Ran, and Seth Petti. 2010. “Approximating Maximum Weight Matching in Near-Linear Time.” In *51st Annual Symposium on Foundations of Computer Science*, 673–82. Las Vegas, NV: IEEE. https://doi.org/10.1109/FOCS.2010.70.

Duncan, Thomas J, and Judy M Vance. 2006. “Development of a Virtual Environment for Interactive Interrogation of Computational.” *Journal of Mechanical Design* 3 (Feb 22). https://doi.org/10.1115/1.2409314.

Duvenaud, David, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Lin Lin. 2013. “Structure Discovery in Nonparametric Regression through Compositional Kernel Search.” In *Proc. of the 30th International Conference on Machine Learning*. Vol. 28.

Eddy, John, and Kemper E Lewis. 2002a. “Multidimensional Design Visualization in Multiobjective Optimization.” *9th AIAA/ISSMO Symposium and Exihibit on Multidisciplinary Analysis and Optimization*, no. September: 1–11. https://doi.org/10.2514/6.2002-5621.

———. 2002b. “Visualization of Multidimensional Design and Optimization Data Using Cloud Visualization.” In *Volume 2: 28th Design Automation Conference*, 2002:899–908. Montreal, Canada: ASME. https://doi.org/10.1115/DETC2002/DAC-34130.

Ferguson, Philip M. 2019. “Clever Hans.” *Britannica Online*. Encyclopedia Britannica. https://www.britannica.com/topic/Clever-Hans.

Fredman, M.L., and R.E. Tarjan. 1987. “Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms.” *Journal Ofthe Association for Computing Machinery* 34 (3): 596–615. https://doi.org/10.1109/SFCS.1984.715934.

Grosse, Roger B, R. Salakhutdinov, William T Freeman, and Joshua B Tenenbaum. 2012. “Exploiting Compositionality to Explore a Large Space of Model Structures.” *Uncertainty in Artificial Intelligence*.

Gusfield, Dan. 2002. “Partition-Distance: A Problem and Class of Perfect Graphs Arising in Clustering.” *Information Processing Letters* 82 (3): 159–64. https://doi.org/10.1016/S0020-0190(01)00263-0.

Hanuschak, Gregor Z., Nicholas A. Harrison, Edward F. Crawley, Steven R. Hall, Alejandro D. Dominguez-Garcia, John J. West, and Cornelius J. Dennehy. 2009. “A Comparison of Fault-Tolerant GN&C System Architectures Using the Object Process Network (OPN) Modeling Language.” In *AIAA Guidance, Navigation and Control Conference*, 1–21. Chicago, IL: AIAA 2009-5676. https://doi.org/10.2514/6.2009-5676.

Haskins, Cecilia. 2006. “INCOSE Systems Engineering Handbook - A Guide for System Life Cycle Processes and Activities.” International Council on Systems Engineering.

———. 2019. “Trade Study to Select Best Alternative for Cable and Pulley Simulation for Cranes on Offshore Vessels.” *Systems Engineering*, no. June: 1–12. https://doi.org/10.1002/sys.21503.

Hastie, Trevor J, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. Springer Series in Statistics. New York, NY: Springer New York.

https://doi.org/10.1007/b94608.
Hazelrigg, G A. 1998. “A Framework for Decision-Based Engineering Design.” *Journal of Mechanical Design* 120 (4): 653–58. https://doi.org/10.1115/1.2829328.
Hedar, Abdel-Rahman. n.d. “Test Functions For Unconstrained Global Optimization: Perm Functions.” Accessed November 23, 2015. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2545.htm.
Ho, William, Xiaowei Xu, and Prasanta K Dey. 2010. “Multi-Criteria Decision Making Approaches for Supplier Evaluation and Selection: A Literature Review.” *European Journal of Operational Research* 202 (1): 16–24. https://doi.org/10.1016/j.ejor.2009.05.009.
Hocke, Raimund. 2018. “SikuliX.” http://sikulix.com/.
Hoffman, Patrick E, and Georges G Grinstein. 2001. “A Survey of Visualizations for High-Dimensional Data Mining.” In *Information Visualization in Data Mining and Knowledge Discovery*, edited by Usama Fayyad, Georges Grinstein, and Andreas Wierse, 47–86. San Diego, CA: Morgan Kaufmann Publishers.
Isenberg, Tobias, Petra Isenberg, Jian Chen, Michael Sedlmair, and Torsten Moller. 2013. “A Systematic Review on the Practice of Evaluating Visualization.” *IEEE Transactions on Visualization and Computer Graphics* 19 (12): 2818–27. https://doi.org/10.1109/TVCG.2013.126.
Jamil, Momin, and Xin She Yang. 2013. “A Literature Survey of Benchmark Functions for Global Optimisation Problems Xin-She Yang.” *International Journal of Mathematical Modelling and Numerical Optimisation* 4 (2): 150–94. https://doi.org/10.1504/IJMMNO.2013.055204.
Jung, Jae Hun, and Wolfgang Stefan. 2011. “A Simple Regularization of the Polynomial Interpolation for the Resolution of the Runge Phenomenon.” *Journal of Scientific Computing* 46 (2): 225–42. https://doi.org/10.1007/s10915-010-9397-7.
Kanukolanu, Deepti, Kemper E Lewis, and Eliot H Winer. 2006. “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* 6 (September): 288–99. https://doi.org/10.1115/1.2218370.
Kasprzyk, Joseph R, Patrick M Reed, Gregory W Characklis, and Brian R Kirsch. 2012. “Many-Objective de Novo Water Supply Portfolio Planning under Deep Uncertainty.” *Environmental Modelling and Software* 34: 87–104. https://doi.org/10.1016/j.envsoft.2011.04.003.
Kim, Pansoo, and Yu Ding. 2005. “Optimal Engineering System Design Guided by Data-Mining Methods.” *Technometrics* 47 (3): 336–48. https://doi.org/10.1198/004017005000000157.
Kuchinski, Michael J. 2000. “VERDICT : A Distributed Virtual Environment for System Engineering.” *Systems Engineering* 3 (3): 156–62.
Kuhn, H. W. 1955. “The Hungarian Method for the Assignment Problem.” *Naval Research Logistics Quarterly*, no. 2: 83–97.

Lloyd, James Robert, David Duvenaud, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. 2014. “Automatic Construction and Natural-Language Description of Nonparametric Regression Models.” In *Association for the Advancement of Artificial Intelligence*.

Madni, Azad M. 2014. “Expanding Stakeholder Participation in Upfront System Engineering through Storytelling in Virtual Worlds.” *Systems Engineering* 18 (1): 16–27. https://doi.org/10.1002/sys.21284.

Mareschal, Bertrand, and Jean-Pierre Brans. 1988. “Geometrical Representations for MCDA.” *European Journal of Operational Research* 34 (1): 69–77. https://doi.org/10.1016/0377-2217(88)90456-0.

Marler, R T, and J S Arora. 2004. “Survey of Multi-Objective Optimization Methods for Engineering.” *Structural and Multidisciplinary Optimization* 26 (6): 369–95. https://doi.org/10.1007/s00158-003-0368-6.

McAdams, Daniel A., Robert B. Stone, and Kristin L. Wood. 1999. “Functional Interdependence and Product Similarity Based on Customer Needs.” *Research in Engineering Design* 11 (1): 1–19. https://doi.org/10.1007/s001630050001.

McAdams, Daniel A., and Kristin L. Wood. 2002. “A Quantitative Similarity Metric for Design-by-Analogy.” *Journal of Mechanical Design* 124 (May 2000): 173. https://doi.org/10.1115/1.1475317.

Mehlhorn, Kurt, and Guido Schafer. 2002. “Implementation of O(Nmlogn) Weighted Matchings in General Graphs: The Power of Data Structures.” *Journal of Expirimental Algorithmics* 7: 4. https://doi.org/10.1145/944618.944622.

Montgomery, Douglas C. 2013. *Design and Analysis of Experiments*. 8th ed. Hoboken, NJ: John Wiley and Sons, Inc.

Nagel, Henrik R, Erik Granum, and Peter Musaeus. 2001. “Methods for Visual Mining of Data in Virtual Reality.” In *Proceedings of the International Workshop on Visual Data Mining*, 13–27.

Orsborn, Seth, Peter Boatwright, and Jonathan Cagan. 2008. “Identifying Product Shape Relationships Using Principal Component Analysis.” *Research in Engineering Design* 18 (4): 163–80. https://doi.org/10.1007/s00163-007-0036-8.

Pareek, Shrey, Vaibhav Sharma, and Ehsan T Esfahani. 2015. “Human Factor Study in Gesture Based CAD Environment.” In *Proc. of the ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Confrence*, 1–6. Boston, MA: ASME.

Plaisant, Catherine, Jean Daniel Fekete, and Georges Grinstein. 2008. “Promoting Insight-Based Evaluation of Visualizations: From Contest to Benchmark Repository.” *IEEE Transactions on Visualization and Computer Graphics* 14 (1): 120–34. https://doi.org/10.1109/TVCG.2007.70412.

Reed, P M, D Hadka, J D Herman, J R Kasprzyk, and J B Kollat. 2013. “Evolutionary

Multiobjective Optimization in Water Resources: The Past, Present, and Future." *Advances in Water Resources* 51: 438–56. https://doi.org/10.1016/j.advwatres.2012.01.005.

Reed, Patrick M, and Joshua B Kollat. 2008. "Parallel Evolutionary Multi-Objective Optimization on Large , Heterogeneous Clusters : An Applications Perspective" 5 (November): 460–78. https://doi.org/10.2514/1.36467.

Richardson, Trevor, and Eliot Winer. 2011. "Visually Exploring a Design Space Through the Use of Multiple Contextual Self-Organizing Maps." In *Volume 5: 37th Design Automation Conference, Parts A and B*, 857–66. Washington, DC, USA: ASME. https://doi.org/10.1115/DETC2011-47944.

Ross, Adam M, and Daniel E Hastings. 2005. "The Tradespace Exploration Paradigm." In *INCOSE International Symposium 2005*, 13. Rochester, NY. https://doi.org/10.1002/j.2334-5837.2005.tb00783.x.

Saltelli, Andrea, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, and Debora Gatelli. 2008. *Global Sensitivity Analysis . The Primer*. West Sussex, England: John Wiley and Sons.

Sammon, John W. 1969. "A Nonlinear Mapping for Data Structure Analysis." *IEEE Transactions on Computers* C–18 (5): 401–9. https://doi.org/10.1109/t-c.1969.222678.

Sanfeliu, Alberto, and King Sun Fu. 1983. "A Distance Measure Between Attributed Relational Graphs for Pattern Recognition." *IEEE Transactions on Systems, Man and Cybernetics* SMC-13 (3): 353–62. https://doi.org/10.1109/TSMC.1983.6313167.

Scikit-learn Developers. n.d. "Sklearn.Manifold.MDS." Accessed June 22, 2015. http://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html.

Selva, B.G. Cameron, and E.F Crowley. 2017. "Patterns in System Architecture Decisions." *Systems Engineering* 19: 477–97.

Selva, Daniel. 2012. "Rule-Based System Architecting of Earth Observation Satellite Systems." Massachusettes Institute of Technology.

Selva, Daniel, Bruce G Cameron, and Edward F Crawley. 2014. "Rule-Based System Architecting of Earth Observing Systems: The Earth Science Decadal Survey." *Journal of Spacecraft and Rockets*. https://doi.org/10.2514/1.A32656.

Selva, Daniel, and David Krejci. 2013. "Preliminary Assessment of Performance and Cost of a Cubesat Component of the Earth Science Decadal Survey." In *27th Annual AIAA/USU Conference on Small Satellites*.

Seriai, Abderrahmane, Omar Benomar, Benjamin Cerat, and Houari Sahraoui. 2014. "Validation of Software Visualization Tools: A Systematic Mapping Study." *Proceedings - 2nd IEEE Working Conference on Software Visualization, VISSOFT 2014*, 60–69. https://doi.org/10.1109/VISSOFT.2014.19.

Shimoyama, Koji, Jin Ne Lim, Shinkyu Jeong, Shigeru Obayashi, and Masataka Koishi. 2009. "Practical Implementation of Robust Design Assisted by Response Surface Approximation and Visual Data-Mining." *Journal of Mechanical Design* 131 (6): 61007.

https://doi.org/10.1115/1.3125207.
Sibson, R. 1973. “SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method.” *The Computer Journal*. https://doi.org/10.1093/comjnl/16.1.30.
Simpson, Timothy W. 2004. “Product Platform Design and Customization: Status and Promise.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 18 (1): 3–20. https://doi.org/10.1017/S0890060404040028.
Simpson, Timothy W., Jonathan R. Maier, and Farrokh Mistree. 2001. “Product Platform Design: Method and Application.” *Research in Engineering Design - Theory, Applications, and Concurrent Engineering* 13 (1): 2–22. https://doi.org/10.1007/s001630100002.
Simpson, Timothy W., and Joaquim R.R.A. Martins. 2010. “The Future of Multidisciplinary Design Optimization (MDO): Advancing the Design of Complex Engineered Systems.” *National Science Foundation, Arlington, VA, Tech. Rep*, 1–20. http://www.academia.edu/643586/The_Future_of_Multidisciplinary_Design_Optimization_MDO_Advancing_the_Design_of_Complex_Engineered_Systems%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Future+of+Multidisciplinary+Design+Optimization+(M.
Simpson, Timothy W, Daniel E Carlsen, and Christopher D Congdon. 2008. “Trade Space Exploration of a Wing Design Problem Using Visual Steering and Multi-Dimensional Data Visualization.” *AIAA Multidisciplinary Design Optimization Specialist Conference*, 1–18.
Simpson, Timothy W, and Joaquim R R a. Martins. 2011. “Multidisciplinary Design Optimization for Complex Engineered Systems: Report From a National Science Foundation Workshop.” *Journal of Mechanical Design* 133 (10): 101002. https://doi.org/10.1115/1.4004465.
Strickland, John. 2013. “Revisiting SLS/Orion Launch Costs.” *The Space Review*. http://www.thespacereview.com/article/2330/1.
Stump, Gary M., Mike a. Yukish, Jay D. Martin, and Timothy W. Simpson. 2004. “The ARL Trade Space Visualizer: An Engineering Decision-Making Tool.” In *Collection of Technical Papers - 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 5:2976–86. Albany, NY: American Institute of Aeronautics and Astronautics. https://doi.org/doi:10.2514/6.2004-4568.
Stumpf, Simone, Vidya Rajaram, Lida Li, Weng Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. “Interacting Meaningfully with Machine Learning Systems: Three Experiments.” *International Journal of Human Computer Studies* 67 (8): 639–62. https://doi.org/10.1016/j.ijhcs.2009.03.004.
The Mathworks. n.d. “Cmdscale: Classical Multidimensional Scaling.” Accessed June 22, 2015a. http://www.mathworks.com/help/stats/mdscale.html.
———. n.d. “Mdscale: Nonclassical Multidimensional Scaling.” Accessed June 22, 2015b. http://www.mathworks.com/help/stats/mdscale.html.
Tu, Tea, and Bogdan Filipi. 2016. “Showing the Knee of a 4-D Pareto Front Approximation via

Different Visualization Methods." In *Proc. of the 19th International Multiconference Information Society*, 52–55.

Unal, Mehmet, Gordon Warn, and Timothy W. Simpson. 2015. "Introduction of a Tradeoff Index for Efficient Trade Space Exploration." In *Volume 2B: 41st Design Automation Conference*, V02BT03A033 DETC2015-46895. Boston, MA: ASME. https://doi.org/10.1115/DETC2015-46895.

Unal Mehmet, Gordon Warn, and Timothy W. Simpson. 2016. Quantifying the Shape of a Pareto Front in Support of Many-Objective Trade Space Exploration. *ASME 2016 International Design Technical Conference and Computers and Information in Engineering Conference*. Charlotte, NC. https://doi.org/10.1115/DETC2016-597716

Walker, G R, P A Rea, S Whalley, M Hinds, and N J Kings. 1993. "Visualisation Of Telecommunications Network Data." *BT Technology Journal* 11 (4): 54–63.

Weck, Olivier L de. 2001. "Isoperformance Methodology for Precision Optomechanical Systems."

Wendrich, Robert E., Wadee Al-Halabi, David Ullman, Eric J. Seibel, Kris-Howard Chambers, Olaf Grevenstuk, and Hunter G. Hoffman. 2016. "Hybrid Design Tools in a Social Virtual Reality Using Networked Oculus Rift: A Feasibility Study in Remote Real-Time Interaction." In *Proceedings of the ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1–7. Charlotte, NC: ASME.

Woodruff, Matthew J., Patrick M. Reed, and Timothy W. Simpson. 2013a. "Many Objective Visual Analytics: Rethinking the Design of Complex Engineered Systems." *Structural and Multidisciplinary Optimization* 48 (1): 201–19. https://doi.org/10.1007/s00158-013-0891-z.

Woodruff, Matthew J, Patrick M Reed, and Timothy W Simpson. 2013b. "Many Objective Visual Analytics: Rethinking the Design of Complex Engineered Systems." *Structural and Multidisciplinary Optimization* 48 (1): 201–19. https://doi.org/10.1007/s00158-013-0891-z.

Xing, Li-Ning, Ying-Wu Chen, and Ke-Wei Yang. 2009. "An Efficient Search Method for Multi-Objective Flexible Job Shop Scheduling Problems." *Journal of Intelligent Manufacturing* 20 (3): 283–93. https://doi.org/10.1007/s10845-008-0216-z.

Yeh, T P, and J M Vance. 1998. "Applying Virtual Reality Techniques to Sensitivity-Based Structural Shape Design." *Journal of Mechanical Design* 120 (December 1998).

Yin, Xiaolei, and Wei Chen. 2008. "A Hierarchical Statistical Sensitivity Analysis Method for Complex Engineering Systems Design." *Journal of Mechanical Design* 130 (7): 071402. https://doi.org/10.1115/1.2918913.

Young, Peter. 1996. "Three Dimensional Information Visualisation." *Computer Science Technical Report No. 12/96*. https://doi.org/10.1.1.54.3719.

Zeidler, Eberhard. 1995a. *Applied Functional Analysis; Main Principles and Their Applications*. Vol. 109. Applied Mathematical Sciences. New York, NY: Springer New York. https://doi.org/10.1007/978-1-4612-0821-1.

———. 1995b. *Applied Functional Analysis: Applications to Mathematical Physics*. Vol. 108. Applied Mathematical Sciences. New York, NY: Springer New York. https://doi.org/10.1007/978-1-4612-0815-0.