High-dimensional Design Space Visualization For Conceptual Structural Design

by

Caitlin T. Mueller

B.S. in Art and Design, Department of Architecture
Massachusetts Institute of Technology, 2007

M.S. in Structural Engineering, Department of Civil and Environmental Engineering
Stanford University, 2008

Submitted to the School of Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computation for Design and Optimization

at the

Massachusetts Institute of Technology

June 2014

Signature of Author: Signature redacted
School of Engineering
May 19, 2014

Certified by: Signature redacted
John A. Ochsendorf
Professor of Civil and Environmental Engineering and Architecture
Thesis Supervisor

Accepted by: Signature redacted
Nicolas G. Hadjiconstantinou
Professor of Mechanical Engineering
Co-Director, Computation for Design and Optimization (CDO)

High-dimensional Design Space Visualization for Conceptual Structural Design

by

Caitlin T. Mueller

Submitted to the School of Engineering
on May 19, 2014 in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computation for Design and Optimization

Abstract

This thesis focuses on visualizing high-dimensional design spaces for early-stage design problems in structural engineering and related disciplines. The design space, which is defined as the $n + 1$-dimensional surface that relates n design variables to a performance metric, contains all possible solutions to a formulated design problem. Graphical views of the design space are highly useful for designers because they organize a wide range of design possibilities in a compact, intuitive, and logical manner, illuminating global patterns, variable behaviors and relationships, and the nature of paths taken during iterative design processes. Design problems with two or fewer variables can easily be visualized in Euclidian space, through a curve or surface, but high-dimensional problems are difficult to display graphically. This is the key challenge addressed in this thesis.

The thesis includes a critical review of existing methods for high-dimensional design space visualization, highlighting the unmet needs across a range of approaches. In response to these needs, the thesis makes a key contribution in the form of a new design space visualization method, called isoperforming parallel coordinate clusters (IPC clusters), that overcomes the issues of previous techniques. The IPC cluster approach is demonstrated on several conceptual structural design problems, and its application in optimization, directed exploration, and related design strategies is illustrated. Finally, the thesis concludes with a discussion of applications, impact, and future research directions.

Key words: design space visualization, conceptual design, structural design, structural optimization

Thesis supervisor: John A. Ochsendorf
Title: Professor of Civil and Environmental Engineering and Architecture

Acknowledgements

This thesis benefitted from the helpful feedback, guidance, and general assistance of several mentors and colleagues. First, I thank my thesis advisor, Professor John Ochsendorf, for his enthusiasm, encouragement, and tough questions during the development of this work. I also acknowledge the input of several additional faculty members, particularly Professor Karen Willcox and Professor Olivier de Weck, who provided thoughtful feedback at multiple points during my time in the CDO program, and who inspired aspects of this work in their multidisciplinary system design optimization class.

For their administrative support and guidance, I am grateful to the Computation for Design and Optimization program administrators: Laura Koller, Barbara Lechner, and Kate Nelson. Finally, I thank my parents, Mark and Liz Mueller, and my husband, Martijn Stevenson, for their love and support.

Table of Contents

List of Mathematical Symbols

SYMBOL	MEANING
A	Cross-sectional area of a structural member
d_c	Minimum Euclidian distance between cluster centroids in IPC cluster plot
$f(\boldsymbol{x})$	Objective function evaluation of the performance of design $\boldsymbol{x}$
I	Moment of inertia of a structural member
k	Number of clusters generated using the k-means clustering algorithm
k_i	Number of clusters generated at ith isoperformance level
m	Number of design data points used to generate surrogate model
M	Number of approximate data points generated using surrogate model
n	Number of design variables
n_p	Number of isoperformance levels to include in IPC cluster plot
n_c	Maximum number of clusters to generate in IPC cluster plot
p_i	ith isoperformance level
p_{max}	Maximum isoperformance level in IPC cluster plot
$\boldsymbol{X}$	Data set containing variations of design point x used to create design space visualization
$\widetilde{\boldsymbol{X}}$	Approximate data set containing variations of design point $\boldsymbol{x}$ and predicted performance values
$\boldsymbol{X}_{iso,i}$	Data set containing variations of design point $\boldsymbol{x}$ at ith isoperformance level
$\boldsymbol{x}$	Design vector containing all individual design variables
$\bar{\boldsymbol{x}}_i$	Centroid design of ith cluster of designs
x_i	ith design variable
γ	Parametric angle defining the geometry of a Michell truss
δ	Step size between isoperformance levels in IPC cluster plot
ε	Tolerance used to identify isoperforming designs in IPC cluster plot

CHAPTER 1

Problem Statement

This thesis focuses on visualization of high-dimensional design spaces for conceptual design, with an emphasis on applications in structural engineering. Visualization is an important tool for designers as a way to clarify complicated design problems in an intuitive, organized, and graphical manner. However, the question of how to effectively visualize design problems with three or more variables remains open. This thesis aims to address this question with a critical review of existing visualization methods (Chapter 2), the introduction of a new visualization method (Chapter 3), and a series of case studies that demonstrate its use in conceptual design problems (Chapter 4). The following chapter provides further motivation and context for this problem.

1.1 Design space visualization (DSV)

Engineering design has traditionally made use of graphical aids that connect design decisions to performance implications, such as the plot shown in Figure 1.1, which illustrates the impact of height and section shape on the axial load capacity of a steel wide-flange column. Such graphics are an example of design space visualization (DSV). They are useful because they allow a designer to quickly understand a design problem as a system of parameterized alternatives with varying performance.

Formally, the design space is a multidimensional mapping of design variables (sometimes called decision variables or parameters) to a quantitative performance metric. The space can be defined by analytical

expressions, or by a black-box simulation or network of simulations. In the case of an analytically defined problem, equations are a more abstract way to understand the design space, but can still provide important global information. In the case of a simulation-based space, it is very difficult to understand global performance without a visualization of the design space.

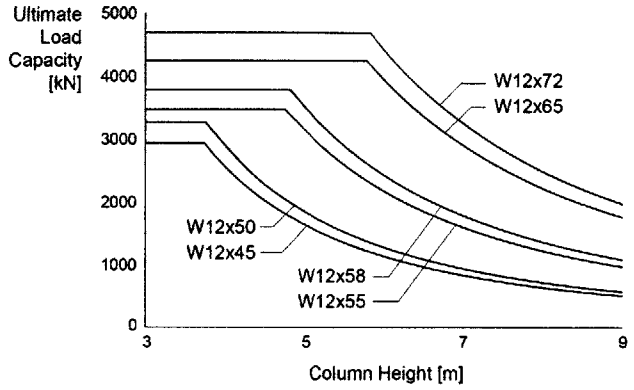


Figure 1.1: *Traditional design space plot showing the relationship between the design variable, column height, and performance, ultimate load capacity, for a variety of wide-flange steel sections.*

Simulation-based performance evaluation has become increasingly common in structural and other engineering design disciplines due to increases in available computational speed. While this type of numerical analysis is able to capture complexity and accuracy better than analytical representations in many cases, a side effect is that designers have a harder time understanding patterns and global behavior in design problem systems. DSV methods can help address this issue by graphically illustrating how simulated design alternatives connect and compare. Such information can help engineers considerably in conceptual design exploration and optimization.

This thesis deals specifically with the challenges of visualizing simulation-based high-dimensional design spaces, or design spaces with three or more variables, which cannot be displayed graphically using conventional plotting techniques. The following sections introduce and motivate this topic in greater detail, and provide an overview of the needs and goals of DSV techniques.

1.2 Design space vs. objective space

First, it is important to distinguish the scope of this thesis, the design space, from other visualizations associated with engineering design, specifically the objective space. While visualizations of the design space illustrate the relationship between design variables and one performance metric, the objective space illustrates how design alternatives trade off in terms of multiple competing performance objectives. Objective spaces are useful to visualize multi-objective optimization problems, common in many engineering disciplines, where increased performance in one objective can lead to decreased performance in another. Visualizations of objective spaces often highlight the Pareto frontier, the group of designs at the edge of this tradeoff.

An example of the design space and objective space for a design problem with one variable and two objectives is shown in Figure 1.2. The design problem involves a simply supported seven-bar truss made of hollow steel tubes of a fixed cross section, shown in Figure 1.2a. The design variable, the vertical position of the noted truss joint, impacts two performance metrics: the structure's weight and its load capacity, computed based on allowable stress and local buckling considerations. These two relationships are visualized in the design spaces in Figure 1.2c. The tradeoff between the objectives is visualized in the objective space in Figure 1.2b.

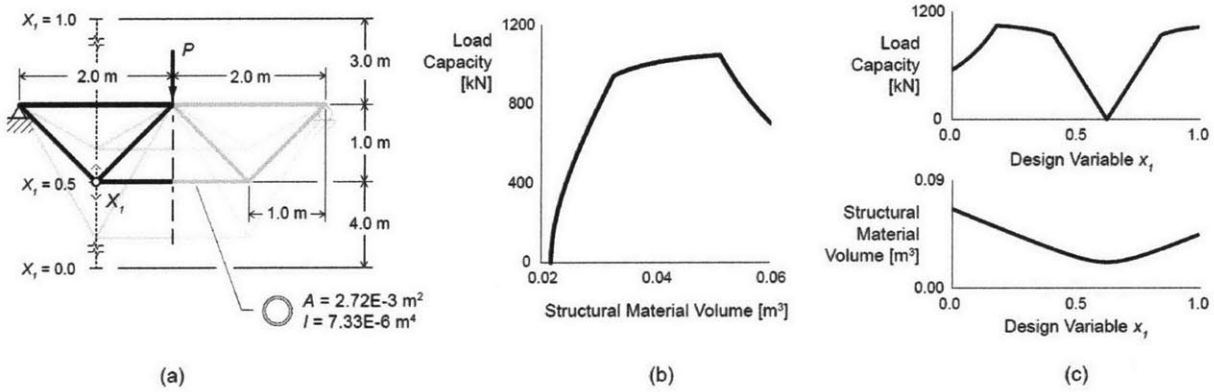


Figure 1.2: *Two types of visualizations of the two-objective, one-variable design problem in shown in (a): (b) the objective space, showing the tradeoff between two objectives, and (c) the design spaces, showing the relationships between the design variable and the objectives.*

While both types of visualization provide useful information to designers, DSV relates more directly to conceptual design decisions because of its inclusion of design variables. Furthermore, objective space visualization has been well-studied compared to DSV (Stump et al., 2003; Blasco et al., 2007; Kollat & Reed, 2007; Jordan et al., 2008; Carlsen et al., 2008). This thesis will therefore address the DSV of single-objective problems only.

1.3 Benefits of DSV

Visual representations of the design space help engineers understand a problem in an organized, systematic way that can help inform the critical conceptual decisions that carry through for the rest of the design process. Specifically, this thesis identifies and addresses three key areas in which DSV can potentially offer advantages:

1. Global characteristics and patterns: DSV offers a global overview of a design problem in terms of variables and performance, and can help answer questions about the number and relative performance of global minima, the flatness of high-performing design space regions, continuity and singularities, and relationships between variables. This information can inform designers about families of designs to consider more closely, and those to avoid.

2. Individual behavior of design variables: Beyond global behavior, DSV can also convey information about the relative importance and behavior of individual design variables. For example, which variables contribute significantly to changes in performance, and which matter less? Which

variables must be set to particular values for reasonable performance? The answers to these questions can help designers simplify and reformulate the design problem so that good solutions are clearer and easier to find.

3. Exploration and optimization information: While the first two benefits help designers prepare for design space exploration and optimization approaches, DSV can also be used during exploration and optimization to better understand these processes. In both cases, visualization shows the designers how considered designs connect to each other, and how direct or meandering the path toward a selected solution is. This information can feed back into the exploration and optimization processes to improve their performance.

While numerical methods have been established to address some of these issues as well, this paper nevertheless argues that visualization offers additional qualitative information unavailable through purely numerical means. DSV can therefore serve as a complement to numerical data and feedback about the design space, enhancing the designer's understanding of complex problems and better equipping him or her to solve them.

More broadly, DSV can be seen as a means to facilitate the "Design by Shopping" paradigm, a design approach formalized by Balling (1999). This idea is motivated by the need of designers to consider many alternative options, prior to formalizing their design goals in the strict, numerical manner required by traditional optimization. In contrast with optimization, the shopping approach aims to present designers with a catalog of options and affiliated prices (i.e. performance). Existing research on the implementation of the shopping approach has concentrated on multi-objective problems and Pareto-optimal solutions (Stump et al., 2003). However, single-objective design problems can also be explored using a shopping-like approach, and DSV methods are a compelling way to creative visual catalogs of options for designers.

1.4 Challenges of high-dimensional design spaces

A two-variable design problem and its resulting design space are shown in Figure 1.3. The problem again involves a simply supported seven-bar truss, but this time the cross sectional areas are not fixed. The two variables are the horizontal and vertical coordinates of the lower left node, and like all variables in this thesis, are projected to the [0, 1] range. The objective function computes the structural material volume required to support the given load, calculated as a summation of member lengths multiplied by member areas. The member areas are based on internal forces, and consider both allowable stress and Euler buckling in their sizing. They are assumed to be made out of steel tubes with a wall thickness set to 5% of the diameter.

This problem is easily visualized in three-dimensional Euclidean space as a surface, and designers can intuitively understand the relationship between the two design variables and the performance metric through the DSV. All three of the benefits listed in Section 1.2 apply here, as illustrated in Figure 1.4:

1. Globally, there are two minima, one of which performs slightly better than the other. The space is discontinuous when x_2 is set between the two minima, leading to very poor performance (Figure 1.4a).

2. The setting of x_1 does not significantly affect performance compared to x_2, and could be eliminated as a variable (Figure 1.4b).

3. Depending on the starting position, a gradient-based optimization approach can find the optimal solution relatively efficiently. An evolutionary exploration approach is able to identify a range of high-performing solutions (Figure 1.4c).

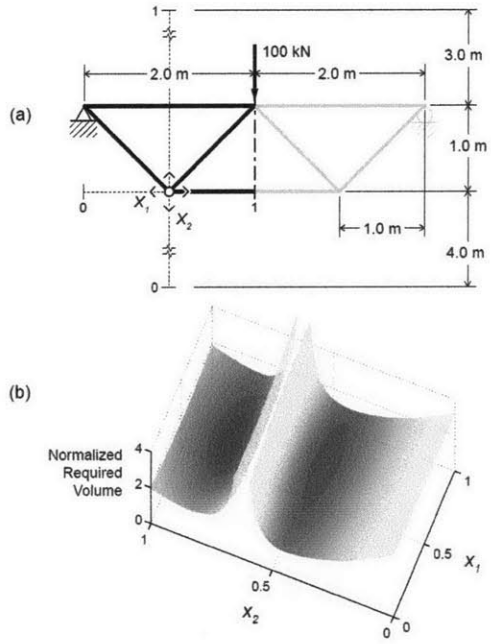


Figure 1.3: *A two-dimensional design problem (a) and its design space (b), which can be visualized in 3D Euclidean space.*

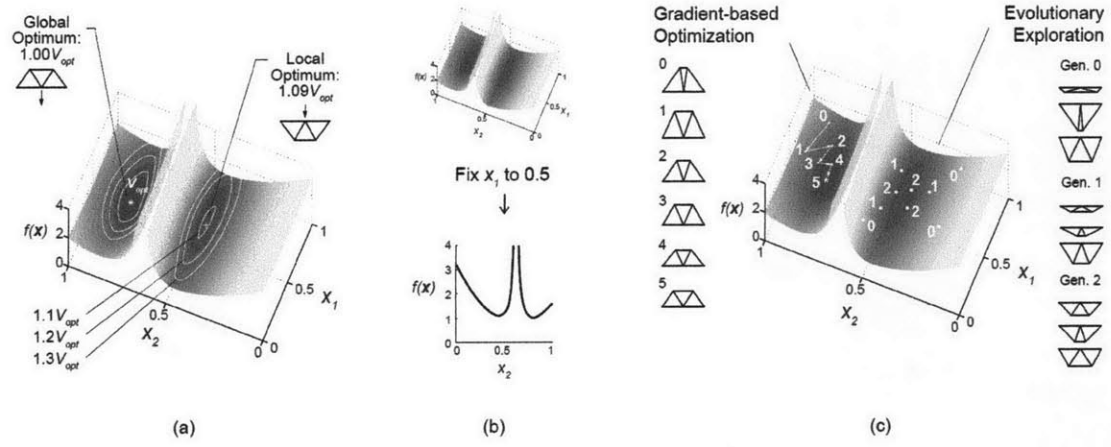


Figure 1.4: *Conceptual design aided by the design space visualization from Figure 1.3.*

Problems with more than three variables can also benefit from DSV, but cannot be visualized as conventional 2D curves or 3D surfaces. This challenge requires less familiar visualization techniques that represent four or more dimensions on a two-dimensional paper or screen. A second obstacle is the so-called "curse of dimensionality," which refers to the fact that higher-dimension problems require more data points to be represented thoroughly. For example, a two-variable design space can be sampled at 10 settings for each variable with 10^2 samples, but a six-variable space requires 10^6 samples for the same amount of coverage. This means that techniques that work for low-dimensional problems may not be practical for high-dimensional versions.

1.5 Organization of thesis

This thesis aims to address these challenges by presenting a new DSV technique for design problems with three or more variables that achieves the three benefits listed in Section 1.2. Chapter 2 critically reviews existing methods for high-dimensional design space visualization, and establishes specific unmet needs in this area. Chapter 3 introduces the new DSV method, called *isoperforming parallel coordinate clusters* (IPC clusters) and details its implementation and use. Chapter 4 gives several examples of how this method can be used in the conceptual design exploration and optimization of structures. Finally, Chapter 5 concludes with a discussion of potential impact and areas for future work.

CHAPTER 2

Existing multidimensional DSV methods

Several classes of methods have already been developed for visualizing high-dimensional data, some related to design spaces specifically, and some more general. This section provides a critical overview of these existing methods, summarizes their individual benefits and drawbacks, and shows that none resolves all of the needs established in Section 1.2. To fairly compare the presented existing methods, each is illustrated for the three-variable design problem shown in Figure 2.1, which modifies the problem introduced in Figure 1.3 by adding one additional variable, and again uses required structural material volume as the performance metric.

It is important to note that the following examples intentionally distinguish variation in performance from variation in design variables, which is a key difference between DSV and more general multidimensional data visualization problems. In some cases, there is a clear and built-in way to highlight variation in performance. In other cases in which no standard exists, strategies such as color intensity (darker color indicates higher performance) are used as consistently as possible.

2.1 Custom plots for specific problems

The most traditional method for high-dimensional visualization is to create custom-designed plots for individual problems, such as the graphic shown in Figure 1.1. This method typically uses a two-dimensional graph, with one design variable along the horizontal axis, and the performance metric along the vertical axis. The additional variables are represented by plotting "families" of design alternatives, distinguished by line style,

color, marker type, etc. Many design guides developed for specific structural problems use this approach (Samyn, 2004; Allen & Iano, 2012). Figure 2.2 illustrates a custom plot DSV for the problem introduced in Figure 2.1. As shown in the figure, only x_2 is represented continuously, and the other two variables must be shown at discrete values.

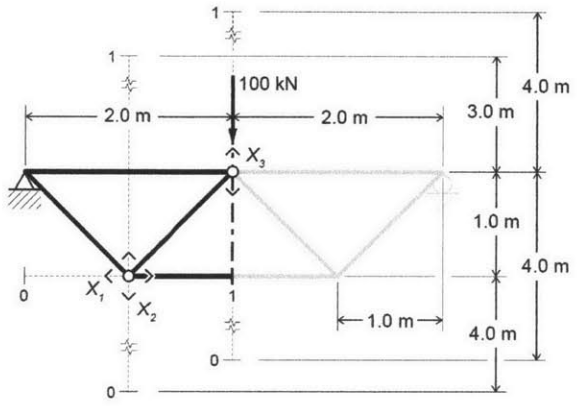


Figure 2.1: *A 3-dimensional design problem that cannot be visualized in Euclidean space.*

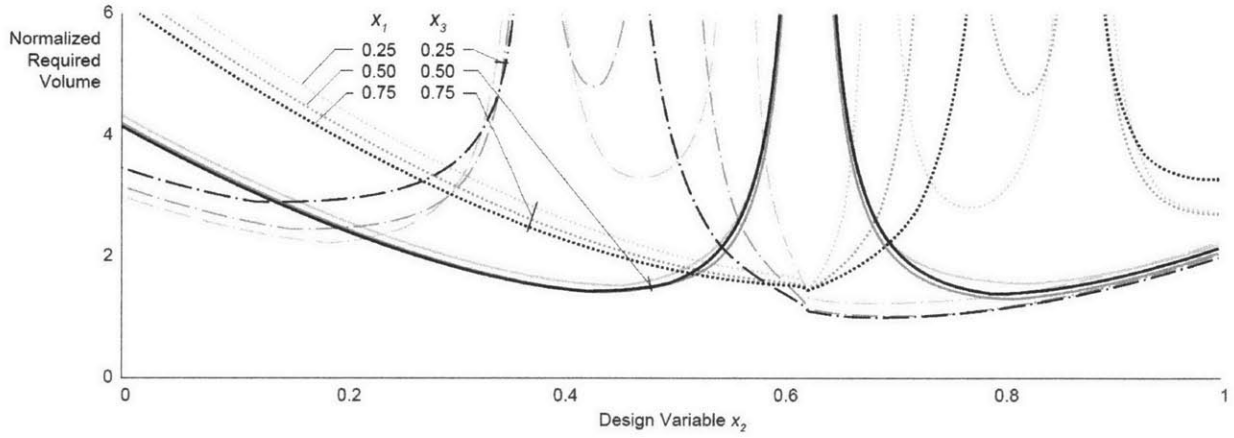


Figure 2.2: *Custom DSV for the problem shown in Figure 2.1.*

Custom plots work very well as DSV for certain design problems, especially when one variable is an obvious choice for the horizontal axis. This variable typically contributes most strongly to changes in performance, while additional variables lead to smaller changes, resulting in design families. There are several difficulties with this feature. First, it requires the designer to already know which variable is most important prior to creating the DSV, instead of allowing the DSV to provide this information. Second, the visualization becomes difficult to interpret when several variables are similarly important or have a strong relationship. In the case of Figure 2.2, both x_2 and x_3 are important, resulting in a somewhat crowded and unintuitive graphic. Finally,

because the additional variables are only shown at discrete values, it is easy to miss patterns in their behavior or interesting regions of the design space. These drawbacks limit the applicability of custom plots as a systematic DSV method for high-dimensional design problems.

2.2 Projection plots

Projection-based plots overcome the limitations of custom plots by being problem-independent in their implementation. This family of DSVs shows the design space through multiple two-dimensional views of design points, or projections onto two-dimensional planes. A popular version of this approach for n-variable design problems creates an n-by-n matrix of projected scatterplots that show the space projected onto each pairwise combination of variables (Carr et al., 1987; Monmonier, 1989; Elmqvist et al., 2008). Individual points can be colored based on performance. A second type of projection shows each design variable plotted against the performance metric individually. An effective combination of these approaches places these performance projections on the diagonals of the n-by-n matrix. This DSV concept is shown in Figure 2.3a for the design problem introduced in Figure 2.1.

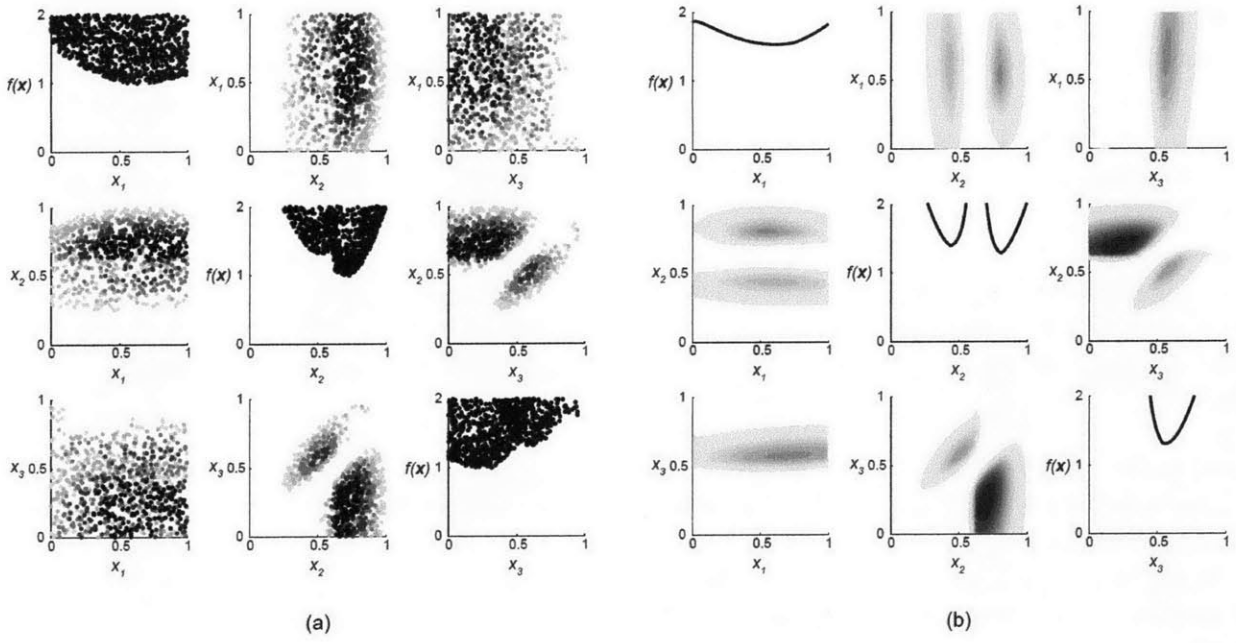


Figure 2.3: *Two projection-based DSVs for the problem shown in Figure 2.1: (a) projected scatterplots, and (b) projected slice plots cut at baseline variable settings. In the nondiagonal plots, darker colors indicate better performance.*

The advantages of the projected scatterplot DSV method is that it is completely systematic, and that it shows the entire design space, unlike the custom plot approach. However, there are two significant disadvantages to this approach. First, many of the projected scatterplots are cloudy and lack obvious patterns, due to the importance in the variables that have been compressed in the projection. In Figure 2.3a, this is the case in the plots between x_1 and x_2, and between x_1 and x_3 (the plots between x_2 and x_3 are clear because x_1 is a relatively

unimportant variable in this problem). Second, because the diagonal performance scatterplots do not communicate relationships between variables, they can create the mistaken impression that certain variable combinations perform much better than they do.

A second type of projection-based DSV uses contour plots instead of scatterplots. These avoid the cloudiness of scatterplots by setting the unused variables in pairwise plots to baseline values, resulting in a clean and familiar contour plot (Forrester et al. 2008). A similar approach can be used on the diagonal plots, resulting in a sharp line instead of a scatterplot silhouette. This approach is exemplified in Figure 2.3b. The projected slice plot method improves upon scatterplots in terms of interpretability with a high cost: a much smaller slice of the design space is represented, making it likely that important or interesting regions are overlooked. Furthermore, any insights about variable relationships and settings are only valid for the small slices of the design space that are displayed. The criticism about the diagonal plots remains an issue as well: they are deceptive in their disregard for variable relationships.

2.3 Parallel and radial coordinates

The problems with projection-based DSVs arise because they can only show partial views into the design space at a time. A separate class of DSV avoids this issue by moving beyond traditional Euclidian spatial representations. The most common of these is the parallel coordinates visualization technique, which places a series of parallel axes assigned to variables in a row and represents a multidimensional data point with lines between the axes connecting individual variable settings (Inselberg, 1985; Inselberg, 2009). This method was developed for data visualization generally instead of specifically for DSV, but it can be used for design spaces by signifying performance by the line color. Figure 2.4a shows this approach applied to the problem in Figure 2.1. A second DSV of this type places the variable-based axes in a polar array, instead of in a row, and a single design is represented as a closed polygon (Draper et al., 2009). This thesis calls these visualizations radial coordinates plots for consistency, although they are also sometimes referred to as spider plots, radar plots, star plots, or polar plots. Figure 2.4b shows the design space of the problem in Figure 2.1 using a radial coordinate plot.

These methods link variables with lines across axes that are arranged in parallel or radially, and a single design is represented by a series of linked lines. This method is therefore very good for showing relationships between variables, and can accommodate a very high number of variables. However, there are several drawbacks to coordinate-based methods. First, the visual representation and interpretation is highly dependent on the order of the variable axes, which is generally arbitrary. Some implementations try to correct for this problem by allowing users to rearrange axes in interactive graphics, but this solution is not possible in static displays on paper or screens. Second, in radial coordinates plots, the closed polygons representing individual designs seem to convey important information via their enclosed area, which is not actually meaningful in DSV applications. Parallel coordinates do not have this problem because of their geometry, which makes them to more appropriate choice of the two. Third, although performance can be signified by line color, it is still sometimes difficult to interpret the relative performance of the full design space, especially when there are a lot of overlapping lines. Due to these issues, coordinate-based methods are not sufficient for DSV in their existing state.

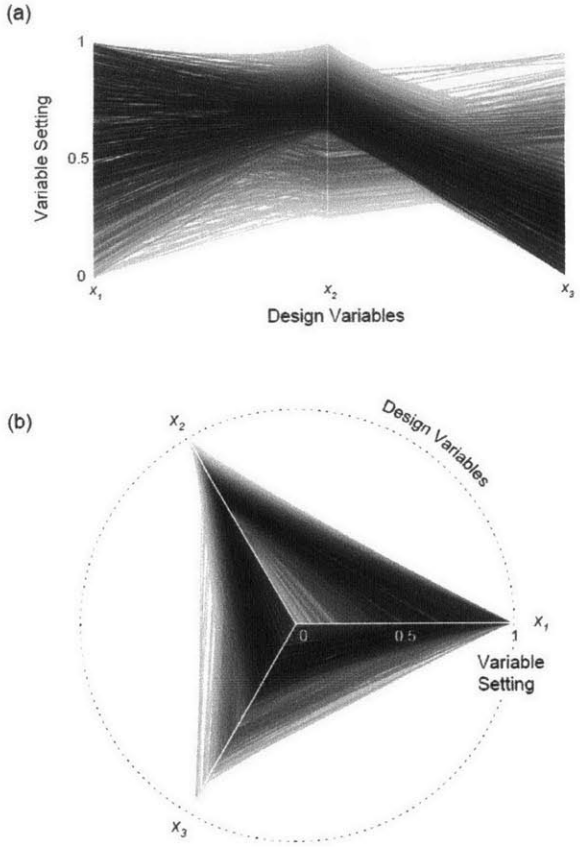


Figure 2.4: *Two coordinate-based DSVs for the problem shown in Figure 2.1: (a) parallel coordinates plot, and (b) radial coordinates plot. In both cases, darker colored lines indicate designs that perform better.*

2.4 Chernoff faces and other glyph plots

Other visualization methods attempt to overcome the limitations of Euclidean space through more unusual means. A prominent method in high-dimensional general data visualization is the glyph plot, in which each data point (or design) is represented by a single instance of a graphic, called a glyph. Visual features of the glyph vary based on variable settings. One of the most well-known glyph plots is the Chernoff face plot, so called because each glyph is a drawing of a human face (Chernoff, 1973). Up to seventeen variables can be captured in this representation, represented by facial features such as forehead size, chin shape, and space between eyes. This method tries to take advantage of the special human ability to distinguish small differences in faces, so that a high number of variables are legible. To apply the Chernoff face plot to DSV, color and ordering of the glyphs can be used to denote performance. Figure 2.5a shows the Chernoff face plot DSV for the problem introduced in Figure 2.1. In this example, x_1 is represented by face size, x_2 by forehead/jaw relative arc length, and x_3 by forehead shape (MathWorks, 2014).

For design problems in which the design can be represented clearly at a small scale, glyph plots can also be generated that use an image of the design itself as the glyph. This approach, called the literal glyph plot in this thesis, is illustrated in Figure 2.5b. This has the strong advantage of removing a layer of abstraction in variable interpretation. However, many design problems are difficult to display in this way, due to large geometric size, three-dimensionality, non-geometric variables, or other issues.

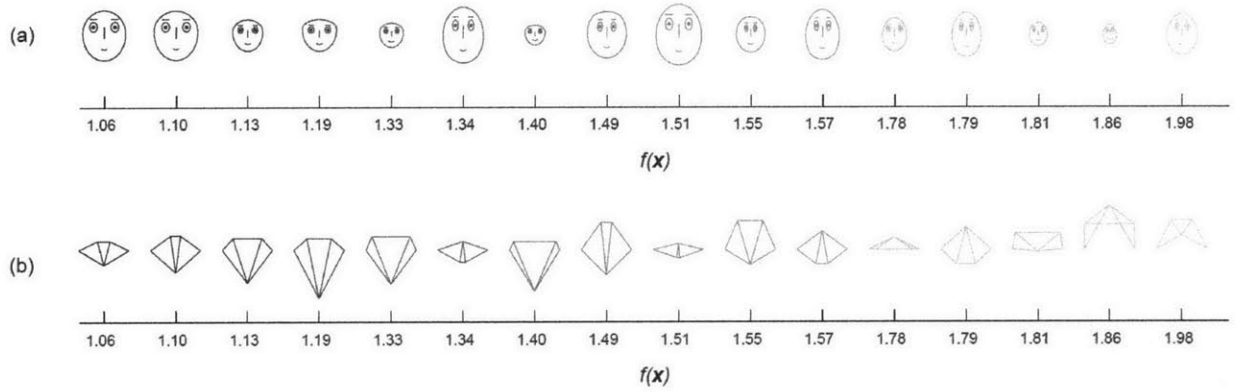


Figure 2.5: *Two types of glyph-based DSVs for the problem shown in Figure 2.1: (a) Chernoff face plot, which represents design variable settings with facial features, and (b) Literal glyph plot, which shows images of the varying designs themselves. In each case, both the order and color indicate performance.*

Both Chernoff face and literal glyph plots are limited by the number of glyphs that can be displayed and a lack of organization in terms of layout. Only a very small portion of the design space can be shown practically, and the designs cannot be arranged with any logic related to variable settings. Because of these drawbacks, they are not an appropriate DSV method for high dimensional problems.

2.5 Movement, interaction, and linked plots

Due to the limitations of the reviewed plotting techniques, a considerable amount of research has focused on improving the way a designer views these visualizations. Several compelling tools have been developed that provide dynamic, interactive environments for users to explore the design space actively. Interaction offers considerable advantages over static visualization, because the user can have full access to the design space but only visualize a small part of it at once, reducing complexity and focusing the graphics. Tools that allow 3D projected spaces to be rotated by the user, for example, use movement to compensate for the illegibility of many projected views (Swayne et al., 1998; Young et al., 2011). Tools that allow the user to "shop" through the design space catalog, by clicking on point in the design space and gaining more information, keep excessive information from overwhelming the user all at once (Simpson & Meckesheimer, 2004). Finally, tools that incorporate multiple visualizations, such as scatterplots and parallel coordinates plots, balance out the limitations of the individual techniques, and link designs across plots to show the same point in multiple views simultaneously (Ribarsky et al., 1994; Stump et al., 2003).

Tools of this type are important and potentially very useful to designers. However, nearly all make use of the existing, conventional DSV plotting techniques reviewed in this chapter. Although these tools use interactivity to address their limitations to some degree, there is still a need for better underlying plots to move through, interact with, and link between. The scope of this thesis focuses on this need by developing a new static DSV method that works well even without these dynamic features. Despite the widespread use of computers, meaningful static visualizations still play an important role in books, scholarly papers, and design documentation, and are much more enduring in the current climate of ever-changing technologies. Nevertheless, the work presented here is compatible with movement, interaction, and linked views, and could be implemented in a design tool with such features in the future, as discussed in Chapter 5.

2.6 Summary: limitations of existing methods

This chapter has critically reviewed a rage of existing plotting techniques for high-dimensional data, applied to the problem of design space visualization. The strengths and weaknesses of the seven plot types discussed in this chapter are summarized in Table 2.1, which rates the techniques in terms of design space coverage, performance representation, variable relationship representation, legibility, and extensibility.

DSV Method	Figure	Design Space Coverage	Performance Representation	Variable Relationship Representation	Legibility	Systematic and Extensible
Custom Plot	2.2	Poor	Strong	Medium	Strong	No, problem-dependent
Projected Scatterplot	2.3a	Strong	Strong	Poor	Medium	Yes
Projected Slice Plot	2.3b	Poor	Strong	Poor	Strong	Yes
Parallel Coordinates Plot	2.4a	Medium	Poor	Strong	Medium	Yes
Radial Coordinates Plot	2.4b	Medium	Poor	Medium	Poor	Yes
Chernoff Face Glyph Plot	2.5a	Poor	Medium	Poor	Poor	Up to 17 variables
Literal Glyph Plot	2.5b	Poor	Medium	Poor	Strong	No, problem-dependent

Table 2.1: *Comparison of existing DSV methods in terms of key goals.*

None of the methods perform well in all five categories, and custom plots, projected slice plots, radial coordinates plots, and Chernoff face plots are especially weak. The remaining three methods, projected scatterplots, parallel coordinate plots, and literal glyph plots, each have important strengths individually, and can perform well in all five categories in combination. Based on this analysis, the next chapter proposes a new DSV method that borrows from these three methods, combining them in a new way to overcome their individual limitations.

CHAPTER 3

Isoperforming Parallel Coordinate Clusters

Based on the limitations of existing approaches summarized in Chapter 2, this chapter proposes a new approach for DSV called *isoperforming parallel coordinate clusters* (IPC clusters). This new approach combines projected scatterplots, parallel coordinates plots, and literal glyph plots to show the full design space with an emphasis on performance, links between variables, and global patterns. The chapter begins by explaining the theoretical basis and conceptual overview of the approach, then describes the implementation details, and finally demonstrates how the approach can be used to achieve the goals of DSV.

3.1 Theoretical basis

Before introducing IPC clusters in detail, it is important to review the theoretical basis of the new approach. The IPC clusters approach makes use of three key theoretical developments, briefly described as follows:

1. *Principle of Small Multiples* (Tufte, 1983; Tufte, 1990): This fundamental concept in data visualization, formalized by Edward Tufte, argues for representing data in a series or matrix of similar thumbnail-sized graphic elements. This strategy is economical in that the user must invest in understanding only one graphic, and in return gains access to a richer range of information. Because the data is spread out over a number of views, each view can be relatively simple and clear. The parallel representation invites comparison and contrast between views, which is critical in conceptual design. Some existing

DSV methods make use of small multiples (such as the projection plots discussed in Section 2.2), but there is room for more work in this area.

2. *Isoperformance* (de Weck & Miller, 2002; de Weck & Jones, 2006): Isoperforming, or performance invariant, design options are those that are equally attractive (or unattractive) in terms of a quantitative objective. This concept is important because it addresses the frequent need to consider a diversity of alternatives in conceptual design, as opposed to just the single optimum. Additionally, this concept relates to *satisficing*, an idea developed by Herbert Simon (1956) that expresses the compromises and tradeoffs between a range of quantitative and qualitative goals in design and related disciplines. Isoperforming designs can be represented as contours in Euclidean views of three-dimensional design spaces, but are more difficult to visualize in high-dimensional problems, and there is little existing work in this area.

3. *Cluster Analysis* (Anderberg, 1973; Kaufman & Rousseeuw, 1990): The statistical grouping of data, often called cluster analysis, is a well-established approach in a variety of scientific fields. Algorithms that group data points according to features can reveal and clarify underlying patterns, relationships, and typologies. In conceptual design, it is often helpful to think about alternatives as members of design families as a means for organizing and comparing a broad variety of options. Despite the opportunities to identify design families via clustering, few existing methods for design space visualization or conceptual design more broadly make use of cluster analysis in this sense (one notable exception is given in Zhang et al. (2012), which suggests clustering design data at different scales).

These three ideas form the foundation of the IPC cluster method for design space visualization. In this new method, the design space is shown through an array of small multiples, each of which displays a subset of the design space. These multiples are organized by both isoperformance level and design family cluster. The small multiples themselves are an extension of parallel coordinates plots, modified to incorporate ideas from projected scatterplots and literal glyphs. The following section gives specific details about the method and its implementation.

3.2 Conceptual overview

The IPC clusters approach begins with a version of the parallel coordinates plot that uses line color to signify performance, as shown in Figure 2.4a. In order to represent variable setting on the horizontal axis, the familiar location for design variables in most plots based on Euclidian space, the plot is rotated 90 degrees. Next, the line-based axes are replaced by projected views of the design space for each variable, like the projection plots shown on the diagonals in Figure 2.3. This provides critical information about the shape of the design space in a compact manner. The line segments that represent each design point still connect between each of the parallel coordinates, but now are shifted vertically based on their performance. Multiple designs that differ in variable settings but perform at the same level, or isoperforming designs, can be understood in a single plot, as can multiple designs that are similar in variable settings but different in performance, called design family clusters. Figure 3.1 summarizes this new type of plot.

The IPC cluster approach displays subsets of the design space with this type of visualization, in an organized arrangement of many small views. The small multiples are laid out in a matrix, again like the projection-based methods, but the rows and columns have different meanings and are independent of variables. Figure 3.2 shows this layout conceptually, which is described in the following paragraphs.

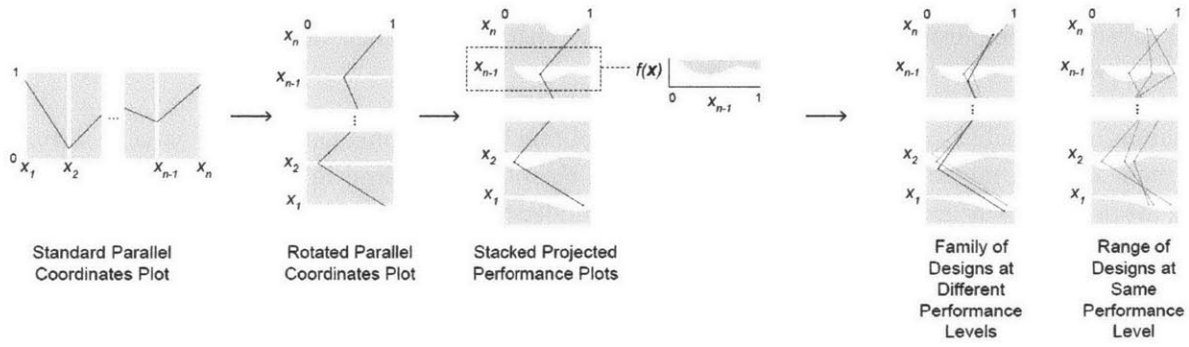


Figure 3.1: *Transformation of the standard parallel coordinates plot to the new parallel coordinate cluster plots, which include a projected silhouette of the design space instead of an axis for each variable.*

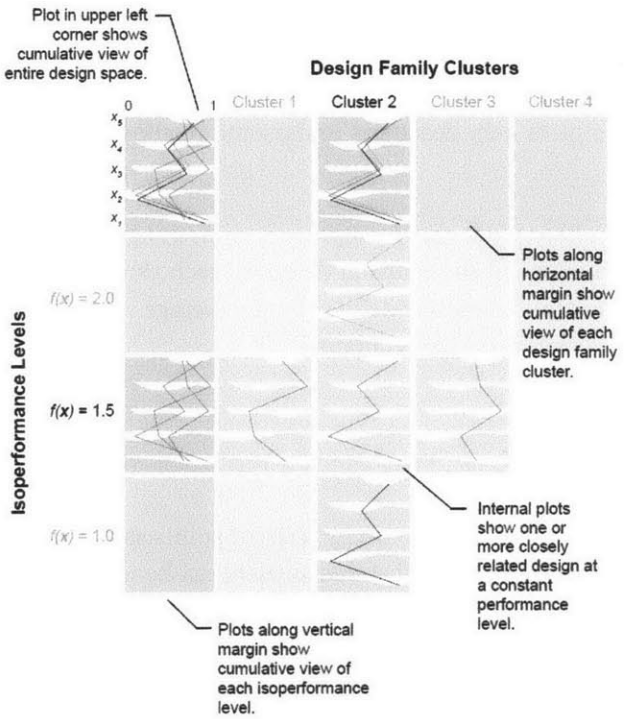


Figure 3.2: *Schematic overview of the isoperforming parallel coordinate (IPC) clusters technique for DSV, which uses multiple displays of the parallel coordinate clusters introduced in Figure 3.1 for different performance levels and design families*

The vertical arrangement of the small multiples corresponds to performance. This has the advantage of familiarity, since traditional DSVs for one or two variables also generally use the vertical axis to convey performance. Each row signifies a group of isoperforming designs, with the lowest row performing the best (in the case of a minimization problem). In traditional Euclidian DSV techniques, isoperforming designs are represented by contour lines, as in Figure 1.4a, which have the advantage of conveying the shape of the design space and the range of design alternatives at a particular level. The isoperforming rows in the IPC cluster method have the same purpose: they show the breadth and character of design variation available at a given performance value.

The columns of the matrix are arranged according to design family clusters that have similar variable settings. The clusters are determined using a statistical clustering algorithm, described in detail in the next section. Varying the design family along the horizontal axis can be understood as a generalized version of variation in a single design variable, which also occurs along the horizontal axis in many familiar plots. Similar clusters are vertically stacked, so that the relationship between similar design ideas at different performance levels can be understood. The number of clusters in each row is the same or greater than that of the row below, reflecting the widening of options that occurs as performance requirements are relaxed. The precise geometry of this widening depends on the specific character of the design problem, and can be understood graphically through the IPC cluster plotting technique.

The top and left margins of this matrix of small multiples contain additional cumulative views that compress each row or column into a single visualization. For example, the far left plot of a particular row shows all of the designs contained in the individual isoperforming cluster plots at once. Similarly, the top plot of a particular column shows all of designs contained in the related family cluster plots at once. In the top left corner, a summary plot shows the entire design space in a single view. Although they are generally too dense to include on their own, these cumulative plots help designers understand relationships between small multiples when presented alongside them.

For design problems in which alternatives can be represented meaningfully by small glyphs, the IPC cluster approach suggests including an image of the centroid design of each cluster as an overlay. Since clustering is based on variable settings, the centroid design is defined by the average variable settings of the cluster, and is a fair mean representation of the group of designs contained.

3.3 Implementation details

To generate the IPC cluster plot, the user must specify a set of points sampled from the design space, $\boldsymbol{X}$, and several parameters so that isoperforming and clustered subsets can be identified. These parameters are p_{max}, the maximum performance value to display; n_p, the number of isoperformance levels to include; ε, the tolerance to use in identifying isoperforming designs; n_c, the maximum number of clusters to consider; and d_c, the minimum Euclidian distance between clusters. Based on these parameters, the subsets are identified in three steps, as summarized in Figure 3.3: (1) Identify isoperforming subsets, (2) Partition subsets into clusters, and (3) Sort clusters so they align across performance levels. These steps are explained in more detail below.

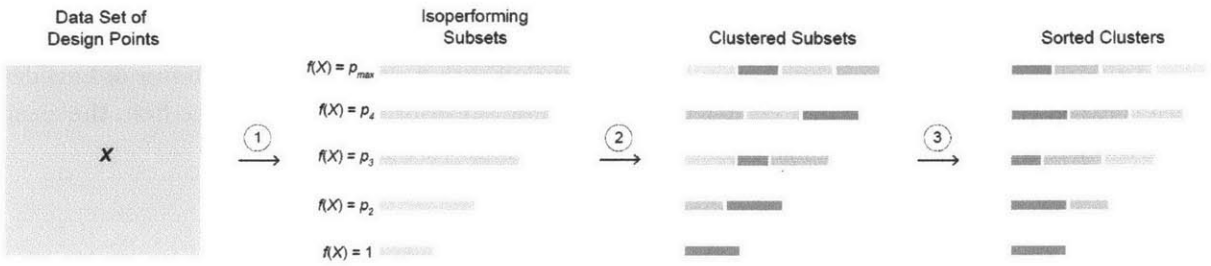


Figure 3.3: *Overview of implementation steps for partitioning design space data into sorted isoperforming clusters for the IPC cluster method.*

Step 1 identifies isoperforming levels (performance levels are normalized by the best performer, so the lowest performance score is always 1.00), and then finds designs that perform at those levels within a specified tolerance. The simplest way to identify these isoperforming designs is by sorting and searching through the data set of design points; this method is shown below. However, more sophisticated methods for identifying isoperforming designs have been proposed in the literature (de Weck & Miller, 2002), and could also be used.

1. Read $\boldsymbol{X}$, p_{max}, ε, and n_p
2. Compute the performance interval, $\delta = \frac{p_{max}-1.00}{n_p-1}$
3. Loop over i from 1 to n_p:
    a. Define the ith performance level, $p_i = 1.00 + (i-1)\delta$
    b. Identify the set of isoperforming designs: $\boldsymbol{X}_{iso,i} = \{\boldsymbol{x} \mid \boldsymbol{x} \in \boldsymbol{X}, p_i - \varepsilon \leq f(\boldsymbol{x}) \leq p_i + \varepsilon\}$

Step 2 partitions each isoperforming subset such that the clusters are far enough apart, so that the number of clusters remains constant or increases as performance levels increase, and so that the number of clusters does not exceed n_c. This step uses the k-means clustering algorithm, a method that partitions data into groups based on Euclidian distance (MacQueen, 1967), using the design vector of variable settings for each data point (performance is not included for clustering). The number of clusters, k, is an input argument for most implementations of this algorithm, including the MATLAB command `kmeans` used for this work (MathWorks, 2014), and is determined iteratively in the IPC cluster method, as shown below.

1. Read n_c and d_c
2. Loop over i from 1 to n_p:
    a. Define the number of clusters starting with the maximum value: $k_i = n_c$
    b. Compute k_i clusters of $\boldsymbol{X}_{iso,i}$ using the k-means algorithm
    c. Is the smallest distance between the centroids of the clusters less than d_c, and is $k_i > 1$ and is $k_i \neq k_{i-1}$?
        i. If yes: decrement k_i by 1, return to step 2a
        ii. If no: stop and save clusters for ith isoperformance level

Step 3 sorts the clusters at each isoperforming level so that design families are connected in vertical columns, which is necessary because the k-means algorithm returns clusters in an arbitrary order. It is not guaranteed that corresponding clusters will be represented at each isoperformance level, but in practice it is observed that strong relationships often emerge. The clusters are sorted such that that cluster closest (in terms of Euclidean distance between centroids) to the first cluster in the preceding isoperforming level is place first, the cluster closest to the second is placed second, and so on. The details for Step 3 are given below.

1. Loop over i from 2 to n_p:
    a. Create a new blank list of sorted clusters at the ith isoperformance level
    b. Loop over each cluster j of $\boldsymbol{X}_{iso,\,i-1}$, with j ranging from 1 to k_{i-1}:
        i. Identify the cluster of $\boldsymbol{X}_{iso,\,i}$ whose centroid is closest to centroid $\bar{\boldsymbol{x}}_{i-1,j}$ and add to the back of the sorted clusters list
    c. Add any clusters of $\boldsymbol{X}_{iso,\,i}$ not yet included to the back of the sorted clusters list

It should be noted that an alternate order of steps is also possible in theory: first, partition the design space into clusters, and then identify isoperforming designs across clusters. However, in practice, this order was not very successful in identifying reasonable cluster families.

3.4 Interpreting the design space with IPC clusters

In terms of the metrics compiled in Table 2.1, the new IPC cluster visualization method has strong coverage of the design space, a strong representation of both performance and variable settings, and is systematic and fairly extensible. Legibility depends on the viewer's familiarity with the method, and does require some investment to gain a deep understanding of the visualization. However, despite the novelty of the IPC cluster method, it maintains several familiar features, such as the horizontal and vertical arrangement of design variables and performance implications, respectively.

This section illustrates how the new visualization method can be used to gain a global understanding of patterns and characteristics of the design space. Figure 3.4 shows the IPC cluster plot method applied to the 3-variable problem introduced in Figure 2.1, with annotations calling out key observations. This visualization was created using the following parameter values: $p_{max} = 2$, $n_p = 5$, $\varepsilon = 0.02$, $n_c = 5$, and $d_c = 0.30$. The result is a plot with five isoperforming rows and five clustered columns, plus the cumulative row and column in the margins.

The bottom of the plot shows the top-performing cluster, which has a relatively tight range for all three design variables. Moving up along the column, the suboptimal clusters become wider in terms of variable settings, indicating they contain a broader range of designs. The defining feature of designs in Cluster 1 is a negative slope between x_2 and x_3, meaning that x_2 is generally high and x_3 is generally low, and high x_1 values. As shown in the glyph plots of the centroid designs, this means that the node associated with x_2 is above the supports, and the node associate with x_3 is below the supports. In the cumulative view of Cluster 1, it can be observed that the slope between x_2 and x_3 remains mostly constant across different performance levels, but shifts to the right as performance decreases. This means that the nodes in question both shift upwards in designs with lower performance (which leads to compression elements with longer buckling lengths).

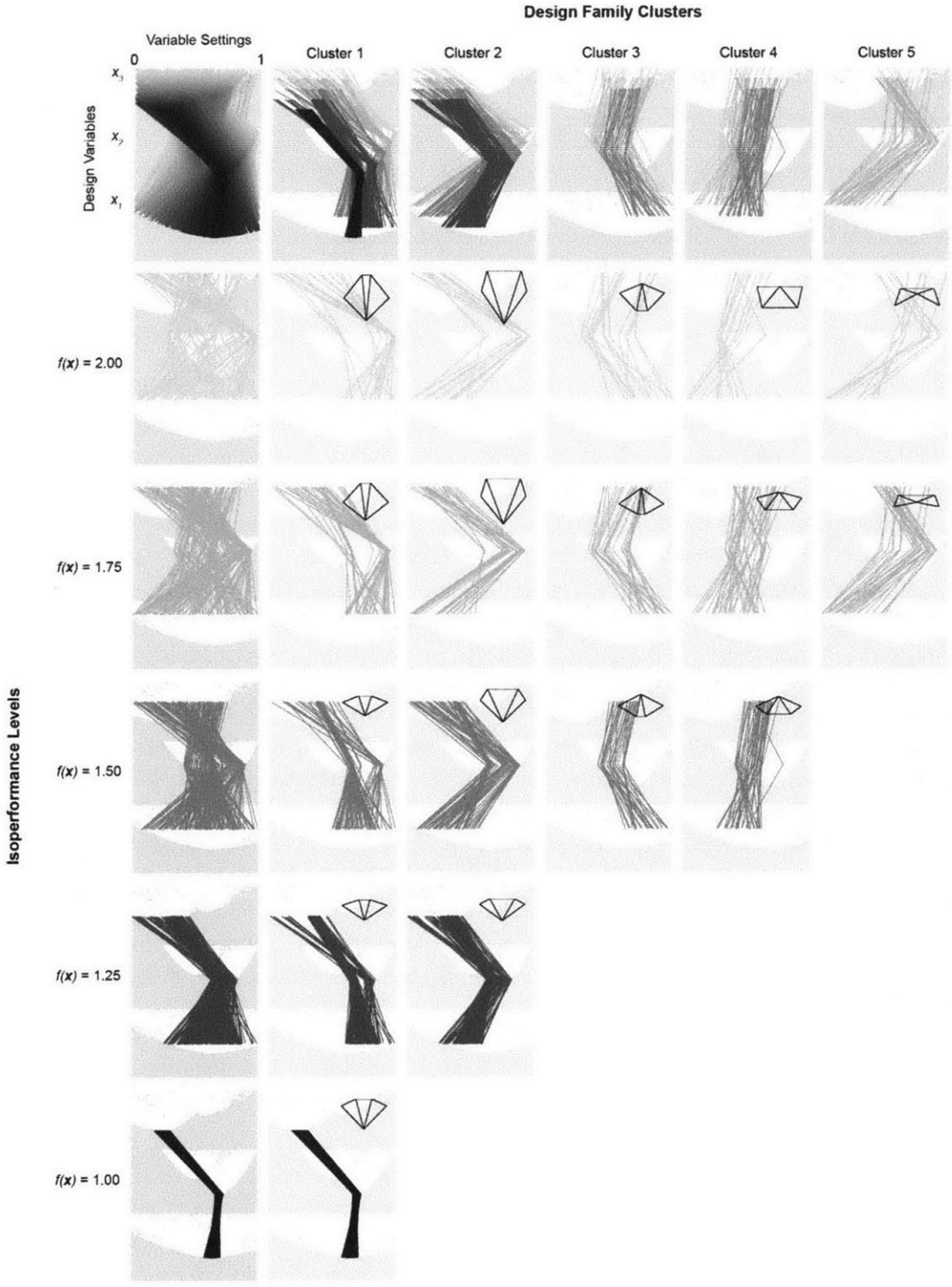


Figure 3.4: *IPC cluster visualization for the problem introduced in Figure 2.1.*

The best-performing designs in Cluster 2 also have a negative slope between x_2 and x_3, indicating a similar type of truss with the central diagonal elements forming a "V" shape. Cluster 2 differs from Cluster 1 almost entirely in the values of x_1 only, which are lower than those of Cluster 1. This leads to a wider "V" shape. As performance decreases, the difference in values between x_2 and x_3 becomes more extreme, resulting in deeper trusses with longer members.

At the third isoperformance level, $f(\mathbf{x}) = 1.50$, two additional clusters appear. Unlike Cluster 1 and Cluster 2, Cluster 3 and Cluster 4 have a positive slope between x_2 and x_3, meaning that x_2 is low and x_3 is high, and the truss diagonals form inverted "V" shapes. From the projected design space silhouettes, it is clear that these clusters are in a different region of the design space with a separate local minimum. Like Cluster 1 and Cluster 2, these two clusters are very similar except for values of x_1, leading to narrow inverted V-shapes for Cluster 3, and wide shapes for Cluster 4. As performance decreases for these two clusters, the trusses get deeper and the nodes generally shift down.

Finally, Cluster 5 appears at isoperformance level $f(\mathbf{x}) = 1.75$. This cluster is less compact and consistent than most of the other clusters, but generally shows low values for x_1, mid-to-high values for x_2, and mid-to-high values for x_3. This leads to trusses with a very wide upright "V" shape that have all nodes higher than the supports. This design strategy is not common and is generally illogical, since the truss is shallowest where it requires the most depth, at midspan. However, it is interesting to note that other designs at the same performance level, especially in Cluster 4, appear to be familiar and reasonable options. This IPC cluster method is able to reveal these types of unexpected relationships and behaviors in a way not possible with existing methods.

In addition to comparing designs within clusters, it is also valuable to look at design options across isoperformance levels. These are equal-cost options, so the amount of variation represents the degree of design freedom or choice available. As expected, the best performing options have the least variation, and diversity of options increases as performance decreases. However, it is interesting to examine the rate at which this tradeoff occurs, and to note which performance levels offer significant jumps in available options. In this problem, significant variation in x_1 is available at a performance 25% worse than the optimal design. A broader type of diversity that includes designs from a different design space region is available at a performance 50% worse than the optimal design. Depending on the goals of the designer, these options may be attractive despite their performance drawbacks. The IPC cluster method displays these tradeoffs in a clear and organized manner.

3.5 Understanding and simplifying variables with IPC clusters

In addition to understanding the general character and organization of the design space as a system, it is often also valuable to think about the behaviors of individual variables. Are their bounds set to appropriate levels, or should they be increased or decreased? If designs at acceptable performance levels are pushing up against the bounds of a particular value, the bounds should be increased to potentially discover more design options; conversely, if all of the designs of interest fall within a narrow range of settings for a particular variable, the bounds could be decreased to crop the design space to the area of interest.

It is also important to consider eliminating variables, to simplify the problem and reduce the dimension of the design space. Variables with a very narrow range at acceptable performance levels could potentially be fixed to a value in this range. Variables with a very wide range at acceptable performance levels could also be fixed, not according to performance, but in accordance with qualitative design goals. In this case, the wide variation indicates broad design freedom. Variables could also be eliminated due to correlation with other variables in regions of the design space of interest. This indicates that the variable has a consistent behavior relative to the other variable that could be formalized as a parametric relationship, eliminating the variable as an independent design decision. Variable importance and elimination have been studied extensively from a numerical perspective, but the IPC cluster method provides a new and potentially more intuitive way to understand variable graphically.

For the problem introduced in Figure 2.1 and visualized in Figure 3.4, there are three variables to be considered. The variable with the most variation in the highest performing regions of the design space is x_1, which varies at least twice as much as the other two variables at $f(\boldsymbol{x}) = 1.25$ (seen in the cumulative plot in the left margin). This variable controls the horizontal position of its node, thereby affecting whether the "V" shape formed by the diagonals is wide or narrow. The high variation in x_1 indicates that this design variable is not very important in terms of performance, and therefore be set according to aesthetic preferences if they exist. Eliminating x_1 would significantly simplify the design problem, reducing the variable count from 3 to 2 and allowing a conventional Euclidean design space visualization to be used. This would also bring the interesting relationship between x_2 and x_3 into greater focus.

It is also noteworthy that x_2 exhibits the least variation within the performance levels considered, from the optimal cluster at $f(\boldsymbol{x}) = 1.00$ up through $f(\boldsymbol{x}) = 2.00$. At the worst performance levels, the bounds for x2 are too wide, and the lowest possible values for x_2 are not represented at all. This suggests that the lower bound could be increased to crop the design space to a more relevant region, which would make exploration and optimization easier. Another approach would be to fix x_2 to its central value in the optimal cluster. This would again simplify the design problem by reducing the dimension of the design space. Unlike eliminating x_1, which reflects design freedom, eliminating x_2 would reflect design restriction; it is not worth exploring variation in x_2 since it must be set to a narrow range for reasonable performance.

Unlike numerical methods for understanding variable importance, the graphical approach enabled by IPC cluster plots provides this type of insight into variables at a global level, across the whole design space, while still providing detailed information about local behavior. In contrast, numerical representations of variable importance must be either specific to a particular point in the design space (such as partial derivatives), or aggregate representations that blur local behavior (such as means and other statistical moments). The IPC cluster method offers an alternative or complement to these methods in a new, graphical manner.

3.6 Visualization for optimization and exploration

In addition to providing a general overview of a design problem and conveying information about individual variables, the IPC Cluster method can be used to visualize steps in an iterative design process, such as pure optimization, free exploration, or something in between such as interactive optimization. In all cases, the path through the design space can be displayed by highlighting designs in each iteration, superimposed on the IPC

cluster visualization. An example of this for the design problem introduced in Figure 2.1 is shown in Figure 3.5, which traces the start of an optimization path through the design space (in a zoomed-in view of the IPC cluster plot from Figure 3.4) using a pattern search optimization algorithm (Torczon, 1997). The final iterations, not shown, are all very close to the final optimal solution, and are therefore less interesting to visualize. Since the designs associated with the iterations do not necessarily fall at the specified isoperformance levels, they are placed at the closest level greater than or equal to their performance.

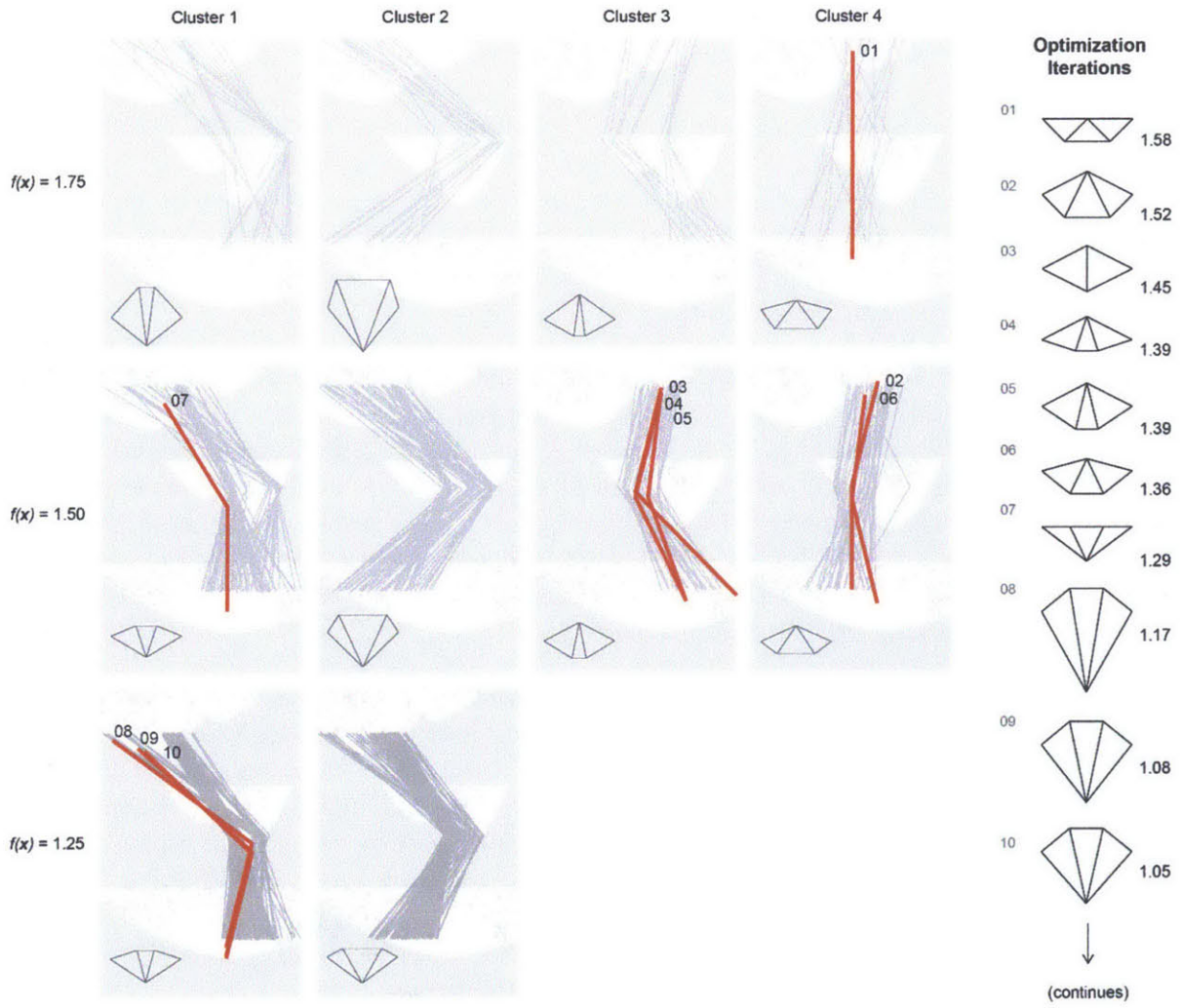


Figure 3.5: *Optimization iterations through the design space of the problem introduced in Figure 2.1, with design points highlighted in red and labeled according to iteration number. The design for each iteration is also shown in a literal glyph plot on the right, with the performance score also labeled.*

While the highlighting works well in a static representation, it could be even more effective in an interactive environment. The "linked views" strategy could be employed for free exploration, such that the user clicks on a design in the IPC cluster, highlighting it, and a literal glyph is also displayed. Conversely, a user could choose or generate a literal glyph within the design space, and the corresponding line segments could be highlighted in the IPC cluster visualization, revealing both performance and the relation to other design options. Interesting designs could be cataloged and stored as highlighted options in the IPC cluster plot as a way to organize and document the design exploration process.

The facilitation of this type of free exploration shows how the IPC cluster method supports the "Design by Shopping" paradigm. This new type of visualization provides a graphical catalog of options that is extensive, yet well organized. Chapter 4 expands on this idea with additional examples, illustrating the use of IPC cluster plots for visualization of optimization, directed exploration, and shopping.

3.7 Design space approximation via surrogate modeling

The previous sections have given the details of the new IPC cluster method, and provided some discussion about ways it could be used as a conceptual design aid. However, it is also important to consider the time and effort required to generate the visualizations.

The IPC clusters visualization method, and DSV methods more broadly, require a large data set of design points (and associated performance) to be produced. Data sets that sample the design space densely can be very time-consuming to generate, due to the computational expense of simulation-based performance evaluation. In this case, the time required to generate the visualization can subvert the point of using an efficient optimization algorithm or focused exploration of the design space, since the best performers can essentially be identified through exhaustive search.

To avoid this issue, it may often be preferable to use an approximate data set, or more precisely, a data set with approximated performance values. This can be obtained by sampling a less accurate but lower cost model instead of the original expression of the objective function. This strategy is called surrogate modeling in the optimization community, and is well documented in the literature (Quiepo et al., 2005; Forrester et al., 2008). Surrogate models are generally statistical response surfaces generated from a set of data points (usually much smaller than the size of the data set needed to for DSV). Compared to simulation-based analysis, sampling the approximate statistical model takes negligible time.

While surrogate modeling has traditionally been used in optimization applications, it is proposed here that the method also be used to generate data for DSV when a thorough sampling of the design space is otherwise cost prohibitive. This approximation process could be used as a pre-processing step prior to generating the IPC cluster plot, as follows:

1. Sample m data points and evaluate their performance from the real design space, using a Latin hypercube or other space-filling sampling plan (Stein, 1987).
2. Build a surrogate regression model using the m data points for training, validation, and test (Hastie et al., 2009).

3. Generate M additional sample points ($M \gg m$), and predict their performance using the surrogate model to obtain an approximate data set, $\widetilde{\boldsymbol{X}}$.
4. Complete steps outlined in Figure 3.3 using $\widetilde{\boldsymbol{X}}$ instead of $\boldsymbol{X}$.

The result of this approach is illustrated in Figure 3.6 and Figure 3.7, which show comparisons of IPC clusters generated with and without approximation, in an overall and close-up view, respectively. In this case, a generalized regression neural network was used as the surrogate model (Specht, 1991), implemented by the MATLAB command `newgrnn` (MathWorks, 2014), with m = 1,000 and M = 200,000 (the data set for the original visualization had 20,000 data points). The comparison shows that the approximation is not perfectly accurate, but is should be close enough to get a reasonable understanding of the design space and its variables, and to provide a backdrop for an optimization or exploration process.

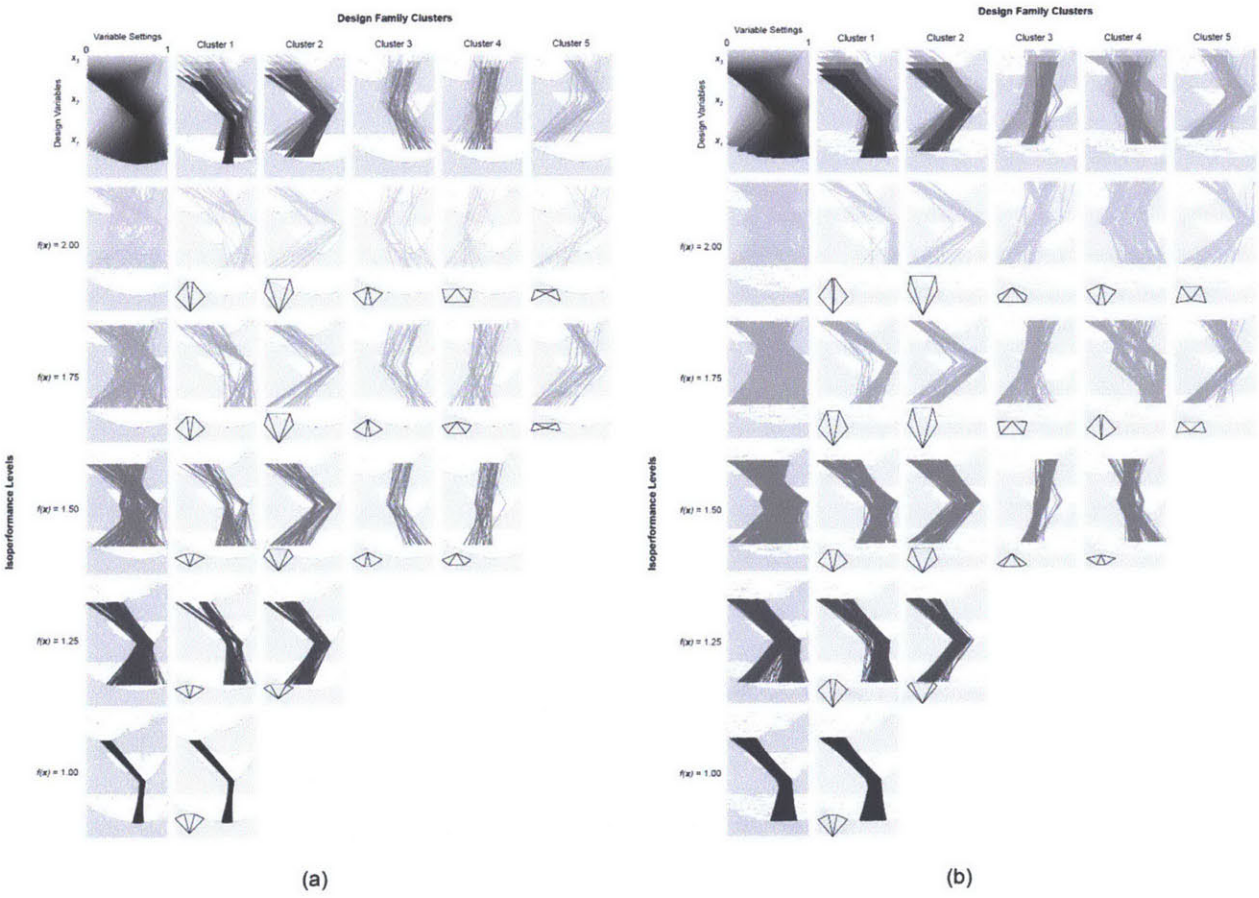


Figure 3.6: *Two overall views of versions of the IPC Cluster plot for the problem introduced in Figure 2.1: (a) the original version, without approximation, and (b) the approximate version generated using surrogate modeling.*

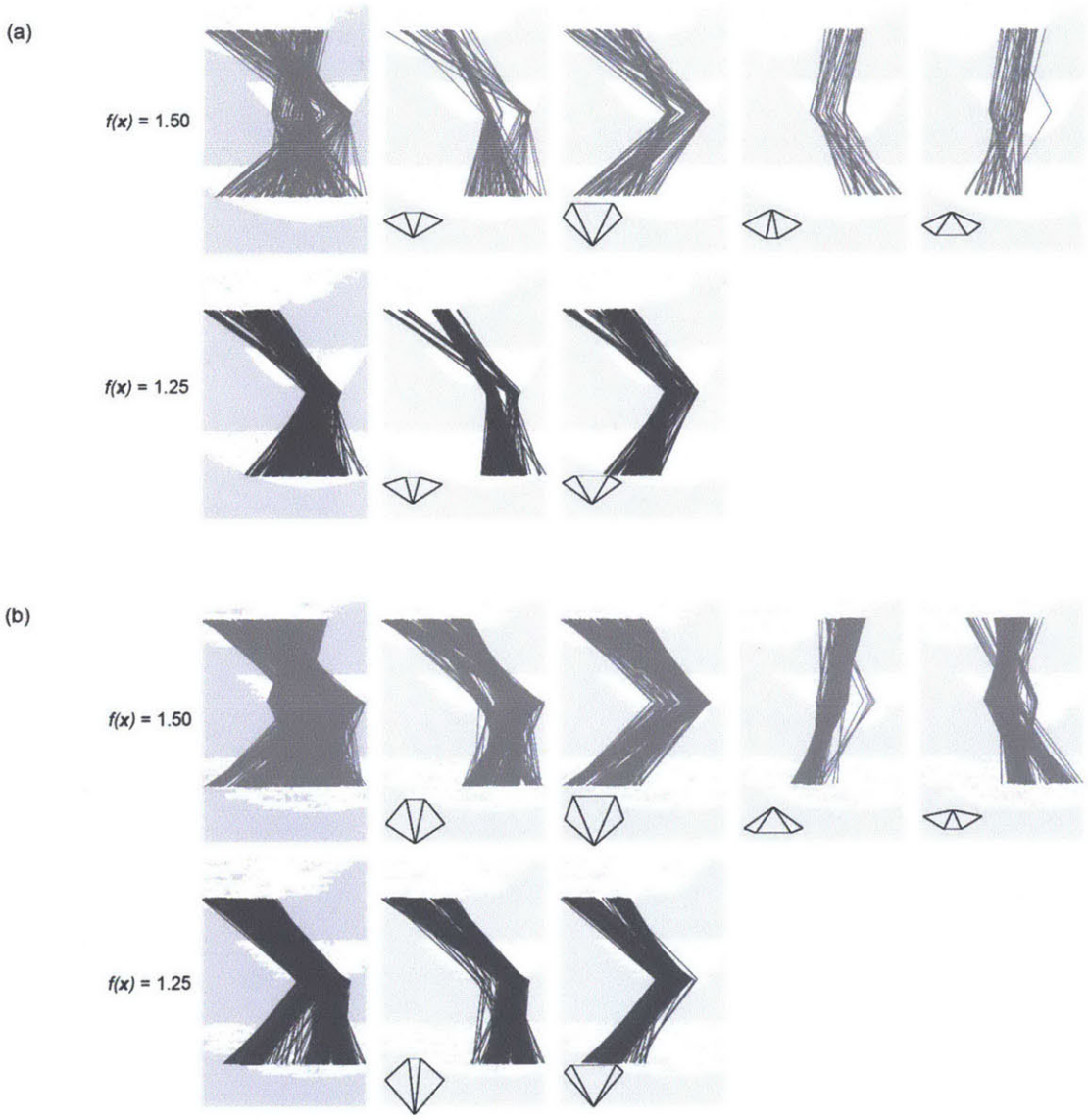


Figure 3.7: *Two close-up views of versions of the IPC Cluster plot for the problem introduced in Figure 2.1: (a) the original version, without approximation, and (b) the approximate version generated using surrogate modeling.*

3.8 Summary of contributions

This chapter has presented a new method for high-dimensional DSV, the IPC cluster method, which expands on the existing parallel coordinates plot using the principles of small multiples, isoperformance, and statistical clustering. The new plotting technique achieve the goals set forth in Section 1.2, in that it provides global design space information, an understanding of individual variables, and a means to visualize optimization and

exploration. The method also supports the design by shopping approach in a broad sense, providing a catalog of options that can quickly be compared in terms of both quantitative performance and unformulated, qualitative goals. Because of these features, the IPC cluster method offers advantages over the existing DSV approaches reviewed in Chapter 2.

The illustrations in this chapter have all focused on a small 3-variable problem, for simplicity and clarity. The next chapter provides several case studies that show how the IPC cluster method could be applied to more complicated problems of higher dimension.

CHAPTER 4

Design Examples

This chapter demonstrates how the new IPC cluster visualization approach, introduced in Chapter 3, can be used on high-dimensional design problems. Two conceptual structural design problems are introduced: a roof truss, and a cantilever truss. Both problems have well-known historical solutions that have been characterized as efficient or optimal. Visualizing the design space of these problems provides insights about these solutions in the context of a broad variety of options.

4.1 Roof truss design (4 variables)

The first design example is a 4-variable planar roof truss, illustrated in Figure 4.1. The truss is symmetrical, simply supported, and has a fixed gabled shape for its top chord. Uniform loading is applied at the joints along the top chord. The design problem is to determine the shape of the lower chord, through the four variables which control the vertical coordinates of joints along the bottom of the truss. The baseline structure has a flat bottom chord, which is necessary when a flat ceiling or floor is required at that level in a building. However, when the roof truss is left exposed, there is much greater flexibility in the bottom chord shape.

Swiss structural engineer Robert Maillart (1872-1940) designed an innovative version of this roof truss in reinforced concrete for a 1924 warehouse in Chiasso, Switzerland, shown in Figure 4.2a. Maillart's version of the truss employs a curved bottom chord with a rise in the center. This shape may initially seem counterintuitive, since the depth is reduced where the bending moment demand is greatest. Its playful

geometry may also come across as whimsical and irrational, compared to the familiar flat-chord version. In fact, the Chiasso design is highly efficient because its shape results in zero forces in the truss's interior diagonal elements under uniform loading. Maillart eliminated these elements in his version, relying on frame action to handle asymmetrical loading.

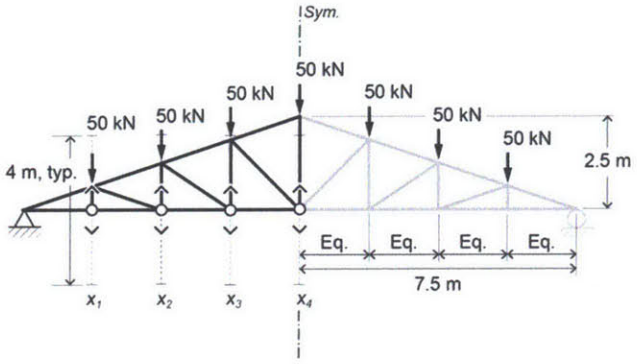


Figure 4.1: *4-variable design problem that considers the shape of the bottom chord of a planar roof truss.*

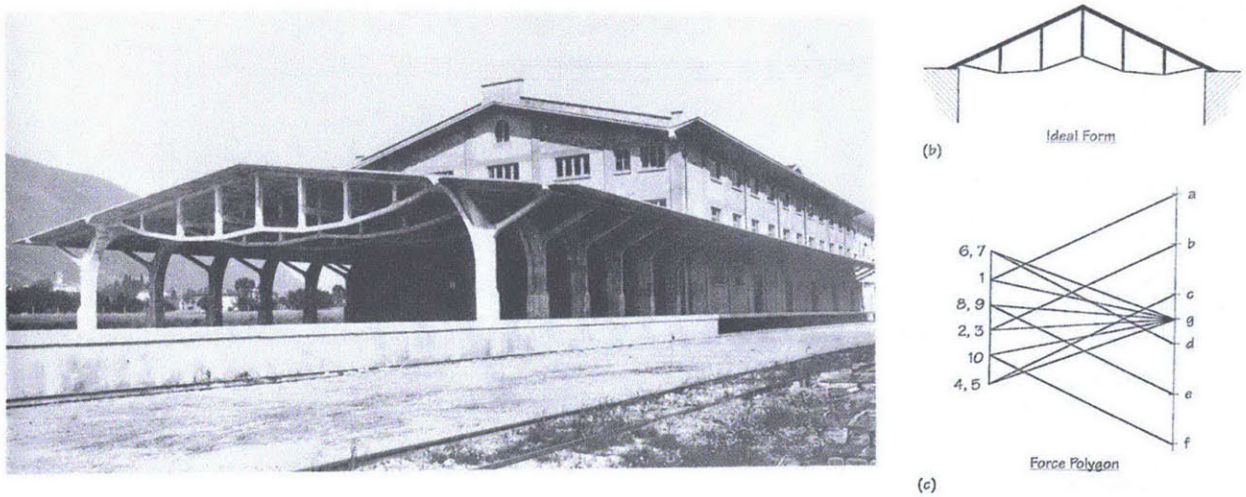


Figure 4.2: *Warehouse roof in Chiasso, Switzerland (1924) by Swiss structural engineer Robert Maillart (1872-1940). Photo in (a) from Billington (1983) and graphic statics in (b) from Allen and Zalewski (2009).*

The efficiency and innovativeness of the Chiasso truss geometry derives from its internal flow of forces. The downward point loads along the top gable are transferred down the vertical elements to the bottom chord, whose curves are the precise funicular shape needed to equilibrate these forces as two hanging cables (a parabola for uniform loading), removing the need for diagonals. The central vertical element carries the reactions of the curved cabled shapes up to the apex of the truss in tension. The gabled top chord equilibrates this vertical reaction with axial force in compression, which remains constant along its length. This flow of forces can be visualized using graphic statics, a graphical technique for structural equilibrium analysis (Allen & Zalewski, 2009), as shown in Figure 4.2b.

The Chiasso truss is therefore high performing in a structural sense, due to the removal of the diagonal elements, and in a constructability sense, due to the constant force in the top gabled chord, allowing the same element size to be used to full capacity along the entire length. However, the quantitative level of this performance, compared to the flat-chord version and other options, is important to understand in conceptual design. This section explores this design problem using the IPC cluster visualization method, with the goals of highlighting Chiasso-like trusses among a wider range of options and investigating new possible solutions.

Figure 4.3 shows an IPC cluster plot for this 4-variable design problem. The plot uses $p_{max} = 2$, $n_p = 4$, $\varepsilon = 0.03$, $n_c = 4$, and $d_c = 0.30$. Like the previous examples in this thesis, the structural performance metric is required material volume, calculated by sizing each member based on its computed internal forces, considering both axial stress and buckling in compression. The members are again assumed to be steel tubes with a wall thickness set to 5% of the diameter. This performance metric does not consider the constructability goal of constant force in the top chord for consistent member sizing, which is sometimes desirable but not required for structural efficiency.

Sixteen designs that behave like the Chiasso truss, characterized by a constant force in the gable chord and zero-force diagonals, are highlighted. These designs have been identified by checking for parabolic curves along the bottom chord. The IPC visualization of these designs and the broader design space illuminates several important points. First, not all versions of the Chiasso roof truss are high performing. Some versions, like designs A through D, require twice the amount of structural material as the most efficient options. Second, the Chiasso-like designs fall into two distinct family clusters; both clusters have similar relationships between the variables (generally an increase in setting from x_1 up to x_4), but Cluster 2 is shifted up in variable settings compared to Cluster 1. This results in trusses that are shallower, which may be desirable to the designer if space is at a premium. Third, a version of the Chiasso roof appears among the optimal set of designs in Cluster 1, but Chiasso-like behavior is not strictly required for high performance, suggesting that other interesting and undiscovered options exist.

Figure 4.4 illustrates an exploration of the design space to investigate such options. An interactive evolutionary algorithm is used to navigate the design space; the algorithm evolves populations of designs with the goal of improved structural efficiency, similar to many optimization approaches. However, it also allows the designer to contribute to the process, by selecting the designs among a generation's top performers to become parents of the next generation (Mueller & Ochsendorf, 2013). This interaction gives the designer a way to express important but unquantifiable design goals. The figure shows the selected designs in each generation, starting with the flat-chord version of the truss from Figure 4.1, and culminating in a final selected design.

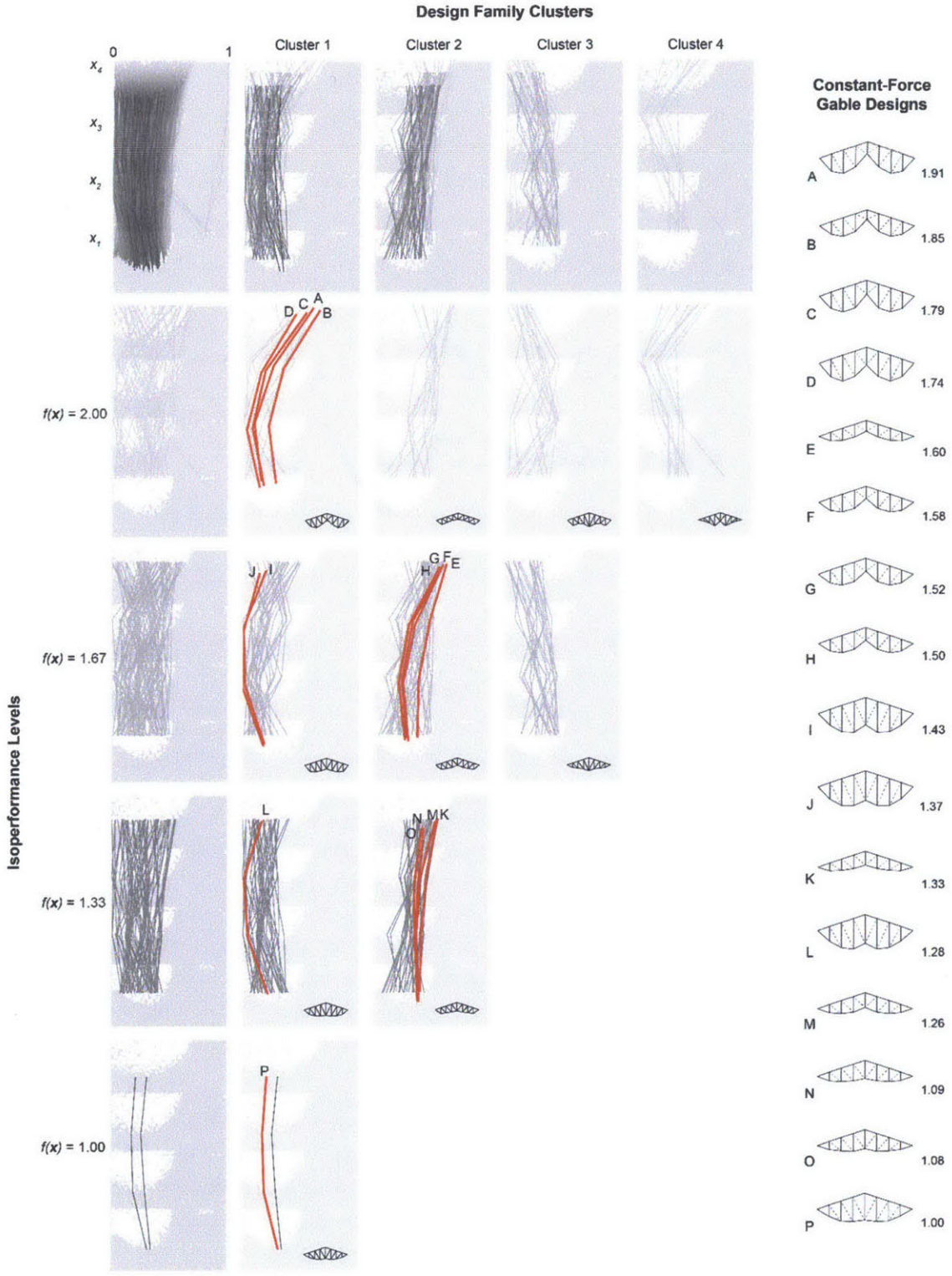


Figure 4.3: *IPC visualization of the roof truss design problem given in Figure 4.1, with Chiasso-like designs highlighted.*

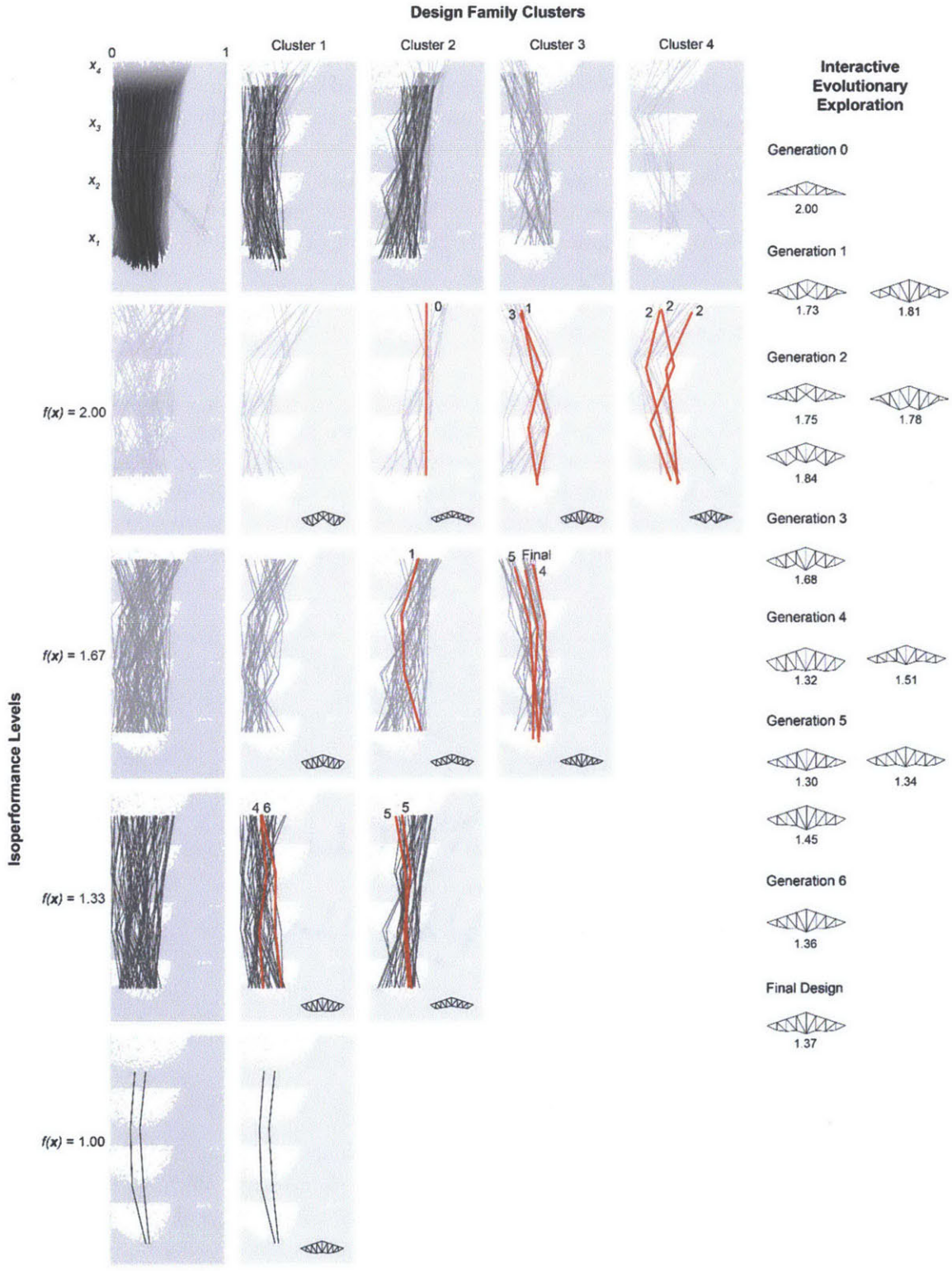


Figure 4.4: *IPC visualization of the roof truss design problem given in Figure 4.1, highlighting evolutionary exploration.*

Because the goal of design space exploration is not pure optimization, the most efficient options in the design space are not necessarily considered. Instead, the algorithm tries to provide higher performing alternatives that nevertheless relate to the designer-selected options. The IPC cluster visualization shows that early in the exploration process, the designer is interested in irregular geometries that correspond to zig-zag-like representations in parallel coordinates. The algorithm produces increasingly regularized versions of these options, resulting in improved performance, while still maintaining a similar character. By Generation 4, the designer has converged toward designs that have a slightly lowered central joint, expressed as a low x_4 value in the plot. In the final design, the designer has selected a relatively shallow version of this expression, resulting in an elegant and unfamiliar shape with performance that is 32% better than the initial flat-chord option.

4.2 Cantilever truss design (8 variables)

The second design example considers the geometry of a cantilevered truss supporting a point load at the tip. The problem has eight variables projected to the [0, 1] range, as shown in Figure 4.5, and the performance metric is again the required structural weight. The known optimal solution to this problem is a Michell structure, named for the Australian structural engineer A. G. M. Michell who discovered this structural form (1904). The general cantilevered Michell truss is shown in Figure 4.6a. The geometry of the truss follows orthogonal stress trajectories in an equivalent solid beam, resulting in a system of approximated curves.

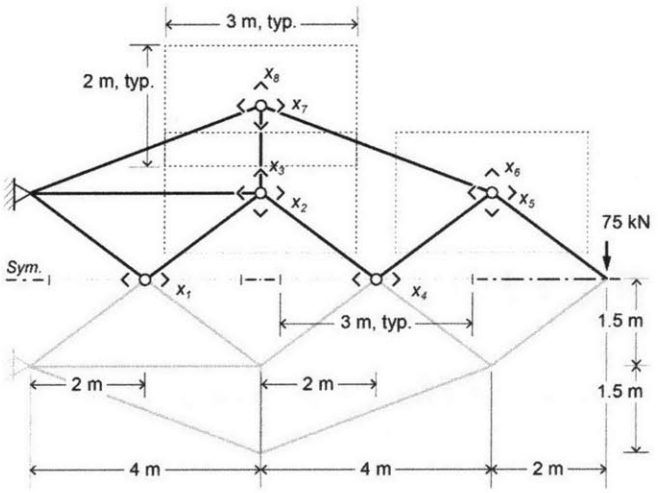
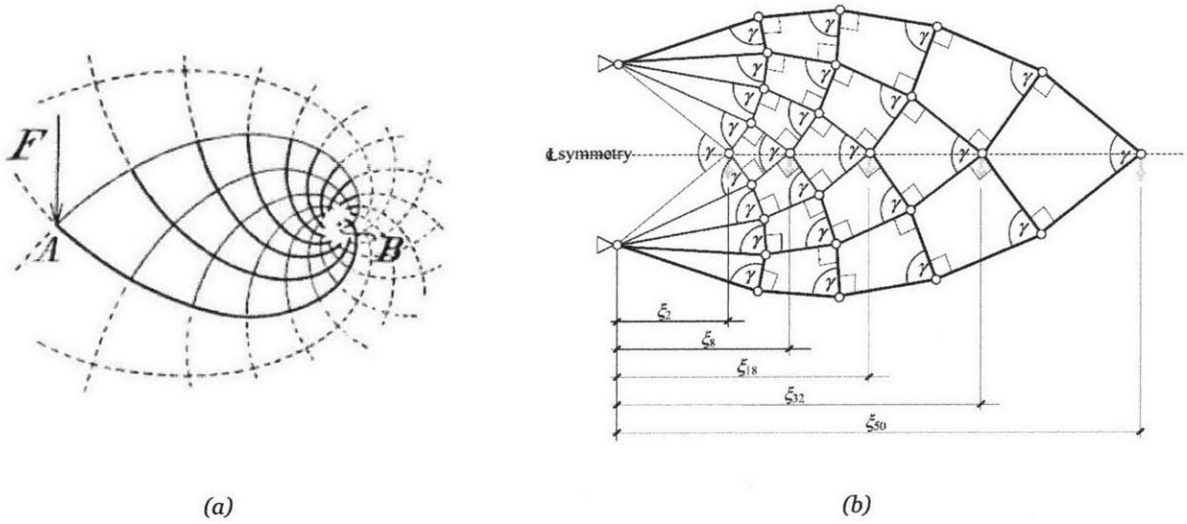


Figure 4.5: *8-variable design problem that explores the geometry of a cantilevered truss.*

Recent research in Michell trusses has shown that the cantilevered version can be parameterized and generated by a single angle, γ, which governs the geometry, and by the number of members, which corresponds to the level of discretization of the infinite stress field, as shown in Figure 4.6b (Mazurek et al., 2011). For a given total length and number of members, there is a single Michell solution. The Michell solution for the problem given in Figure 4.5 is shown in Figure 4.7.



(a) (b)

Figure 4.6: *Optimal cantilever structures, originally proposed by Michell (1904) and discussed theoretically, as shown in (a), and more recently studied geometrically (Mazurek et al., 2011), as shown in (b).*

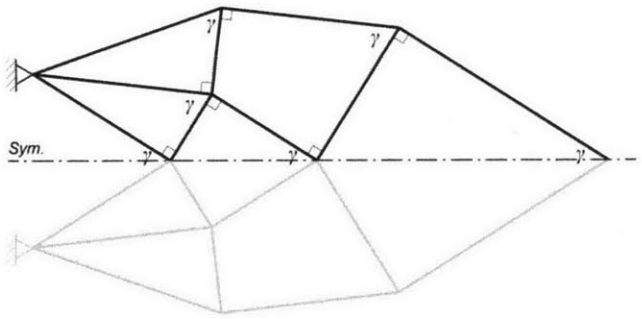


Figure 4.7: *Optimal Michell-like geometry for the problem introduced in Figure 4.5. There is exactly one Michell solution for the 18-bar configuration and 10m span, generated by $\gamma = 1.123$ radians.*

As with the Chiasso roof truss, while the optimality of the Michell truss is known, the question of how suboptimal variations of the geometry degrade performance remains open. This section therefore presents visualizations of the Michell truss design space, using the IPC cluster technique, to investigate the relationships between the Michell truss and alternative design options. Figure 4.8 shows an IPC cluster plot for this 8-variable design problem, with designs highlighted from an interactive evolutionary design session. The plot uses $p_{max} = 1.2$, $n_p = 4$, $\varepsilon = 0.01$, $n_c = 6$, and $d_c = 0.55$. Figure 4.9 shows the same IPC cluster plot, but with optimization iterations highlighted instead. In both plots, the optimal Michell design is also highlighted, and is shown to be alone at the best isoperformance level. Both plots also show a range of suboptimal designs that perform up to 20% worse than the Michell version, and that exhibit a modest amount of diversity in terms of aesthetic characteristics.

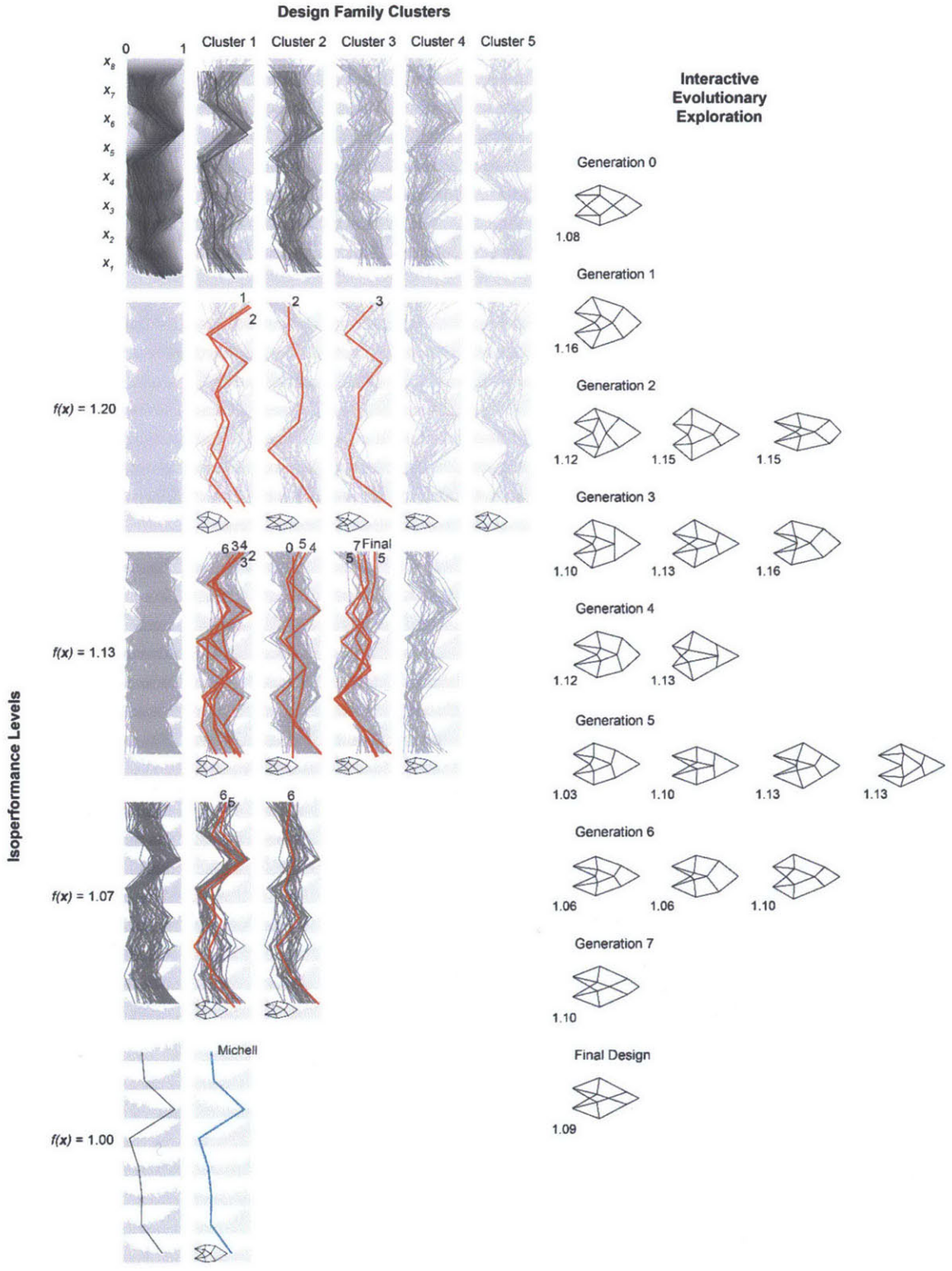


Figure 4.8: *IPC visualization of the cantilever problem given in Figure 4.5, with evolutionary exploration highlighted.*

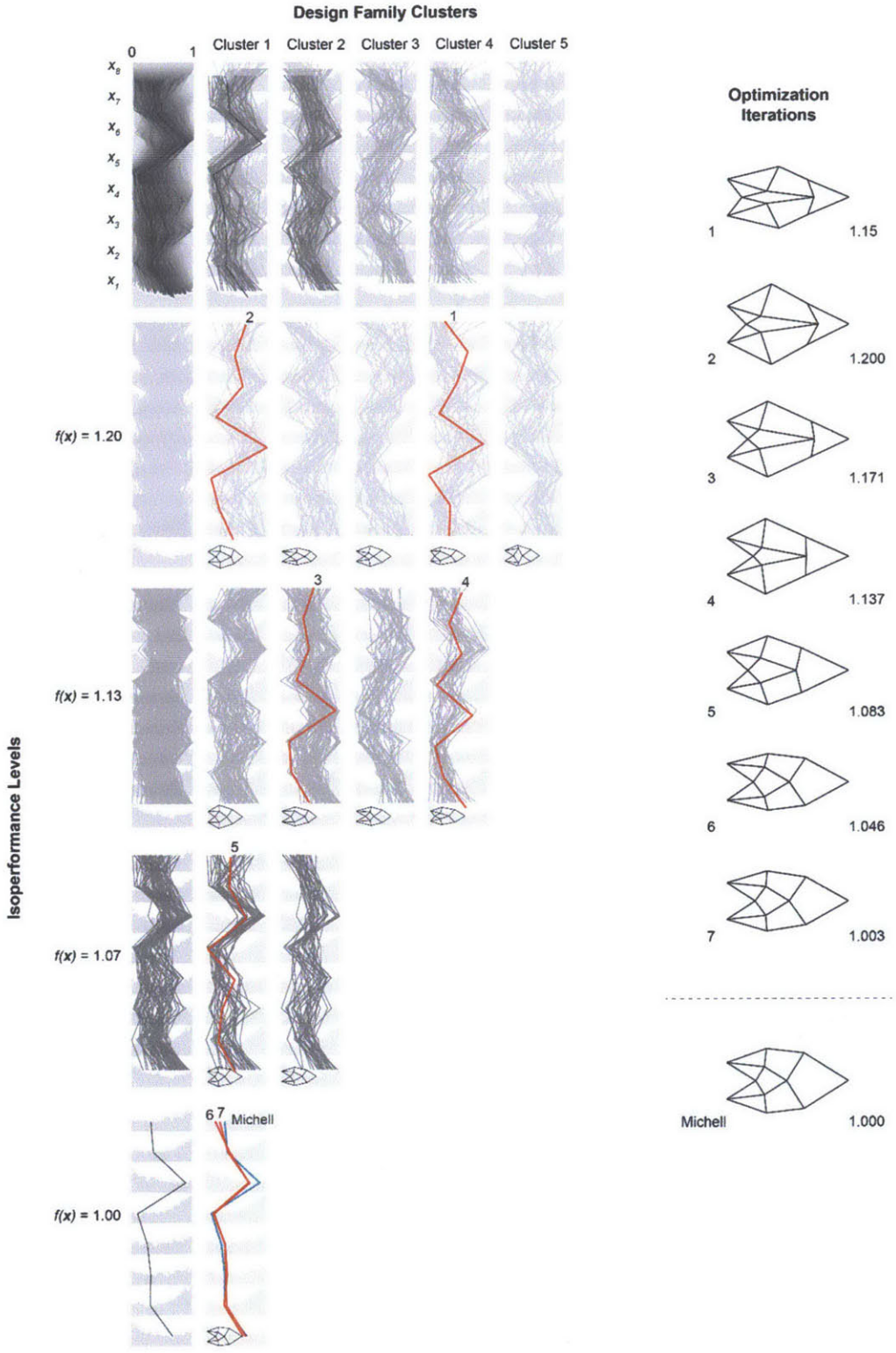


Figure 4.9: *IPC visualization of the cantilever problem given in Figure 4.5, with optimization iterations highlighted.*

In the interactive evolutionary exploration shown in Figure 4.8, the user tries to select designs that are visually appealing and that improve constructability, defined in this case by maintaining colinearity between members across connections when possible. Evolutionary pressure based on reducing material volume pushes the designer toward Michell-like designs, which follow the spiraling stress trajectories and therefore have no colinearity. The user navigates this tradeoff over seven generations, eventually settling on a design that performs 9% worse than the Michell optimum, with a visually compelling geometry and several pairs collinear members that could constructed with single elements. The evolutionary process did not lead to significant performance improvement compared to the starting design, but it did allow the user to understand and explore the design space with qualitative goals in mind.

The optimization iterations highlighted in Figure 4.9 show that the algorithm (in this case an evolutionary optimization algorithm without interaction) was able to converge relatively close to the optimal Michell solution in seven iterations, beginning with a design that performs 15% worse. The difference between the suboptimal design found in Iteration 7 and the Michell design is small in terms of performance (0.3%), but noticeable graphically in both the parallel coordinates representation and the literal glyph plots.

4.3 Discussion

This chapter has demonstrated the IPC cluster method on two conceptual structural design problems, the simply supported roof truss, and the cantilevered truss. Both examples have historical solutions, which the IPC cluster plots can highlight and compare to a broader range of options. In the case of the roof truss, a range of solutions that behave like Maillart's Chiasso design can be found, and while a few versions are among the best performers in the broader design space, many do not use less material that other design options (although they do have nonstructural benefits in terms of constructability). In contrast, for the cantilever problem, Michell's unique optimal solution is shown to be the undisputed best performer. Still, a range of interesting and visually distinct solutions that use within 10% of the optimal material volume can be found in the design space through interactive evolutionary exploration.

In both cases, the IPC cluster plot provided a meaningful backdrop to organize designs of interest, and helped add important insights beyond what is legible from the literal glyphs alone (or numerical data) alone. However, global patterns and variable relationships were a bit easier to interpret in the 4-variable roof truss problem. At eight variables, the cantilever problem led to visualizations that were more challenging to interpret, suggesting that there is an upper limit to the number of variables that can be reasonably displayed with the IPC cluster method. The next and final chapter considers this issue, among several others, in a discussion of future research directions.

CHAPTER 5

Conclusion

This chapter summarizes the contributions of the thesis, and discusses potential applications and directions for future work. This thesis has established the unmet need in conceptual design for a high-dimensional design space visualization technique that communicates global patterns, individual variable behavior, and the paths of design iterations found by optimization, exploration, or other processes. In response to this need, this thesis has proposed a new type of DSV plot, called the IPC cluster plot, that extends existing work in parallel coordinates plotting using the principals of small multiples, isoperformance, and statistical clustering. The advantages of the IPC cluster plot over existing methods include full coverage of the design space, systematic legibility, portrayal of variable relationships, and a strong representation of design performance. The IPC cluster technique has been demonstrated for three conceptual design problems, a simple 3-variable problem in Chapter 3, and two more complicated problems in Chapter 4. These examples show that the new approach can provide unprecedented insights into high-dimensional design problems, and can organize a range of design alternatives in a logical, legible manner.

5.1 Applications and potential impact

The aim of this research is to help designers and engineers make better decisions in conceptual design. By understanding the design space of alternatives in a global, systematic way that is also graphical and fast, designers can interpret complicated problems and perceive the performance implications of their design

choices. Design space visualization makes informed, evidence-based design faster and easier, compared to methods that are purely numerical, and can potentially lead to earlier integration of performance goals.

The main contribution of this thesis is the new IPC cluster visualization technique, which makes important steps toward realizing this potential. Unlike existing visualization methods, which are either non-systematic, narrow in design space coverage, or illegible in terms of variable relationships or performance, the IPC cluster method is general, broad, and highly focused on the link between variable behavior and performance. This plotting method can help designers identify interesting regions of the design space, simplify or eliminate variables, and visualize design iterations or alternatives.

In its current version, the IPC cluster method can be applied to any design problem with real-valued variables and a quantitative performance metric. While the examples in this thesis have focused on structural design problems, the approach is independent of domain, and could be used in a variety of engineering, design, and decision-based disciplines. There are several key directions for future research that could make the IPC cluster method even more widely applicable, discussed in the following section.

5.2 Directions for future work

Important areas for future research in DSV, and in the IPC cluster method specifically, include expanding variables types, incorporating constraints, improving clustering stability, identifying isoperformers, extending to multiobjective problems, and implementing interactivity. This section briefly discusses each of these topics and suggests future research steps.

First, all of the examples shown in this thesis have continuous real-valued variables projected to the [0, 1] range. However, many design problems include other variable types, including binary, integer, and discrete variables. All of these variable types could be incorporated into the IPC cluster method, since they are supported by the underlying parallel coordinates plotting technique. Still, they would present some challenges, since the projected design space silhouette used as an underlay would not be possible. It is unclear how these variable types would impact legibility of the visualization; this should be studied further.

Second, the examples in this thesis have all been unconstrained problems, meaning that all designs within variable bounds are feasible. However, many design problems include formulated constraints, or equations that bound the feasible design space along more complicated boundaries. The IPC cluster method could incorporate constraints implicitly by only showing designs that are feasible, which would eliminate some possible combinations of variable settings within the variable bounds. It would potentially be even more beneficial to develop an explicit way to display design space constraints, however, as it would help designers understand the feasible region(s) of the design space more actively.

A third direction for future work is the statistical clustering method used to generate the plots. As discussed in Chapter 3, this thesis used the k-means clustering algorithm, and determined the appropriate value for k in an iterative, semi-empirical manner. However, the stochastic nature of the algorithm can lead to inconsistent results; on one run, the algorithm produces logical clusters, but a second run on the same data produces clusters that are less appealing. To control for this, the implementations used in this thesis used specific seeds for MATLAB's random number generator so that the output would be the same on each run. However, the

greater problem of stabilizing the quality of the clusters needs to be addressed. Future work in this area should also focus on the vertical stacking of clusters, so that they form logical families consistently. In this work, trial and error was often required to obtain reasonable sorting and stacking, again due to the stochastic nature of the clustering algorithm.

A fourth area for future research is to incorporate more sophisticated methods for identifying isoperforming designs, such as those suggested in the literature (de Weck & Miller, 2002). This thesis used a simple enumeration (or approximation) and search approach, but this method can be slow for computationally expensive or high-dimensional problems. Identifying isoperformers in a more efficient manner would allow the visualizations to be completed more quickly without degrading their quality.

Fifth, while this thesis focused on single-objective problems, it would be beneficial to eventually extend the IPC cluster method to multi-objective problems, which are common in engineering and design. This could involve creating multiple linked IPC cluster plots, one for each performance metric, in which the same designs are presented across different design spaces, but would be challenging in terms of the space required for the visualization. Further research is needed to determine the best way to graphically convey design spaces and the objective space together.

Finally, a key next step is to implement the IPC cluster method, which is shown in this thesis through static graphics, as part of an interactive computational design tool, as described briefly in Section 3.6. Giving designers the ability to dynamically explore the design space by clicking on designs in the IPC cluster plot would allow the visualization to act more effectively like a catalog for shopping. Conversely, designers could work in the domain of the literal design representation, and observe the impact of changes through updated highlighted designs on the IPC cluster plot. Designers could also experiment with brushing techniques such as those suggested by Stump et al. (2003), which would allow them to adjust variable bounds and view the updated design space of options in real time.

Beyond these specific steps for future work in the IPC cluster method, an important broad challenge is the practical upper limit on the number of variable that can be visualized. This work concentrates on problems with three or more variables, and can theoretically handle any number of variables. However, in practice, the visualizations become unwieldy for design problems slightly larger than those shown here, around ten variables. While many design problems can be expressed in ten variables or fewer, some problems can have hundreds of variables, and are practically impossible to visualize using any method. Research that addresses this issue, through numerical variable elimination or other means, is needed.

5.3 Concluding remarks

As designers and engineers grow more dependent on black-box simulations to understand the performance of conceptual design alternatives, the need for graphical organization of these options and their performance grows stronger. Design space visualization, which has been long been part of engineering tradition, offers a unique opportunity to meet this need. This thesis has focused on high-dimensional design problems, which are difficult to visualize in the familiar context of Euclidian space, with the goal of illustrating key patterns and critical relationships in seemingly messy and disordered design spaces. Bringing clarity to black-box design

problems with discrete design data points is challenging because there are no longer smooth analytical expressions to manipulate, but it is critical for this same reason. Simulated performance feedback has the potential to contribute powerful and detailed information beyond what is possible with simple analytical models. Organizing and processing this information early in the design process, when the largest and most impactful decisions are made, can lead to significant improvements in the quality of the end product. The research presented in this thesis is an important step in making this more feasible.

APPENDIX

MATLAB Code for IPC Clusters

This appendix provides the MATLAB source code used to generate the IPC cluster design space visualizations shown in this thesis. The code contains 3 functions, described briefly below and then given in full subsequently.

- `IPCCluster.m`: The main algorithm that divides a given data set into isoperformance levels and clusters, and creates the overall plot.
- `ParCloud.m`: A subroutine that draws a single small multiple plot (of the type introduced in Figure 3.1) in the IPC cluster visualization.
- `ParCoord.m`: A subroutine that draws a single design point as a series of line segments per the parallel coordinates plotting technique, colored to indicate performance.

Additional code may be required to generate IPC cluster plots for specific problems, including a function that generates a literal glyph of a design point, and functions that find or generate designs of interest to be highlighted in the plot.

IPCCluster.m

```
function [ ] = IPCCluster( X, y, pmax, np, nc, epsilon, dc, factor, bg, points, lines, des,
desplot, highlights, hldata )

    % X = dataset (row = point, column = variable setting)
    % y = corresponding performance values (row = point)
    % pmax = maximum isoperformance level
    % np = number of isoperformance levels
    % nc = maximum number of clusters to generate
    % epsilon = tolerance for identifying isoperforming designs
    % dc = minimum distance between clusters
    % factor = with factor for plotting
    % bg = bool: plot background?
    % points = bool: plot projected design space silhouette?
    % lines = bool: plot parallel coordinate lines?
    % des = bool: plot literal glyphs?
    % desplot = function that plots literal glyph
    % highlights = bool: plot highlighted designs?
    % hldata = data for highlighted design plotting


    % Count variables
    vars = size(X,2);

    % Set up isoperformance levels
    delta = (pmax - minval)/(np-1);
    binmin = 1;
    leveldata = cell(np, 1);
    leveldatahl = cell(np,1);
    scorelevels = [];

    % Set up empty bins for cluster data
    clustdata = cell(nc,1);
    hlclustdata = cell(nc,1);

    % Set up plots colors
    intcolor = 0.9 * [1 1 1];
    extcolor = 0.85 * [1 1 1];

    % Prepare figure and plotting
    figure;
    axes('Position',[.005 .005 .99 .99], 'xtick', [], 'ytick', [],'box', 'on', ...
        'handlevisibility', 'off','color','none')
    set(gcf,'color','w');

    % Plot full design space composite in upper left corner
    subtightplot(np+1, nc+1, 1, [], [], [], 'color', extcolor);
    if ~des
        ParCloud([X, y], [X, y], pmax, vars, bg, points, lines, false, hldata );
    end
    set(gca,'xtick',0:1:1);
    set(gca,'xaxislocation','top');
    set(gca,'ytick',1:1:vars);
    set(gca,'xcolor',extcolor);
    set(gca,'ycolor',extcolor);
```

```
set(gca,'YTickLabel',sprintf('x%d|',[1:1:vars]));
if ~bg
    axis off;
end

% Step 1: Get isoperforming designs
for i = 1:np

    % Compute and save isoperformance level
    binmax = binmin + delta;
    scorelevels = [scorelevels; binmin];

    % Identify isoperforming designs
    rows = y > binmin - epsilon & y < binmin + epsilon;
    bindata = [ X(rows, :), y(rows, :)];

    % Identify designs among "highlight" data to go in ith isoperformance level
    hlbindata = [];
    if highlights
        hlrows = hldata(:,vars+1) <= binmin + epsilon*delta & hldata(:,vars+1)...
            > binmin - delta + epsilon*delta;
        hlbindata = hldata(hlrows,:);
    end

    % Save isoperformance data
    leveldata{np + 1 - i} = bindata;
    leveldatahl{np + 1 - i} = hlbindata;

    % Incremement isoperformance level
    binmin = binmax;
end

% Prepare for clustering
prevclust = 0;

% Step 2: Generate clusters for each isoperformance level
for i = 1:np

    % Get ith isoperformance data set
    bindata = leveldata{np + 1 - i};
    hlbindata = leveldatahl{np + 1 - i};
    cdata = [bindata(:,1:vars)];
    numclust = nc;
    smallDist = true;

    while smallDist == true;

        % If the data set is too small, reduce the numer of clusters
        if size(cdata, 1) < numclust
            numclust = numclust - 1;

        % If the number of clusters is the same as the previous row, stop (we
        % need at least as many clusters in each ascending row)
        elseif numclust == prevclust

            % If the number of clusters is the maximum, this is our first loop,
            % so we assign clusters
            if numclust == nc
```

```
            [clusters, centroids] = kmeans(cdata, numclust, ...
              'emptyaction','singleton');
            numclust = size(centroids, 1);
            break;

        % Otherwise, we already have our clusters from the previous loop, and
        % we have 1 more cluster than numclust is set to
        else
            numclust = numclust + 1;
            break;
        end

    % If the data set only contains one point, there is only one cluster that
    % we assign manually
    elseif size(cdata,1) == 1
        clusters = [1];
        centroids = cdata(1,:);
        numclust = 1;
        break;

    % Otherwise, we need to check the whether the distance between clusters
    % meets our mindist requirement
    else
        [clusters, centroids] = kmeans(cdata, numclust, 'emptyaction','singleton');
        numclust = size(centroids, 1);
        dist = CheckMinDist(centroids);

        % If we haven't met the distance requirement and we have at least 2
        % clusters, we reduce the number of clusters and iterate again
        if (dist < dc && numclust > 1)
            numclust = numclust - 1;

        % Otherwise, we stop
        else
            smallDist = false;
            break;
        end
    end
end

% Save the number of clusters from the ith level
prevclust = numclust;


% Step 3: Sort centroids to align with previous row, if it exists
if i > 1
    newc = centroids;
    [sortedc, sortedcentroids] = SortCentroids(oldc, newc);
else
    sortedc = [1:1:size(centroids, 1)];
    sortedcentroids = centroids;
end

% Determine location for plot
index = (nc + 1)*(np + 1 - i) + 1;

% Plot Lefthand isoperformance composite plots
subtightplot(np+1, nc+1, index, [], [], [], 'color', extcolor)
```

```
if ~des
    ParCloud([X, y], bindata, pmax, vars, bg, points, lines, false, ...
    hlbindata );
end

% Set plotting properties
ylabel(sprintf('%0.2f',scorelevels(i)),'rot',0);
set(gca,'xtick',[]);
set(gca,'ytick',[]);
set(gca,'xcolor',extcolor);
set(gca,'ycolor',extcolor);
set(get(gca,'YLabel'),'Color','k')

if ~bg
    axis off;
end

% Identify the appropriate cluster for each design in the highlight data set
% for the ith isoperformance level
hlclusters = [];
for j = 1:size(hlbindata,1)
    point = hlbindata(j,1:vars);
    ind = 1;
    mindist = 1e8;

    for k = 1:size(sortedcentroids,1)
        cen = sortedcentroids(k,1:vars);
        dist = norm(point - cen, 2);
        if dist < mindist
            mindist = dist;
            ind = sortedc(k);
        end
    end

    hlclusters = [hlclusters; ind];
end


% Plot each interior view for the ith isoperformance level
for j = 1:numclust

    % Determine location for plot
    index = (nc+1) * (np + 2 - i) - (nc - j);

    % Get data for ith performance level, jth cluster
    thisdata = bindata(clusters == (sortedc(j)),:);
    thishldata = hlbindata(hlclusters == (sortedc(j)),:);
    clustdata{j} = [clustdata{j}; thisdata];
    hlclustdata{j} = [hlclustdata{j}; thishldata];

    % Plot interior IPC clusters
    subtightplot(np+1, nc+1, index, [], [], [], 'color', intcolor);

    % Plot either literal glyph or cluster as parallel coordinates
    if des
        desplot(sortedcentroids(j,1:vars),'k',[0.1,0.1],.5);
    else
        ParCloud([X, y], thisdata, pmax, vars, bg, points, lines, highlights, ...
```

```
                    thishldata);
            end

            % Set plotting properties
            set(gca,'xtick',[]);
            set(gca,'ytick',[]);
            set(gca,'xcolor',intcolor);
            set(gca,'ycolor',intcolor);

            if ~bg
                axis off;
            end
        end

        % Save cluster centroids for sorting at the next isoperformance level
        oldc = sortedcentroids;
    end

    % Plot top composite clusters
    for i = 1:numclust

        % Determine location for plot
        index = 1 + i;
        subtightplot(np+1, nc+1, index, [], [], [], 'color', extcolor);

        % Get data for ith cluster (all isoperformance levels)
        thisdata = clustdata{i};
        thishldata = hlclustdata{i};

        % Plot parallel coordinate views
        if ~des
            ParCloud([X, y], thisdata, pmax, vars, bg, points, lines, false, thishldata );
        end

        % Set plotting properties
        set(gca,'xtick',[]);
        set(gca,'xaxislocation','top');
        xlabel(strcat({'Cluster '}, {num2str(i)}));
        set(gca,'ytick',[]);
        set(gca,'xcolor',extcolor);
        set(gca,'ycolor',extcolor);
        set(get(gca,'XLabel'),'Color','k')

        if ~bg
            axis off;
        end
    end

    % Final plotting properties
    pix = 1000;
    set(gcf,'Position',[100, 0, pix * factor * ((nc+1)/((vars/2)*(np+1))), pix]);
    set(gcf,'color','none');

end

% Internal function for computing the smallest Euclidian distance between a group of centriods
function [dist] = CheckMinDist(centroids)
```

```
    [m,n] = size(centroids);
    dist = 1e8;

    for i = 1:m
        for j = i+1:m
            dist = min(dist, norm(centroids(i,:)-centroids(j,:)));
        end
    end

end

% Internal function for sorting a new group of centroids based on the previous group
function [sortedc, sortedcentroids] = SortCentroids(oldc, newc)

    sortedc = [];
    newc_orig = newc;

    for i = 1:size(oldc, 1)
        co = oldc(i,:);
        mindist = 100;
        closestc = [];
        for j = 1:size(newc, 1)
            cn = newc(j,:);
            if norm(cn - co) < mindist
                mindist = norm(cn - co);
                closestc = cn;
            end
        end
        ind_orig = find(newc_orig(:,1) == closestc(1,1));
        sortedc = [sortedc; ind_orig];
        ind = find(newc(:,1) == closestc(1,1));
        m = size(newc,1);
        if ind > 1 && ind < m
            newc = [newc(1:ind-1,:); newc(ind+1:m,:)];
        elseif ind == 1
            newc = newc(2:m,:);
        elseif ind == size(newc, 1)
            newc = newc(1:m-1,:);
        end
    end

    for i = 1:size(newc)
        ind = find(newc_orig(:,1) == newc(i,1));
        sortedc = [sortedc; ind];
    end

    sortedcentroids = zeros(size(newc_orig));
    for i = 1:size(sortedc)
        ind = sortedc(i);
        sortedcentroids(i,:) = newc_orig(ind,:);
    end

end
```

ParCloud.m

```
function [ ] = ParCloud( datafull, dataiso, pmax, vars, bg, plotpoints, lines, highlights,...
 hldata )

    % datafull = data set, incl. performance (row = point, col = variable, end col = perf)
    % dataiso = isoperforming data set for this plot, including performance (row = point, col
    %  = variable, final col = performance)
    % pmax = maximum isoperformance level
    % vars = number of design variables
    % bg = bool: plot background?
    % plotpoints = bool: plot projected design space silhouette?
    % lines = bool: plot parallel coordinate lines?
    % highlights = bool: plot highlighted designs?
    % hldata = data for highlighted design plotting

    % Normalize data by maximum isoperformance level, and remove data beyond that
    datafull(:,vars+1) = datafull(:,vars+1) / pmax;
    datafull(datafull(:,vars+1) > 1, vars+1) = NaN;

    % Sort data in descending order
    datafull = sortrows(datafull, vars+1);
    datafull = flipud(datafull);

    % Identify valid subset of full data set (i.e. data within performance range)
    datasub = datafull(~isnan(datafull(:,vars+1)),:);

    % Plot projected design space silhouette as backdop for each variable level
    hold all;
    color = [1 1 1];
    cloudsep = 1.5;

    for i = 1:vars
        normscores = ((datasub(:,vars+1) - (1/pmax)) / (1 - 1/pmax)) / cloudsep + (1 ...
              - 1/cloudsep);
        points = [normscores + (i-1), datasub(:,i)];
        if plotpoints
            scatter(points(:,2), points(:,1), 2, color, 'fill');
            hold all;
        end
    end

    % Create parallelc coorindate plots of design space and highlighted data
    ParCoord(dataiso, pmax, vars, false, true, cloudsep, lines, false);
    if highlights
        ParCoord(hldata, pmax, vars, false, true, cloudsep, lines, true);
    end

    % Set plotting properties
    xlim([0 1]);
    ylim([0 vars])
    pbaspect([1 vars/2 1]);

end
```

ParCoord.m

```
function [ ] = ParCoord( data, pmax, vars, plotscore, points, cloudsep, lines, highlights )

    % data = data set, including performance (row = point, col = variable, final col =
    %    performance)
    % pmax = maximum isoperformance level
    % vars = number of design variables
    % plotscore = bool: is score included in data?
    % points = bool: plot projected design space silhouette?
    % clousep = separation between variable coordinate levels
    % lines = bool: plot parallel coordinate lines?
    % highlights = bool: plot highlighted designs?

    % Normalize data by maximum isoperformance level, and remove data beyond that
    data(:,vars+1) = data(:,vars+1) / pmax;
    data(data(:,vars+1) > 1, vars+1) = NaN;

    % Sort data in descending order
    data = sortrows(data, vars+1);
    data = flipud(data);

    % Set plot styles for highlighted data
    lwcallout = 1.0;
    linewidth = 0.5;
    ccallout = 'r';
    linetype = '-';
    clinetype = '-';

    % Plot each data point in parallel coordinates
    num = size(data, 1);
    for i = 1:num

        % Identify individual data point and performance score
        des = data(i,:);
        score = des(vars + 1);
        if points
            normscore = ((score - (1/pmax)) / (1 - 1/pmax)) / cloudsep + (1 - 1/cloudsep);
        else
            normscore = 0;
        end

        % Generate variable locations for data point
        points = [];
        for j = 1:vars+1
            points = [points; j-1 + normscore, des(j)];
        end

        % If the data point has a performance score, plot it
        if ~isnan(score)

            % Determine color for lines based on score
            scorecolor = (score - (1/pmax)) / (1 - 1/pmax);
            color = (scorecolor + (1-score)/4) * [.8 .8 .8];

            % Set plotting properties for highlighted designs
```

```
            fit = false;
            if highlights
                    linewidth = lwcallout;
                    linetype = clinetype;
                    color = ccallout;
                    fit = true;
            end

            % Identify parallel coordinates to plot
            if plotscore
                plotpoints = [points(:,1), points(:,2)];
            else
                plotpoints = [points(1:vars,1), points(1:vars,2)];
            end

            % Plot design point in parallel coordinates
            if (lines || fit)

                plot(plotpoints(:,2), plotpoints(:,1), 'Color', ...
                      color,'linewidth',linewidth,'linestyle',linetype);
                hold all;

                % Plot highlights seperately, including label
                if highlights
                    length = size(plotpoints,1);
                    label = 'sample';
                    text(plotpoints(length,2), ...
                      plotpoints(length,1),label,'Color','k','BackgroundColor','none')
                    hold all;
                end
            end
        end
    end

    % Set plotting properties
    if plotscore
        if points
            varlength = vars + 1;
        else
            varlength = vars;
        end
    else
        if points
            varlength = vars;
        else
            varlength = vars-1;
        end
    end

    axis([0 1 0 varlength]);
    set(gca,'YTick',[0:1:varlength]);
    set(gca,'XTick',[0:.5:1]);
    pbaspect([1 varlength/2 1]);
    box off;

end
```

References

Allen, E., & Iano, J. (2012). *The Architect's Studio Companion: Rules of Thumb for Preliminary Design*. John Wiley & Sons.

Allen, E., & Zalewski, W. (2009). *Form and Forces: Designing Efficient, Expressive Structures*. Hoboken, NJ: John Wiley & Sons.

Anderberg, M. R. (1973). *Cluster Analysis for Applications*. New York: Academic Press.

Balling, R. (1999). Design by shopping: A new paradigm? *Proceedings of the Third World Congress of structural and multidisciplinary optimization (WCSMO-3)*, (pp. 295-297).

Billington, D. (1983). *The Tower and the Bridge*. New York: Basic Books.

Blasco, X., Herrero, J. M., Sanchis, J., & & Martínez, M. (2007). Decision making graphical tool for multiobjective optimization problems. *Bio-inspired Modeling of Cognitive Tasks*, 568-577.

Carlsen, D., Malone, M., Kollat, J., & Simpson, T. W. (2008). Evaluating the performance of visual steering commands for user-guided pareto frontier sampling during trade space exploration. *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. 499-509). American Society of Mechanical Engineers.

Carr, D. B., Littlefield, R. J., Nicholson, W. L., & Littlefield, J. S. (1987). Scatterplot matrix techniques for large N. *Journal of the American Statistical Association, 82*(398), 424-436.

Chernoff, H. (1973). The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 361-368.

de Weck, O. L., & Jones, M. B. (2006). Isoperformance: Analysis and design of complex systems with desired outcomes. *Systems engineering, 9*(1), 45-61.

de Weck, O. L., & Miller, D. W. (2002). Multivariable isoperformance methodology for precision opto-mechanical system. *AIAA-2002-1420, 43rd AIAA/ASME /ASCE/AHSStruct Struct Dyn Mater Conf.*

Draper, G., Livnat, Y., & Riesenfeld, R. F. (2009). A survey of radial methods for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 759-776.

Elmqvist, N., ragicevic, P., & Fekete, J. D. (2008). Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics, 14*(6), 1539-1148.

Forrester, A., Sobester, A., & Keane, A. (2008). *Engineering Design via Surrogate Modeling: A Practical Guide.* West Sussex, UK: John Wiley & Sons.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). New York: Springer.

Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer, 1*(2), 69-91.

Inselberg, A. (2009). *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications.* Dordrecht: Springer.

Jordan, D. D., Spencer, D. B., Simpson, T. W., Yukish, M. A., & Stump, G. M. (2008). Optimal spacecraft trajectories via visual trade space exploration. *AAS/AIAA Space Flight Mechanics Meeting*, (pp. 27-31).

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding Data in Groups: An Introduction to Cluster Analysis.* New York: John Wiley & Sons.

Kollat, J. B., & Reed, P. (2007). A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (VIDEO). *Environmental Modelling & Software, 22*(12), 1691-1704.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, (pp. 281–297).

MathWorks. (2014, May). *Design generalized regression neural network.* Retrieved from MATLAB R2014a Documentation: http://www.mathworks.com/help/nnet/ref/newgrnn.html

MathWorks. (2014, May). *Glyph plots.* Retrieved from MATLAB R2014a Documentation: http://www.mathworks.com/help/stats/glyphplot.html

MathWorks. (2014, May). *K-means clustering*. Retrieved from MATLAB R2014a Documentation: http://www.mathworks.com/help/stats/kmeans.html

Mazurek, A., Baker, W., & Tort, C. (2011). Geometrical aspects of optimum truss like structures. *Structural and Multidiscplinary Optimization, 43*(2), 231-242.

Michell, A. G. (1904). The limits of economy of material in frame-structures. *Philosophical Magazine, 8*(47), 589-597.

Monmonier, M. (1989). Geographic brushing: Enhancing exploratory analysis of the scatterplot matrix. *Geographical analysis, 21*(1), 81-84.

Mueller, C., & Ochsendorf, J. (2013). From analysis to design: A new computational strategy for structural creativity. *Proceedings of the 2nd International Workshop on Design in Civil and Environmental Engineering*, (pp. 46-56).

Quiepo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., & Tucker, P. K. (2005). Surrogate-based analysis and optimization. *Progress in Aerospace Sciences, 41*, 1–28.

Ribarsky, W., Ayers, E., Eble, J., & Mukherjea, S. (1994). Glyphmaker: creating customized visualizations of complex data. *Computer, 27*(7), 57-64.

Samyn, P. (2004). *Etude De La Morphologie Des Structures: a L'aide Des Indicateurs De Volume Et De Deplacement*. Brussels: Académie Royale de Belgique.

Simon, H. (1956). Rational choice and the structure of the environment. *Psychological review, 63*(2), 129-138.

Simpson, T. W., & Meckesheimer, M. (2004). Evaluation of a graphical design interface for design space visualization. *Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics & materials conference*. Palm Springs, CA.

Specht, D. F. (1991). A general regression neural network. *Neural Networks, IEEE Transactions, 2*(6), 568-576.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics, 29*(2), 143-151.

Stump, G. M., Yukish, M., Simpson, T. W., & Harris, E. N. (2003). Design space visualization and its application to a design by shopping paradigm. *ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. 795-804). American Society of Mechanical Engineers.

Swayne, D. F., Cook, D., & Buja, A. (1998). XGobi: Interactive dynamic data visualization in the X Window System. *Journal of Computational and Graphical Statistics, 7*(1), 113-130.

Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on optimization, 7*(1), 1-25.

Tufte, E. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.

Tufte, E. (1990). *Envisioning Information*. Cheshire, CT: Graphics Press.

Young, F. W., Valero-Mora, P. M., & Friendly, M. (2011). *Visual statistics: seeing data with dynamic interactive graphics*. Hoboken, NJ: John Wiley & Sons.

Zhang, X., Simpson, T., Frecker, M., & Lesieutre, G. (2012). Supporting knowledge exploration and discovery in multi-dimensional data with interactive multiscale visualisation. *Journal of Engineering Design, 23*(1), 23-47.