Development of an interactive engineering design optimization framework

by

Adam Cutbill

B.A.Sc., Simon Fraser University, 2012

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Mechatronic Systems Engineering
Faculty of Applied Sciences

Approval

Name:	**Adam Cutbill**
Degree:	**Master of Applied Science**
Title:	***Development of an interactive engineering design optimization framework.***
Examining Committee:	**Chair: Behraad Bahreyni** Associate Professor

G. Gary Wang
Senior Supervisor
Professor

Krishna Vijayaraghavan
Supervisor
Assistant Professor

Edward Park
Internal Examiner
Professor
School of Mechatronic Systems Engineering

Date Defended/Approved: October 29th , 2014

Abstract

Engineering optimization is often completely automated after initial problem formulation. Although purely algorithmic approaches are attractive, keeping the engineer out-of-the-loop also suffers from key drawbacks. First, problem formulation is a challenging task and a poorly formulated problem often causes extra efforts and extended optimization time. Second, stakeholders may not trust the results of an optimization algorithm when presented without context. This thesis uses information visualization to keep designer in-the-loop during design optimization formulation, modeling, optimization, and result interpretation stages. Parallel coordinates is the central representation used, accompanied by two-dimensional projections for navigation and a scatterplot matrix for overview. Methods are presented to split the design and performance spaces into meaningful regions by clustering and by interaction. A new data-mining technique is also presented to find relationships between black-box constraints to remove redundant and unimportant constraints. A software prototype is developed and successfully applied to an automotive assembly optimization problem.

Keywords: Interactive optimization; engineering design; black-box optimization; information visualization; constraint redundancy identification; parallel coordinates

Acknowledgements

First, I owe my deepest gratitude to my senior supervisor, Dr. G Gary Wang. Dr. Wang has provided me with countless opportunities, support, and has served as a mentor from the beginning of my journey at SFU. Learning from him, has been a privilege. Throughout the preparation of this thesis and associated work, his support, professionalism, constructive criticism and encouragement have been outstanding. He truly cares about his work and students. It was an honor to work at PDOL.

Additionally, I would like to thank my supervisor, Dr. Krishna Vijayaraghavan, for taking the time to serve on my supervisory committee. Dr. Behraad Bahreyni, my committee chair, has also helped me throughout my time at SFU. He was an incredibly knowledgeable and kind supervisor during my undergraduate capstone project, and I would like to sincerely thank him for serving as chair for this work as well. Dr. Edward Park is an extraordinary instructor. I would like to personally thank him for serving as examiner, and for teaching some my favorite courses at SFU.

Of course, I'd like to express gratitude to my parents for their understanding, patience and encouragement as I completed this work. Last, but not least, I thank my lab mates, classmates and friends for being supportive, friendly and simply great people to work with.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronyms

FEA	Finite element analysis
CAD	Computer aided design
InfoVis	Information visualization
SciVis	Scientific visualization
PCP	Parallel coordinates plot
PCA	Principal components analysis
FA	Factor analysis
IEC	Interactive evolutionary computation
HuGS	Human guided search
VDS	Visual design steering
ATSV	ARL Trade space visualizer
MDO	Multidisciplinary design optimization
PPV	Physical programming based visualization
SC	Six-hump camel back optimization problem
PV	Pressure-vessel optimization problem
PR	Polynomial regression
HDMR	High-dimensional model representations
MBDO	Metamodel based design optimization
SSE	Sum of squared errors
RMSE	Root mean squared error
NRMSE	Normalized root-mean squared error
CV	Cross validation
RMSECV	Root mean squared error of cross validation
NRMSECV	Normalized root-mean square error of cross validation
MPS	Mode pursuing sampling
TR-MPS	Trust region mode pursuing sampling
SFU	Simon Fraser University
KPC	Key product characteristic

Notation

$\boldsymbol{X}$	Matrix of sample points
$\boldsymbol{x}$	Individual sample point
$\boldsymbol{x}^*$	Optimum sample point
$\boldsymbol{x}_s$	Standardized sample point
$\boldsymbol{x}_s'$	Standardized and centered sample point
m	Number of sampled points
$\boldsymbol{f}$	Vector of performance values
$\boldsymbol{f}(\boldsymbol{x})$	Objective function(s) mapping sample designs to performance
$\boldsymbol{g}(\boldsymbol{x})$	Vector of inequality constraint functions
$\boldsymbol{h}(\boldsymbol{x})$	Vector of equality constraint functions
$\boldsymbol{w}$	Weights of individual objectives in aggregated objective
$\mathcal{D}$	Design space
$\boldsymbol{D}_l$	Design space lower bounds vector
$\boldsymbol{D}_U$	Design space upper bounds vector
$\mathcal{S}$	Search space
$\boldsymbol{S}_l$	Search space lower bounds vector
$\boldsymbol{S}_U$	Search space upper bounds vector
$\mathcal{F}$	Performance space
$\boldsymbol{K}$	Percentile region
$\boldsymbol{P}$	Performance region
$\boldsymbol{D}$	Design region
L_2^2	Squared Euclidean distance
$\boldsymbol{x}^c$	Cluster centroid
$d(\mathrm{a},\mathrm{b})$	Distance between points a and b
n	Number of design variables to be optimized
$p(\mathrm{a},\mathrm{b})$	Proximity between clusters a and b
k	Number of clusters in k-mean clustering or number of models in k-fold CV
ϵ	Distance cut-off in agglomerative clustering
R	Tank radius in the pressure-vessel problem
T_s	Tank shell thickness in the pressure-vessel problem
L	Tank length in the pressure-vessel problem
T_h	Tank head thickness in the pressure-vessel problem

$\boldsymbol{S}$	Covariance matrix (used for PCA)
$\boldsymbol{\beta}$	Coefficient vector for terms in polynomial regression
$\boldsymbol{H}$	Set of possible metamodels of a given form (hypotheses)
$\hat{h}$	Metamodel with lowest sum of squared errors
$\hat{\boldsymbol{\beta}}$	Coefficients of $\hat{h}$.
$\boldsymbol{X_T}$	Training dataset
$\boldsymbol{X_A}$	Additional testing dataset
$\boldsymbol{X_v}$	Validation dataset
μ	Mean
$\boldsymbol{e}(\boldsymbol{a},\boldsymbol{b})$	Vector of errors between sets $\boldsymbol{a},\boldsymbol{b}$, as defined by the error function
$e(\boldsymbol{a},\boldsymbol{b})$	Error function between sets $\boldsymbol{a},\boldsymbol{b}$ (RSME, RMSECV, NRSME or NRMSECV)
$\boldsymbol{e}$	Binary vector of constraint violations (Chapter 6)
$\boldsymbol{y}$	Binary vector signaling constraint redundancy (0) or necessity (1)
$\boldsymbol{C}$	A set of items (constraints, for this thesis).
$\boldsymbol{C_B}$	A set of constraints that are co-occur (are violated as a bundle).
$\boldsymbol{\sigma}(\boldsymbol{I})$	The count of observations where ALL items occur in set I.
$\boldsymbol{\gamma}(\boldsymbol{I})$	The count of observations where ANY items occur in set I.
N	Number of observations (Chapter 6).
$s(\boldsymbol{C})$	The support of set $\boldsymbol{I}$.
$j(\boldsymbol{C})$	The Jaccard similarity of set $\boldsymbol{I}$.
$c(\boldsymbol{C_A} \rightarrow \boldsymbol{C_B})$	The confidence of the association rule $\boldsymbol{A} \rightarrow \boldsymbol{B}$.
nT	Limit on number of samples for constraint checking (Chapter 6)
nS	Number of sample points generated for each iteration of constraint checking (Chapter 6)
nI_S	Maximum number of iterations without change in constraint rules (Chapter 6)
V^2	Quantity proportional to variance in KPC point distance from nominal

Chapter 1. Introduction

1.1. Preliminaries

In the information age, a wealth of data is recorded daily. Consumer transactions are logged by loyalty programs, videos are voted for by viewers and the physical world is digitized by sensors. Indeed, there is so much data available that the majority is stored in data warehouses and left untouched. For a sense of scale, in 2010, Facebook processed 30 billion digital items every month, including photos, videos and comments [1]. Globally, it is estimated that 9.57 zettabytes of data (ten million million gigabytes) were processed by enterprise servers in 2008 [2]. The data overload has called for rapid growth in data-driven research which merges information visualization, machine learning and data mining. Academics and industry, in many disciplines, are constantly looking to turn *raw data* into valuable *information*. However, data research has progressed so rapidly, due to demand, that it has created a knowledge gap between the state-of-the-art and domain specific applications. This thesis aims at partially bridging the gap between interactive information visualization and engineering design optimization.

With the computational capabilities available today, engineers are increasingly encouraged to simulate complex models on computers [3]. For instance, structural analysis can now be performed by Finite Element Analysis (FEA) as opposed to solving lengthy algebraic equations. In software, design performance is modeled and estimated before building physical prototypes. As an additional step, software models may be rapidly optimized, with optimization algorithms, leading to a shorter design process, reduced costs and improved products. Sequentially, the engineer provides the model, objectives, constraints, and design variables (i.e. the problem formulation). Next, the algorithm tunes the variables to yield the best performance (according to the objectives and constraints). This automated design process is data driven; yet, optimization is rarely visualized or made interactive.

The lack of transparency in optimum design creates three major pitfalls. First, it is challenging for engineers to catch mistakes or redundancies in their problem formulations. Second, after clicking “start”, designers are unable to steer the optimization process using their expertise or preferences; typically, the algorithm controls where to search for the optimum. Third, when an optimum solution is found, the result may be unconvincing if presented without context. In fact, it may very well be incorrect due to an error in the model, an error in the problem formulation, or an error in the algorithm.

This thesis focuses on improving transparency in optimization by visualizing and mining the data that is iteratively generated. By keeping the user engaged, they are able to provide input throughout the optimization process. Interactive optimization is not a new concept and there is plenty of research dedicated to the topic, as explained in the literature review section. This work, in particular, introduces an integrated framework for interactive optimization with a focus on multivariate visualization. The specific goal is to promote transparency and interaction in single objective optimization of black-box models.

1.1.1. Numerical Optimization and Black-box Optimization

Numerical optimization is a well-developed mathematical topic whose scope includes tuning design variables to find the *best* design(s) according to an objective function and constraints. Although gradient based optimization dates back to Isaac Newton and the beginning of calculus, modern optimization was established during World War II by the British Air Forces’ *Operational Research* unit. The unit originally evaluated how to redesign weapons and equipment, but eventually grew in scope to predict battle outcomes and influence policy using their algorithms [4]. Early algorithms, such as the simplex method [5], were algebraic and deterministic, but were restricted to systems of linear equations. This type of problem is now called linear programming (or quadratic programming for quadratic models).

As models grew in complexity, new techniques became necessary to broaden optimization’s scope. Many modern methods are now iterative and stochastic (i.e. they do not guarantee the same output each run), but they can be applied to more realistic models without the restriction of linearity or even continuity. Famous approaches include genetic algorithms--based on evolution [6], simulated annealing--based on statistical mechanics [7], and particle swarms--based on swarm social behaviours [8].

Figure 1 shows the process followed by a typical iterative stochastic optimization algorithm. The goal is formulated as an objective or cost function, which is minimized by testing candidate designs until stopping conditions are reached. Once the process is terminated, a result is presented (usually as text).

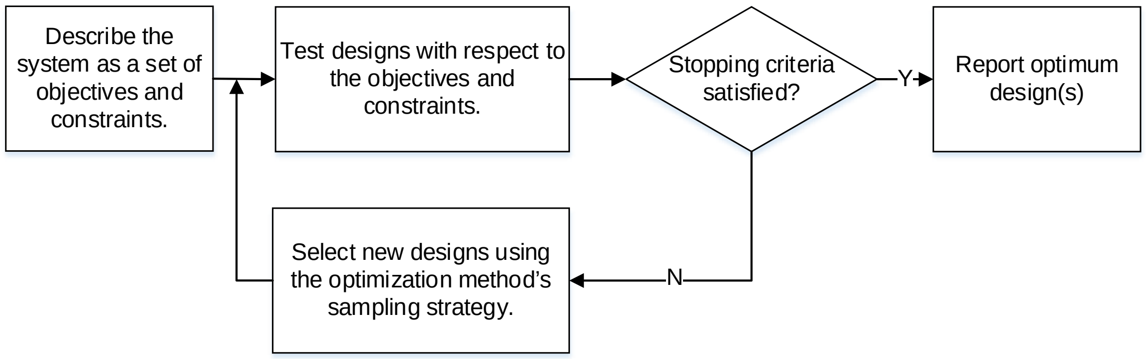

Describe the system as a set of objectives and constraints.
Test designs with respect to the objectives and constraints.
Stopping criteria satisfied?
Y
Report optimum design(s)
N
Select new designs using the optimization method's sampling strategy.

Figure 1 Numerical optimization flowchart

As mentioned, optimization traditionally involves algebraic cost functions. However, with modern methods, improving the response from an engineering simulation is also a valid goal. In these cases, the model is treated as a *black-box* [9]. The simulation's inputs and outputs are known to the optimization algorithm, but the physical phenomenon being simulated is not. As an analogy, consider testing the input and output voltages of an integrated circuit (IC). The behaviour of the IC can be approximated using an oscilloscope without knowing its internal components. Similarly, black-box optimization extends optimization theory to simulations which are too complex to evaluate algebraically (e.g. Finite Element Analysis, or FEA), by considering only the input and output response. This is the central application of the techniques presented in this thesis.

The advantage of numerical optimization and automated design is self-evident. A large portion of the design effort is efficiently executed by a computer instead of an expert. Efficiency matters as a single test simulation may require hours to complete, even on state-of-the-art workstations [9]. In practice, there may also be millions of feasible designs to test. Therefore, a robust mathematical strategy to intelligently find an optimum design, with as few simulations as possible, is highly desirable. This is the focus of computationally intensive optimization research. There is no exact numerical definition of “computationally intensive”. However, a simulation that requires more than one minute can be costly for optimization purposes (especially if there are many variables to optimize).

As exciting as automated design may appear, removing engineers from the design process (after problem formulation), is inherently problematic. First, it is likely that the problem formulation will initially contain errors. Problem formulation requires engineers to abstract physical systems into computer models while maintaining as much fidelity as possible. Consequently, making invalid assumptions and mistakes in modeling is natural. In addition to errors in modeling, design optimization requires precise descriptions of objectives, constraints, variables, and search bounds. All of these parameters are also difficult to define *a priori*.

Furthermore, there is no consensus on what types of formulation errors are most common. For instance, Messac argues that important constraints are often missing for large scale problems [10] while Karwan et al. explains that constraints are often redundantly defined in linear programming [11]. Additionally, Balling found that even if a formulation is sensible, stakeholders may not be able to define the objective they truly want until they see some results [12]. The bottom line is that correctly formulating an optimization problem is challenging. Therefore, methods to assist in problem formulation are required.

Aside from improper formulation, another challenge in automated design is to convince the design engineer (and co-workers) that an optimization result is in fact optimal or sensible [13]. Even if the problem is formulated as intended; the result may conflict with the designer’s intuition. The optimum may have far better performance than expected, or include unanticipated variable values (outliers). By keeping designers engaged during optimization, and allowing for exploration of the data afterwards, the results are less surprising. However, to remain in the loop, designers need a mental model of how the changing design variable values affect the output. This can be achieved by visualization.

1.1.2. Visualization

Visualization is presenting data to an audience to form mental models. The aim is to help people understand data, and the real-world phenomena it represents, by leveraging our well-developed cognitive systems. In other words, data is encoded in a manner that can be quickly perceived, accurately interpreted and clearly understood (by the intended audience). Information visualization often focuses on presenting abstract data to illustrate a particular point. Tufte’s book, *The Visual Display of Quantitative Information* [14], presents many examples of successful visualizations that trigger understanding and assist in decision making, as well as unsuccessful visualizations that are incomprehensible or misleading. Visualization is also a powerful tool for explorative analysis where hypotheses are generated by interacting with the data. Such is the case in exploratory data analysis, popularized by Tukey in the 1970s [15].

As an example, consider Anscombe's quartet data [16] tabularized in Table 1. Each set of x-y pairs, from I to IV, has the same x and y means and the same variance. If analyzed with descriptive statistics, the data appears to have come from the same population. However, when visualized as scatterplots in Figure 2, the datasets are distinct: set I is linear, set II is quadratic, set III is linear with an outlier and set IV is constant in x values with an outlier. This exemplifies that the manner that data is represented (e.g. table or scatterplot) strongly affects its interpretation.

Although information has been visualized for centuries, the field’s popularity has jumped dramatically in the past few decades; triggered by exponential increases in computing power and storage [17]. In this time, visualization has grown from static graphs, focusing on a few attributes, to fully featured multivariate visualization platforms such as GGobi [18] and the R programming language [19]. Indeed, modern visualization is highly coupled with statistics, machine learning, mathematics, data mining, user interaction, animation, and navigation.

Table 1 Anscombe's quartet as a table

	I		II		III		IV	
Observation	x	y	x	y	x	y	x	y
1	10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
2	8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
3	13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
4	9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
5	11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
6	14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
7	6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
8	4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
9	12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
10	7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
11	5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89
Mean	9	7.51	9	7.51	9	7.51	9	7.51
Standard Deviation	3.32	2.03	3.32	2.03	3.32	2.03	3.32	2.03

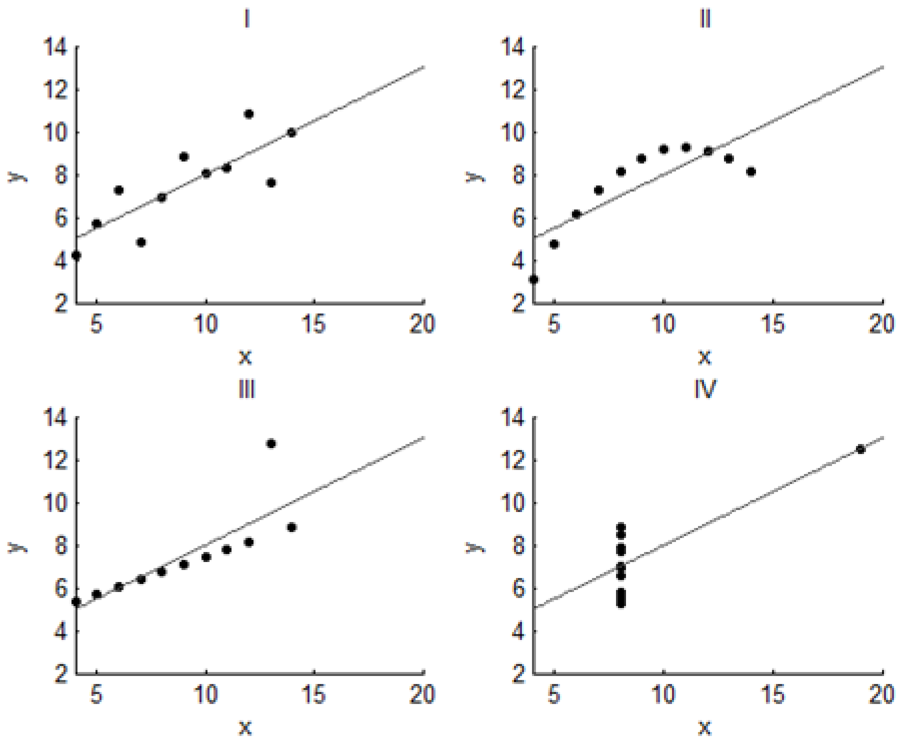


Figure 2 Anscombe's quartet as scatterplots

Due to the advancements made in data visualization and related fields, analysts are more empowered now than ever. For instance, in 2011, researchers identified subgroups in breast cancer patients that were previously unknown by simply applying new visual analysis techniques to existing genomic data [20]. In computer aided design (CAD), scientific visualization shows physical phenomena such as stress in structures, current flow in circuits and fluid flows in turbines. Outside of CAD environments, information visualization is an area of interest to keep designers engaged during design optimization [21,22]. This thesis presents a framework for visualizing optimization data as it is iteratively sampled.

1.2. Scope

The scope of the work presented is limited to visualization and mining of data in an engineering design optimization context. The focus is on visualizing the design space and performance spaces for optimization problems by using parallel coordinates, enhanced with basic data mining techniques. A method to mine for relationships among constraints, during problem formulation, is also presented. In terms of interaction, queries and clustering are used to split the design space into regions which have similar performance or variable values. Parallel coordinates is the central representation used to illustrate this task. Furthermore, scatterplots are used for navigation and overview. Regions may also be locally modeled with polynomial regression or high dimensional model representation (HDMR).

As this is visualization in an optimization context, the visualization is connected directly to a computationally efficient optimization algorithm (TR-MPS), allowing for on-the-fly visual design steering. The convergence of the algorithm is also shown iteratively, as the optimization progresses. Multi-objective and multidisciplinary optimization visualization is beyond the scope of this thesis. The focus is on visualizing the effect of many variables on a single objective with constraints, and on visualizing optimization progress iteratively. This contrasts the majority of work done in visualization for optimization which focuses on visualizing the effect of preferences/weights in multi-objective problems [23,24].

1.3. Research Goals

This research aims to develop methods and tools to help design engineers

1) To visually steer design optimization in multivariate spaces via region selection and visual feedback.

2) To visualize the progress of optimization for multivariate problems, and

3) To identify relationships among constraints in the problem formulation.

1.4. Thesis Structure

This thesis is split into eight chapters. The introductory chapter provided motivation for the work and specific goals. The following literature survey chapter (Chapter 2) introduces information visualization, its past uses in interactive optimization and the state-of-the-art. Next, the methodology chapter (Chapter 3) provides necessary theory for numerical optimization, regions, clustering, principal component analysis, interaction, and navigation. Once regions are defined, metamodels can be built with polynomial regression or HDMR as explained in Chapter 4, which covers local metamodeling. Chapter 5 introduces a framework which aggregates concepts from information visualization into prototype software for interactive optimization. In Chapter 6, a new method is introduced to identify relationships among constraints, incorporating association analysis. Chapter 7 shows how the developed framework may be used in locator optimization for automotive assembly fixtures. Finally, Chapter 8 provides summary of the work and presents potential future work.

Chapter 2. Literature Survey

This chapter provides an introduction to information visualization and its past use for optimization. A discussion is provided that covers some of the key concepts, with a focus on multivariate visualization techniques. In addition, visualization used specifically for design optimization is discussed.

2.1. Information Visualization

Visualization in engineering generally comes in two forms, information visualization (InfoVis) and scientific visualization (SciVis) [17]. Scientific visualization is likely what comes to mind when considering visualizing in an engineering context. In SciVis, the data is projected onto a simulated physical model (e.g. a CAD model) to show stress, temperature or other physical properties of a system. Information visualization (InfoVis), on the other hand, is not tied to a spatial model (e.g. a scatter plot). The growth in scope of InfoVis (including data mining, statistics etc.) has also spawned the term *visual analytics* to reflect the broadness of the field [25].

Although the need for taxonomy has been debated [26], it is important to clarify that the work in this thesis treats data in an abstract (InfoVis) sense. This ensures that the methods can be applied to various black-box problems without being tied to CAD. For instance, consider the scientific visualization of an FEA model for a bridge in Figure 3, colored by deformation (in [mm]), under a distributed load. The colors in this example show the location and magnitude of the deformation. This visualization is clear and intuitive, but ANSYS was used to create it. Had the data come from another CAD package (e.g. SolidWorks), a model from that software would be necessary (along with the ability to map deformation to color and position).

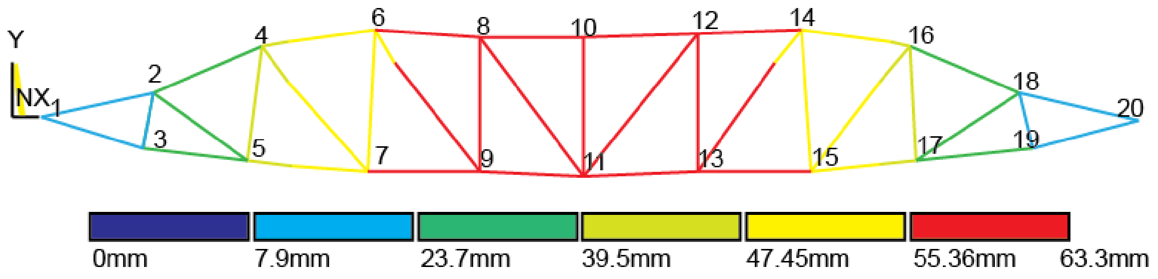


Figure 3 FEA Bridge analysis: scientific visualization

Now, consider Figure 4. The data is the same. Yet, in this representation, the positions of points are dependent only on the value of the data and the chosen representation (a line plot). There is no intrinsic physical notion of space or model tied to the plot. This is information visualization.

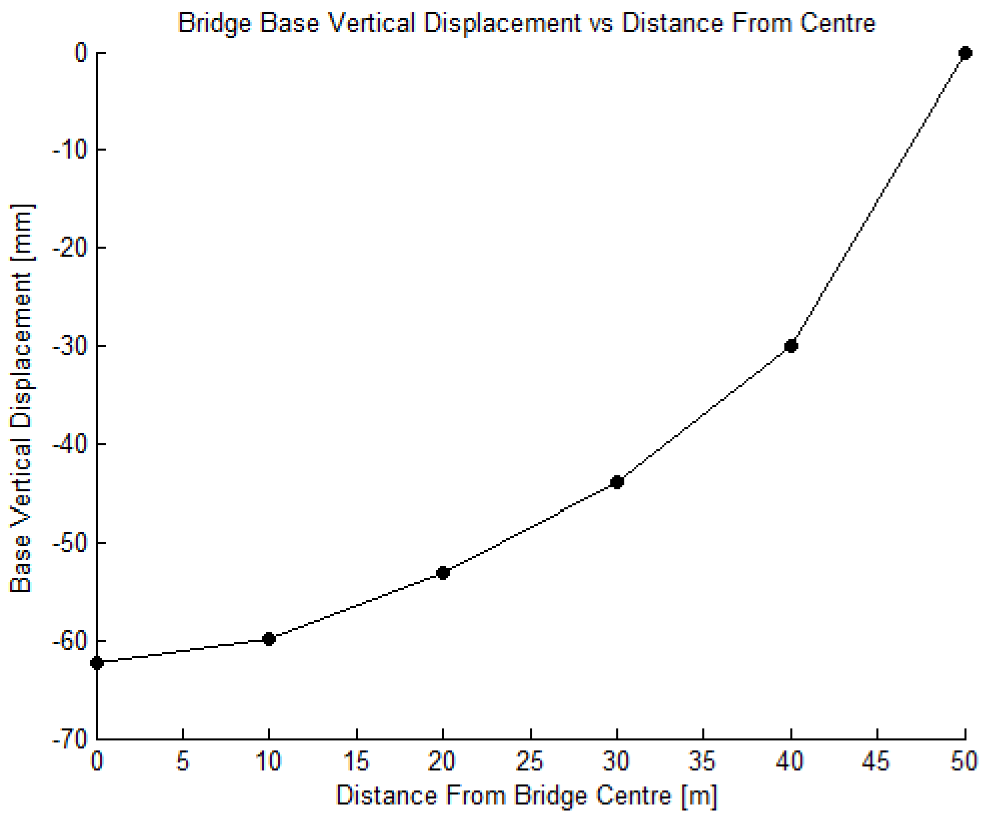
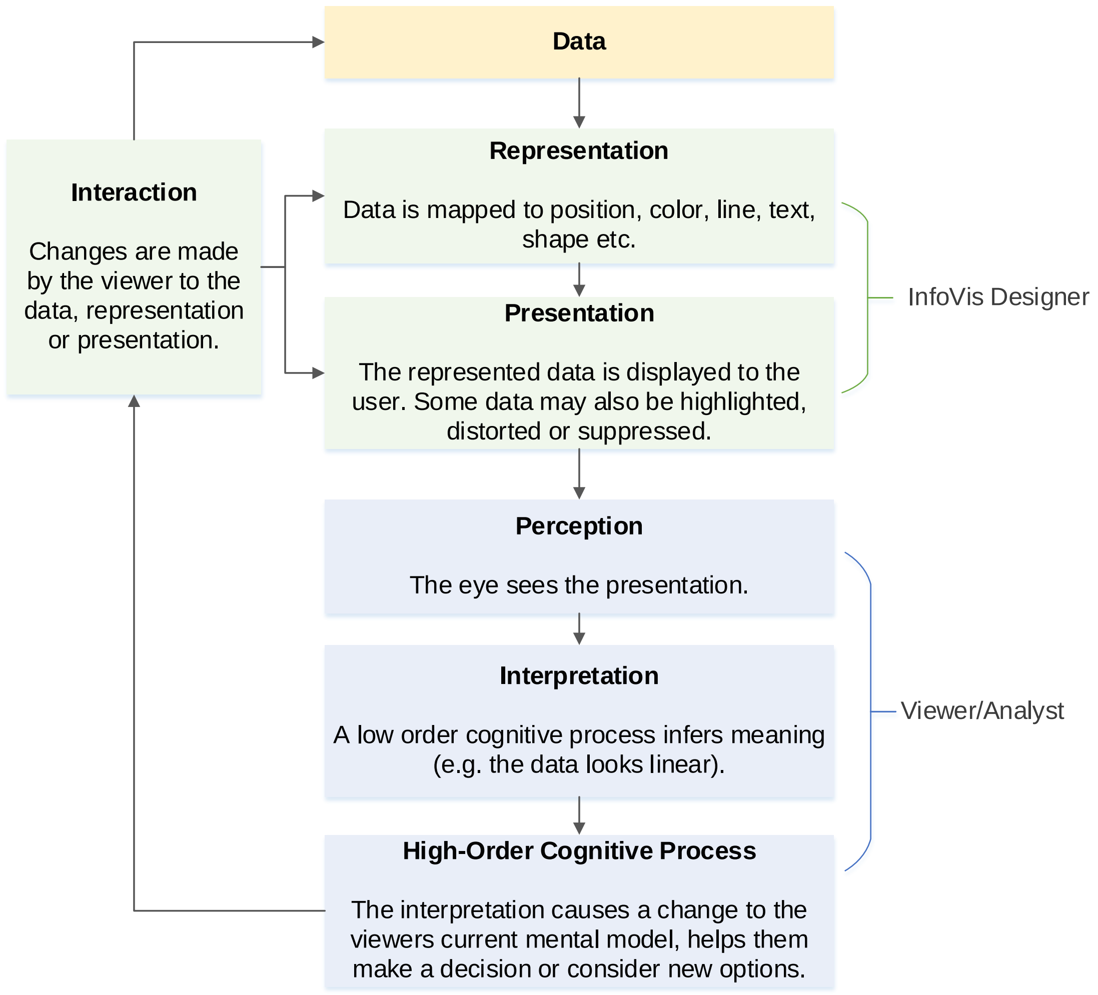


Figure 4 FEA Bridge analysis: information visualization

As a brief breakdown of InfoVis, Figure 5 shows the role of each step. Topics which are controlled by the visualization designer include representation, presentation and interaction. The remaining steps are performed by the viewer.



Data
Representation
Data is mapped to position, color, line, text, shape etc.
Presentation
The represented data is displayed to the user. Some data may also be highlighted, distorted or suppressed.
Interaction
Changes are made by the viewer to the data, representation or presentation.
InfoVis Designer
Perception
The eye sees the presentation.
Interpretation
A low order cognitive process infers meaning (e.g. the data looks linear).
High-Order Cognitive Process
The interpretation causes a change to the viewers current mental model, helps them make a decision or consider new options.
Viewer/Analyst

Figure 5 Flowchart of the visualization process (adapted from [17])

2.1.1. Representation Considerations for Quantitative Data

The representation of quantitative data has a history dating to ancient times, usually as information overlaid on geographic maps. In the 18th century, quantitative data representation was modernized with scatterplots, bar charts and line graphs, popularized by Playfair [14,27]. A representation is a mapping of data to a display for a particular purpose. For instance, in Table 1, the table encodes Anscombe's quartet numerically as organized text, making it easy to find exact values for an x-y pair. In contrast, the scatterplots show trends, but are more difficult to read numerically. When choosing a representation, it is important to consider properties of the data (number of points, dimensionality, and type), as well as the kinds of insight the audience is seeking (relationships, distributions, values, or outliers). Common goals include browsing and generating hypotheses (exploration), testing hypotheses (analytics) or forming summaries that can be described to others (communication).

Extensive research has been conducted in order to determine which encoding methods are possible, and which are most accurately perceived. A pioneer in InfoVis, Jacques Bertin, defines the eight primary visual variables of 2D graphics: x-position, y-position, size, value (opacity), texture, color, orientation, and shape [28]. These are shown in Figure 6. However, not all of these variables are equal in terms of cognition. In 1984, experiments by Cleveland and McGill [29] found that position, length (size), and slope (orientation) are most accurately perceived; not surprisingly, area is poorly perceived. In 1986, Mackinlay, extended this work to non-quantitative data types (Ordinal or Categorical) [30] . These encodings have been used to make many well-known representations, including scatter plots (position), bubble charts (area and position) and bar charts (length and position). A list and discussion of different types of 2D plots is beyond the scope of this work, however many examples can be found in Spence’s book [17].

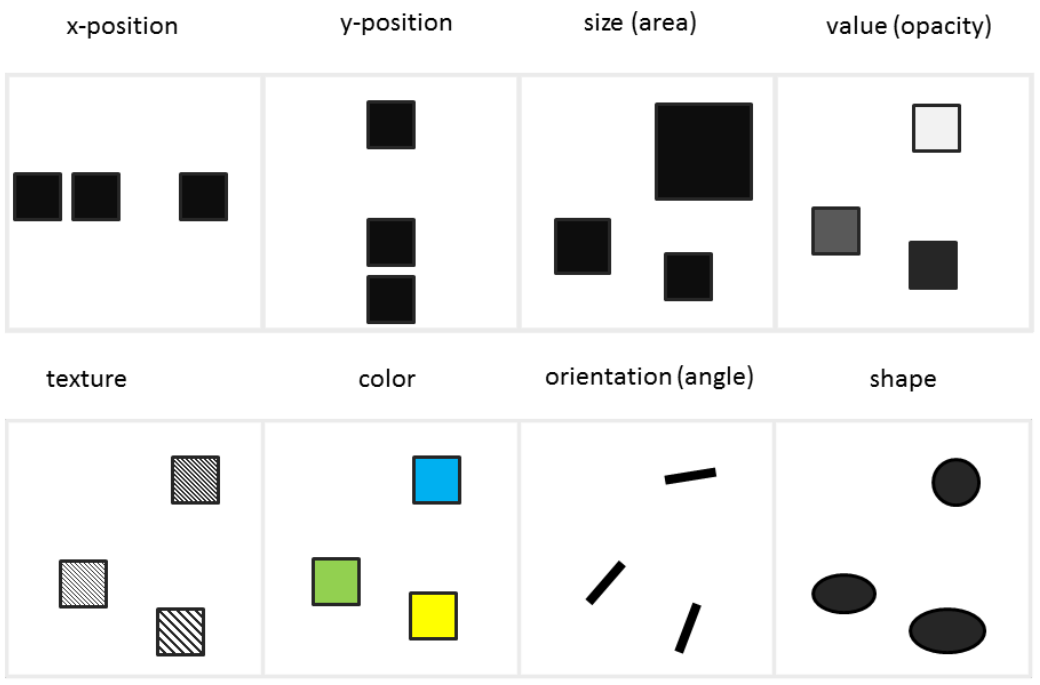


Figure 6 Bertin's visual variables (adapted from [28])

2.2. Multivariate Representations

Visualizing data with more than three attributes is inherently difficult [31]. The challenge is particularly relevant today, as datasets grow, not only in the number of observations, but also in the number of recorded attributes. In engineering design, higher fidelity implies higher dimensionality. The structural design of a beam, for example, includes many factors which may simultaneously affect its stress limits, such as shape, thickness, material, length, and loading conditions.

Visualizations that can concurrently show the relationships between more than three attributes are *multivariate* (a.k.a. *hypervariate*) to indicate there are more dimensions than the medium used for display [17,32]. Although it is possible to simply combine some of Bertin's variables in a scatterplot, to do so would be complex, cognitively inefficient, and limited to eight dimensional data. Instead, alternative representations have been developed. An extensive list of multivariate visualizations and their histories can be found in survey papers by Wong [33] and Grinstein [31]. Furthermore Jones, presents many multivariate methods in the context of optimization and operations research in [34]. Below is a brief introduction to two popular multivariate visualizations and statistical techniques which are used in this thesis. The terms dimensions, attributes, and variables are interchangeable here.

2.2.1. Multivariate Representation Examples

This section shows two possible representations for a famous multivariate dataset, Fisher's Iris data [35]. The dataset is popular for introducing multivariate visualization and statistical techniques in literature. The data summarizes four attributes (Petal Width, Petal Length, Sepal Width, and Sepal Length) of three *Iris* flower species (I.Setosa, I.Versicolo, I. Virginica) corresponding to red, green, and blue respectively.

The scatterplot matrix

The scatterplot matrix, shown in Figure 7, is a group of scatterplots organized such that each row and column corresponds to one attribute. Scatterplot matrices are intuitive for identifying correlations, clusters, and groups in two-dimensional subspaces.

Although they are simple to understand, there are a couple of clear disadvantages to scatterplot matrices. First, the number of scatterplots is proportional to the square of the number of attributes. Although, the number of subplots can be halved by only showing the plots below or above the diagonal, the number of plots remains prohibitively large and limits plotting space. Second, only two-dimensional relationships are shown in each square, meaning the audience has to piece together higher order relationships. Nonetheless, scatterplot matrices are effective overviews of data [36,37]. It is common to use these plots as a starting point to find interesting pairs of variables to explore further.

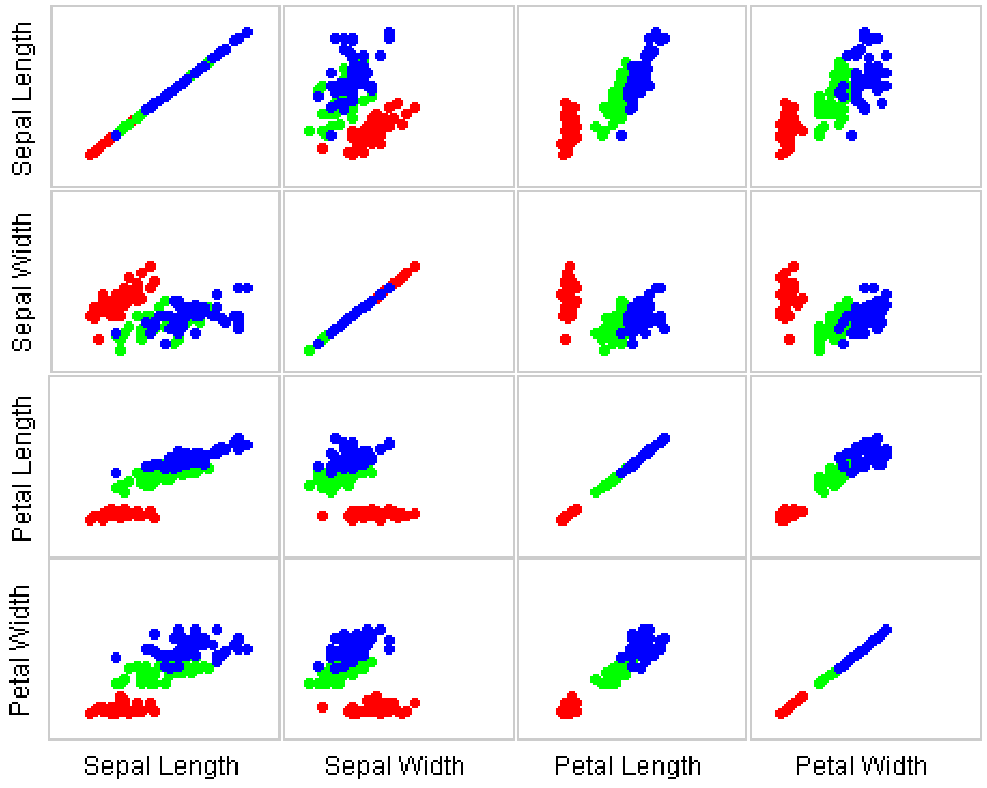


Figure 7 Scatterplot matrix of Fisher's iris data

The parallel coordinates plot

The parallel coordinates plot (PCP), established by Inselberg in 1985 [38], draws each observation as a polyline, which crosses a set of parallel axes. The value for each attribute is indicated by where the observation intersects its axis as shown in Figure 8. For example, the *I.Setosa* species, shown in red, has a small sepal length, a generally larger sepal width, and small petals. There also appears to be an outlier in the *I.Setosa* sepal width. Many parallel axes can be placed adjacent to each other without a significant increase in computational cost or space. It is also possible to quickly identify multivariate relationships and groups directly from a PCP.

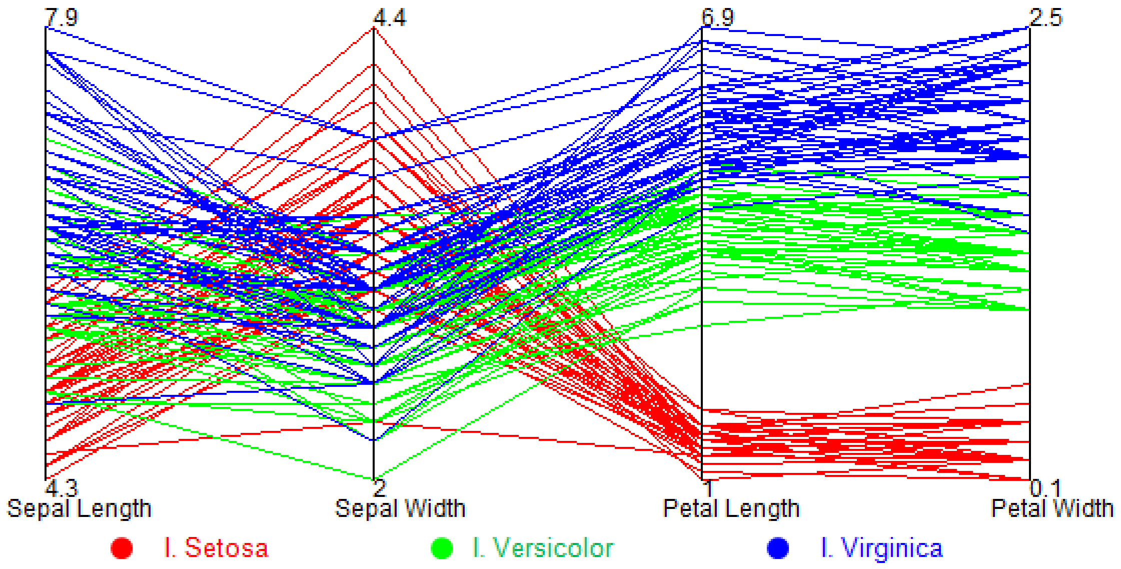


Figure 8 Parallel coordinates plot of Fisher's iris data

The major downsides to parallel coordinates are also clear from Figure 8. First, the amount of ink used to show the data is exceptionally large. Tufte argues that graphs should look to minimize the amount of ink used, and increase data-density instead [14]. The line representation of points also causes them to be frequently drawn over one another (overdraw). The crossing lines make it difficult to distinguish between (or to follow) particular points. Also, it is difficult to identify exact numeric values by reading the graph. These limitations have been the subject of extensive research on PCPs in the past three decades [39–43], since Inselberg's paper.

An effective example of summarizing data is the *Hierarchal Parallel Coordinates* method [39]. To reduce the amount of overdraw; Fua et al. cluster observations into subgroups, using hierarchical clustering, which are represented by polygons with varying opacity, instead of lines. By adjusting the clustering parameters, the data can be shown in varying levels of detail. For instance, if the criteria for points to form clusters is strict, many individual points will be shown (as each will form its own cluster); if the criteria is loosened, points will form loose clusters and be summarized as fewer polygons.

In Fua's method, the opacity and size of a polygon is proportional to the variance of the points in the cluster, while the mean is drawn as a solid line. For illustration, the Iris data is shown using polygons below (clustered by species). Typically the metric for clustering is a distance between points, not a class label like species. However, Figure 9 illustrates how clusters can be represented in PCP, using polygons instead of (or in addition to) lines. A simplified implementation of this idea is used in Chapter 3 to show regions as polygons bounding the data. This allows users to see individual points, but also suggests possible sub-regions.

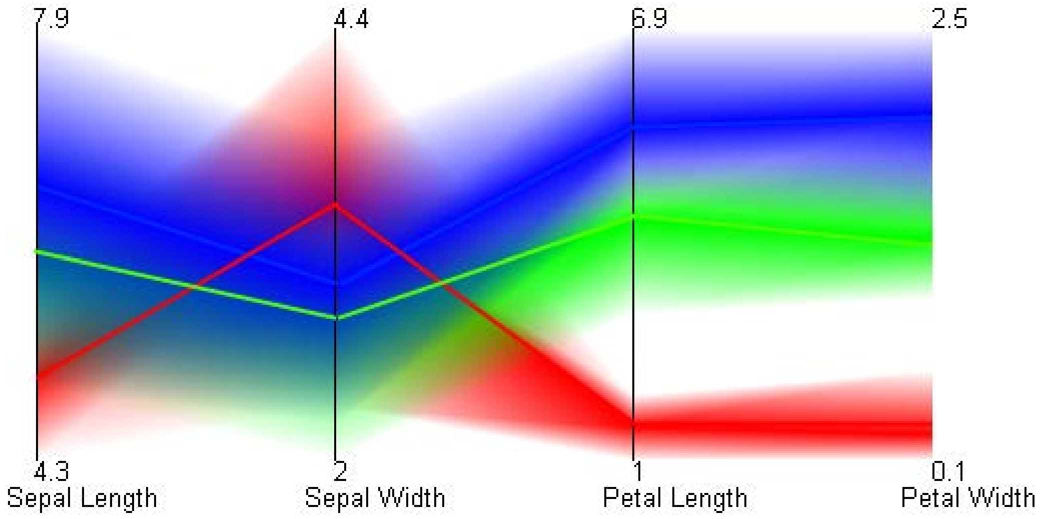


Figure 9 Parallel coordinates plot of Fisher's iris data as polygons

2.2.2. Progress in Multivariate Visualization

Significant progress has been made in multivariate visualization during three intense phases of research beginning in the 1970s. Wong presents an excellent review of the work until the 1990s [33]. In fact, Wong suggests that as of 1992, research has passed the discovery phase and is in the elaboration and assessment phase. In other words, new representation techniques are rarely discovered, nor are they necessary. Still, visualization research remains very active. There is considerable inventiveness required to summarize, present, and interact with data efficiently, especially within domain-specific contexts, like design optimization [17]. Interaction is particularly important for visualization of data sets with thousands of observations and/or attributes

[44]. Research is also needed to rigorously and systematically analyze the effectiveness of current methods for performing various tasks.

2.2.3. Dimensionality Reduction

Aside from advances in representation, dimensionality reduction techniques are also practical to map high dimensional spaces to two or three dimensions. Dimensionality reduction is concerned with mapping high dimensional data to preserve statistical properties, as opposed to mapping data for display (i.e. visualization). This is especially valuable in datasets with many dimensions (>10), or sparse data where most attributes have zero values. In these situations, it is redundant (or impossible) to visualize all of the variables. An alternative is to define a new space that uses fewer variables, but is statistically representative of the high dimensional data.

Principal Component Analysis (PCA) [45], a variation of singular value decomposition, is a popular reduction method that preserves linear variability, in a reduced coordinate system. The first axis of the transformed coordinate system, the 1st principal component, is chosen as the direction which captures the maximum amount of variance. The remaining components are chosen to maximize the remaining variance, with the limitation that they are orthogonal to one another. Although it is a general statistics technique, PCA has strong applications in multivariate visualization. It is especially useful for identifying groups, or isolating points as the data becomes separated along the directions of maximum variance. Figure 10 (left) shows the first two PCA components of the Iris dataset. It is clear that the *I.Setosa*, shown in red, is dissimilar to the other Iris species, by its isolation in the 1st PCA direction. The steps of PCA are explained in section 3.3.2 and Appendix.

Factor analysis (FA) is related to PCA, but uses hidden variables (a.k.a. latent factors) to describe the observations, as opposed to using linear combinations of the observed variables. This means it is applicable even if there is low covariance between the observable variables. For example, with the Iris dataset, a variable corresponding to “Petal Area” may have been recorded instead of petal width and length. Petal width and length may be independent; but, this new *area* variable summarizes both. Factor analysis aims to find a small set of hidden variables with strong correlations to the observable variables. The hidden variables are then linearly combined as a lower dimensional representation of the original data. The technical details of FA are beyond the scope of this work; an introduction can be found in [46]. Figure 10 (right), shows the Iris data plotted for a single latent factor. Again, the *I.Setosa* (red) species is isolated.

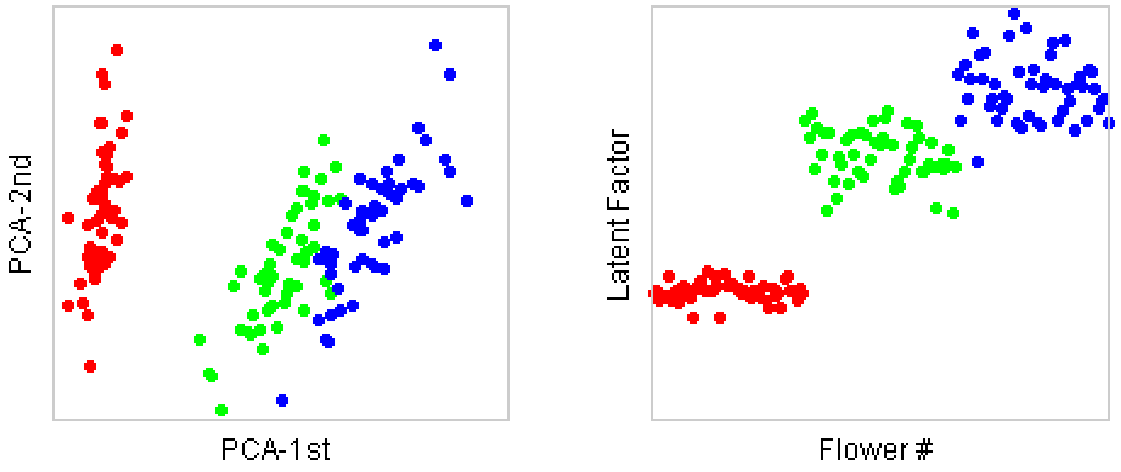


Figure 10 Fisher's iris data [35] with PCA (left) and FA (right)

Dimensionality reduction is not limited to linear factors. Advanced reduction techniques include Kernel PCA [47], locally linear embedding (LLE) [48], multidimensional scaling (MDS) [49] and ISOMAP [50]. Maaten et al. provides a comparative review of twelve of the most popular methods, including the ones listed above in [51].

Although dimensionality reduction is powerful for preserving high dimensional features in low dimensional space, the techniques by themselves suffer from some drawbacks. First, most projection methods, such as PCA and FA, rely on Eigen decomposition, which may be computationally costly and result in non-unique solutions. Second, without knowing what kinds of patterns the data contains (e.g. linear or nonlinear), it is difficult to know which method to use. This is especially true if data is provided for visualization without additional background information, as in black-box optimization.

Perhaps, the most fundamental drawback of dimensionality reduction is that the data is mapped to a new space, and the context information is lost. For instance, in PCA, the components have statistical significance, but are difficult to interpret. Inferring the position of a point in a PCA plot, in terms of the original attributes, is extremely difficult as the point is now rotated and distorted. It is clear that *I.Setosa* is separate from the other flowers in Figure 10, but it is unclear what specific properties make it distinct. This conflicts with the goal of visualization: forming a clear mental model of data that the audience can understand and remember.

Overall, when using reduced spaces it is key to provide interaction that can relate a given point back to the original space. For example, with brushing (as explained in 3.3.1), the reduced space may be used for navigation or region selection, while another plot shows the highlighted points in detail. Dimensionality reduction is also powerful to improve computational efficiency in modeling, clustering, or optimization which also suffer when dimensionality increases (i.e. the *curse of dimensionality* [52]).

2.3. Visualization in Support of Engineering Design

2.3.1. Early Approaches and Applications

Visualization in support of optimum design is relatively young, with domain specific applications published in the 1980s, and the first textbook published in 1996 [34]. Early application specific approaches fell into the category of Interactive Evolutionary Computation (IEC) [53]. In IEC, a human operator acts as the fitness (objective) function and evaluates each design manually. Additional designs are then generated based on the operator's feedback. The process is outlined in Figure 11. IEC is a popular approach for situations where the objective of optimization is difficult to quantify or model. In fact, Takagi found 251 IEC papers published in various fields between the 1990s and 2001 [53]. The disciplines where IEC is used span from graphics and music to data mining and robotics. Fundamentally, this is a human-guided optimization process; however, it is taxing and time consuming to ask an expert to evaluate each potential design.

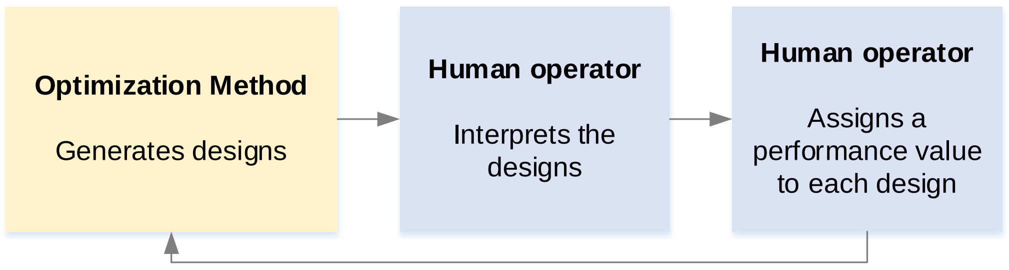


Figure 11 Flowchart of the interactive evolutionary computation (IEC)

In Sims' 1991 work [54], a human operator acts as the fitness function for a 3D plant and flora graphic generator, by assigning an aesthetic value to designs generated in previous iterations. The graphics are rendered and presented to an artist who rates them. A genetic algorithm then automatically generates more graphics in the following iteration, building on the learned operator's preferences. This IEC approach contrasts traditional optimization, where the objective function is defined a priori and evaluated by computer. However, Sims' approach is very logical, as aesthetic success is subjective and difficult to compute.

In 2000, Mitsubishi Electric Research Laboratory used a semi-automated *Human-Guided Search (HuGS)* to find an optimal delivery truck routing schedule, minimizing the number of trucks required to make all deliveries within fixed time windows [55,56]. HuGS is an early example of *Interactive Optimization,* as shown in Figure 12. Further examples of interactive optimization, including HuGS, were also discussed in [57]. A chapter is also dedicated to interactive optimization in Arora's textbook [58]. Interactive optimization uses mathematical objective functions, but allows the designer to adjust the optimization parameters (such as search bounds, constraints or even objectives) without starting an entirely new optimization process.

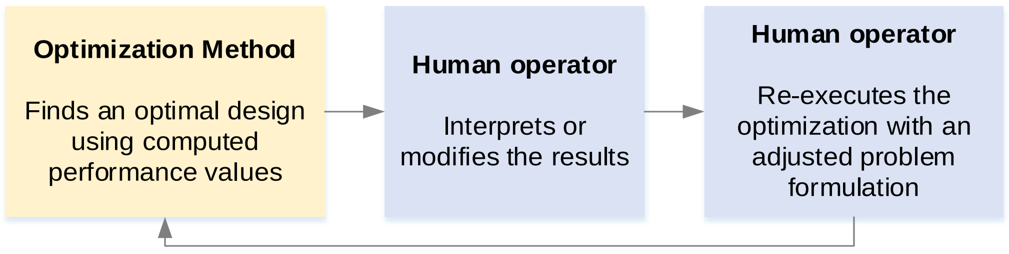

Optimization Method
Finds an optimal design using computed performance values
Human operator
Interprets or modifies the results
Human operator
Re-executes the optimization with an adjusted problem formulation

Figure 12 Flowchart of interactive optimization (adapted from [57])

2.3.2. Visual Design Steering and Graph Morphing

Although early interactive optimization examples were built for specific applications, progress has also been made towards generalization. Winer and Bleobaum defined *Visual Design Steering (VDS),* in 2001, as an umbrella for work which steers computationally intensive optimization towards a solution by visualization [59,60]. In VDS, the algorithm samples in user-selected areas. The pioneering VDS works dates back to Afimiwala and Mayne, in 1979 [61]. Afimiwala used contours of constraints to directly identify the feasible region of two variable problems. Users were able to click on feasible areas (represented as a scatterplot) to generate samples at the clicked location.

To accomplish VDS with additional variables, Winer and Bleobaum employ *Graph Morphing*, which plots 2D or 3D contours of a few chosen variables, and leaves the remaining variables as interactive sliders. As the sliders are adjusted, the graph is “morphed". Essentially, the variables that are not directly plotted influence the shape of the graph as hidden variables. VDS was also used in conjunction with variable importance ranking to find a suitable initial starting point for optimization (using the Automated Design Synthesis method) [22]. It was found that this procedure reduced the number of function evaluations required by approximately 50% on average, in comparison to starting with a random point.

2.3.3. Automatic Trade Space Visualization

In contrast to VDS, Balling’s *Design by Shopping Paradigm* presents a set of designs, from a large sample, allowing users to subsequently formulate their preferences by comparing trade-offs [12]. Balling’s approach is especially pertinent to multi-objective problems where objectives conflict with one another. Sequentially, this is the inverse order of operations to traditional optimization or VDS, which both begin with preferences or objectives and generate samples accordingly. Instead, by comparing the trade-offs from an initial sample set, users may decide which objectives are truly important or necessary, as shown in Figure 13. Penn State’s *Applied Research Laboratory Trade Space Visualizer (ATSV)*, developed by Stump et al., adds interaction techniques for design by shopping with many variables or objectives [21].

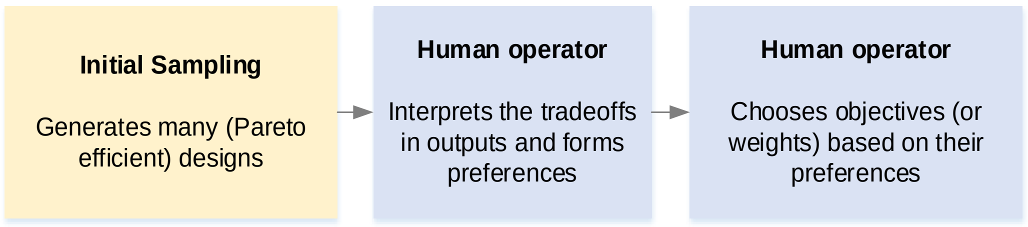

Initial Sampling
Generates many (Pareto efficient) designs
Human operator
Interprets the tradeoffs in outputs and forms preferences
Human operator
Chooses objectives (or weights) based on their preferences

Figure 13 Design by shopping approach to problem formulation

Originally, ATSV relied on heavy initial sampling for exploration. Users would search through thousands of simulated tests in a scatterplot matrix to find a subspace with desired performance for further sampling. However, this is inefficient for computationally intensive problems, as evaluating a large set of initial designs may be prohibitively slow and wasteful. More recently, a link to the simulation or black-box model was added to generate candidate designs on-the-fly using various sampling strategies, starting from a small initial population [62,63]. For examples, an “attractor” could be added in the performance space, which biases sampling towards the objectives near the attractor. This effectively combines ATSV’s multivariate visualization capabilities, design by shopping, and VDS.

Data mining techniques were also prototyped by the same group recently, in 2012, to help support analysis of larger engineering design data sets [64]. The LIVE tool integrates statistical clustering and classification, allowing users to select interesting areas in the engineering data. Users can then choose to create additional samples in the selected regions. This is conceptually similar to the work presented in this thesis; however the region selection, data mining, and visualization methods differ.

2.3.4. Physical Programming and Physical Programming Based Visualization

Messac’s *Physical Programming* approach to multidisciplinary design optimization (MDO) works towards flexible problem formulation [10,65]. In MDO, there are often many competing objectives to be simultaneously minimized. A common solution is to aggregate objectives into a single function using weights, or to prioritize and optimize the system sequentially. In physical programming, instead of assigning weights or priorities, a designer specifies range thresholds for each objective:

unacceptable, undesirable, tolerable, desirable or highly desirable. The method then constructs a *class function*, based on the problem type, which maps the actual objective function value to these qualitative ranges. This changes the dynamic of convergence for multi-objective optimization. For example, moving from unacceptable to tolerable in one objective is more significant than moving from desirable to highly desirable in another. Similarly, moving within the highly desirable range is of little significance. Thus, the preferences are adjusted towards convergence in the desirable ranges for all objectives and away from the unacceptable for any objective.

Physical programming was extended in 2000 to help visualize optimization progression. The method is named *Physical Programing Based Visualization (PPV)* [66]. In PPV, values of design metrics are plotted iteratively as lines, bar graphs and radial charts. A PPV line chart is shown in Figure 14. PPV differs from ATSV or LIVE in that the visualization is focused on the movement of the optimum solution towards the highly desirable range, as opposed to trying to visualize the design space.

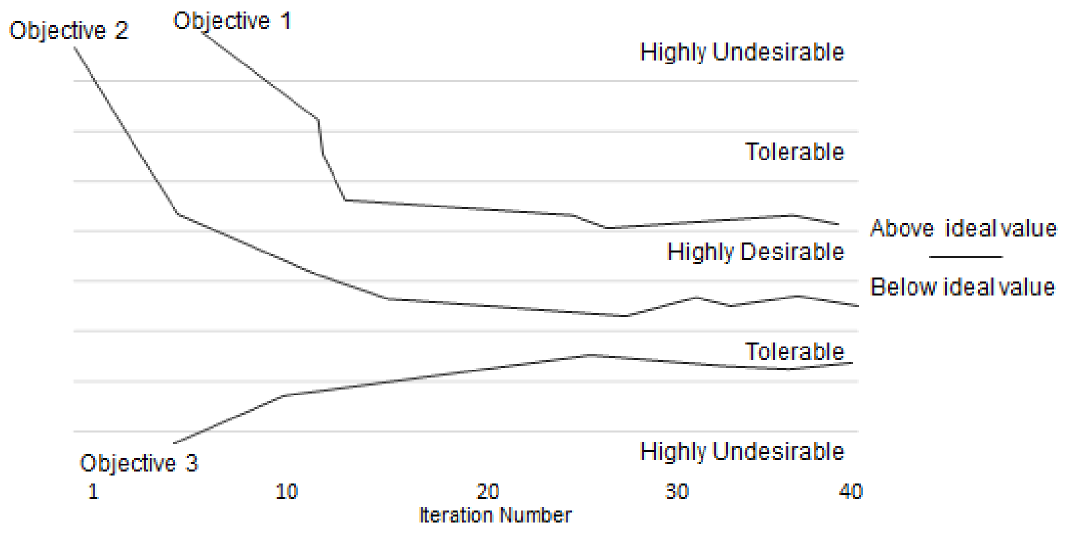


Figure 14 Physical programming history visualization (adapted from [66])

2.3.5. Pareto Frontier Visualization Methods

Many visualization techniques have been published to present Pareto optimal multi-objective results. Pareto optimal designs are those for which no other design is better for all objectives. In other words, to improve any objective value of a Pareto optimal design, a compromise must be made in the remaining objectives. Points in the

Pareto set are shown in black in Figure 15. Representing the Pareto set (a.k.a. the Pareto Frontier) requires as many dimensions as there are objectives. Therefore, for problems with many objectives, multivariate visualization is required to show the objective (performance) space.

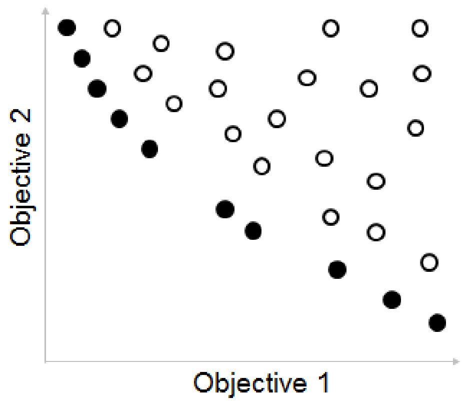


Figure 15 Pareto set of a 2D performance space (shown in black)

Cloud visualization, by Eddy and Lewis [67] in 2002, splits optimization data into two linked (3D) scatterplots: the design space (where variables are used as the axes) and the performance space (where objectives are used as the axes). Users can highlight points in either graph and see the result in the other. Additionally, users can choose a location in the performance space, and the corresponding design point(s) will be highlighted or generated using a genetic algorithm (if the point is nonexistent but possible), as shown in Figure 16. A similar approach to cloud visualization for engineering design was also introduced, by Spence, for analog circuit design with linked histograms [68,69].

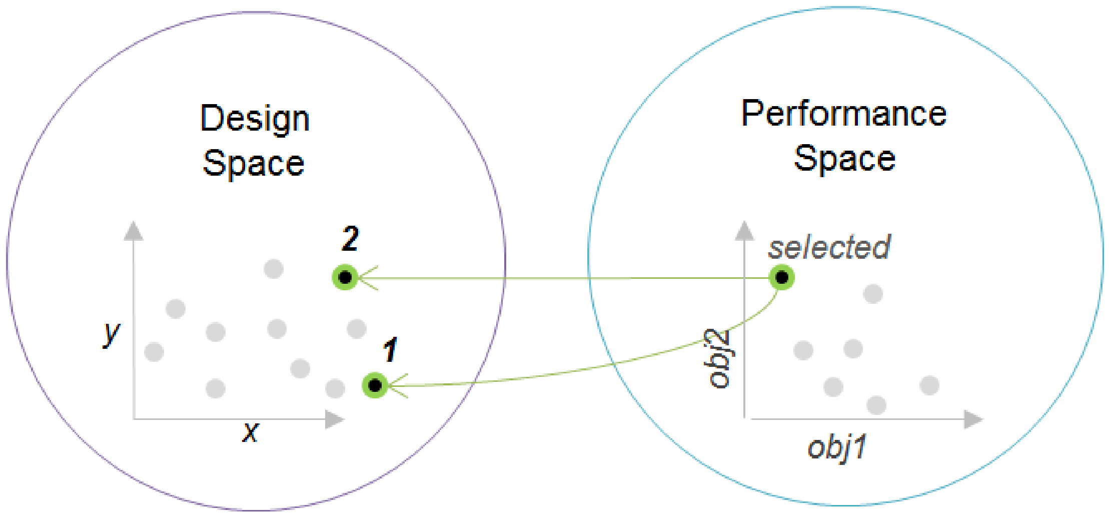


Figure 16 Cloud visualization (with 2D clouds) – adapted from [67]

2.3.6. Parallel Coordinates in Support of Interactive Optimization

Parallel coordinates was isolated as a helpful method for interactive optimization in Froschauer's 2009 thesis work [57]. It was also mentioned in Jones' textbook [34]. Froschauer added optimization techniques to an existing data analysis software (Bulk Analyzer), with a focus on using parallel coordinates to assist with optimization. Froschauer provides an excellent overview of visualization (especially parallel coordinates), interaction, optimization, and their integration. A heavy focus is visualizing multi-objective tradeoffs. For example, a method is presented to construct Pareto sets with different levels of dominance, and highlight them in the Bulk Analyzer. Another method is presented to highlight points based on adjustable "assessment ranges" which classify points as desirable, satisfactory, or undesirable (as in Messac's Physical Programming method). Finally, as in this thesis, there is a method to estimate data that is not present in the set of samples.

Although Froschauer's work is conceptually very similar to the work done here, there are some key differences. Most notably, Froschauer's optimization process is on a given dataset, which was evaluated and simulated a priori. In other words, data is not generated on-the-fly from a cost function or simulation. Instead, the optimum is found according to a defined multi-objective problem, using past data that has been loaded into the bulk-analyzer. In contrast, in this thesis, visualization is used to find areas for further sampling by an integrated optimization method. Furthermore, this thesis presents methods to track optimization progress as new samples are added; navigation using 2D projections; integrated clustering and dimensionality reduction to isolate designs with similar performance; and a method to identify relationships among constraints using data mining.

2.4. Summary

This chapter introduced InfoVis, which is the representation of abstract data that is not tied to a spatial model. Two techniques to display multivariate data were introduced: the scatterplot matrix and parallel coordinates. Aside from visualization, dimensionality reduction is discussed to reduce the number of independent variables while preserving statistical properties. Next, a review of related work is given. Topics covered include interactive evolutionary computing, visual design steering, design by shopping, physical programming based visualization, and cloud visualization. In the following methodology chapter, theoretical details of optimization and visualization are elaborated and defined mathematically.

Chapter 3. Methodology

This chapter presents methodologies related to interactive optimization in mathematical detail. First, an introduction is given to numerical optimization, followed by an introduction to spaces and regions. Next, clustering methods are introduced to split large regions into smaller ones. Clustering is followed by interactive techniques and two-dimensional selection (including PCA). The goal of this chapter is to provide theoretical background for the optimization visualization framework, which follows in Chapter 5.

3.1. Numerical Optimization

Numerical optimization is an algorithmic process that searches through potential designs in order to find the one(s) which provides the lowest cost or highest performance. Generally, there are three components to an optimum design problem. First, there are *design variables*. These are the variables which can be tuned to improve performance. Second, there are *objectives*. Objectives are functions which return a measure of performance (e.g. strength-to-weight ratio, system cost etc.). Third, there are *constraints*. Constraints are limitations placed on the system. These may represent physical, space, or cost limitations which render the system infeasible. Formally, optimization problems are posed in standard form as follows in Eq. (1):

$$\begin{gathered}\min \boldsymbol{f}\,(\boldsymbol{x}) \\ s.t\,\{\boldsymbol{g}(\boldsymbol{x}) \leq 0, \boldsymbol{h}(\boldsymbol{x}) = 0\}\end{gathered} \tag{1}$$

In the equation above $\boldsymbol{x} \in \boldsymbol{X} \subseteq \mathbb{R}^n$ is a set of values for each of n number of design variables; $\boldsymbol{f}(\boldsymbol{x})$ is a set of objective functions which models system costs; $\boldsymbol{g}(\boldsymbol{x})$ is a set of inequality constraints; and $\boldsymbol{h}(\boldsymbol{x})$ is a set of equality constraints, which model limitations. Note, if the goal is maximization (e.g. maximize performance), $\boldsymbol{f}(\boldsymbol{x})$ can be easily negated to make the problem a minimization problem ($\min -\boldsymbol{f}\,(\boldsymbol{x})$).

Equality constraints are cumbersome to deal with. They require the constraint function to have an exact value, which is probabilistically impossible by random sampling for continuous functions. Strategies to deal with equalities include converting equalities to inequalities by adding slack variables; penalizing the performance of a design based on its non-zero constraint value; or simply not supporting equality constraints. In this thesis, equality constraints are not supported.

In single objective optimization, the focus of this work, each design $\boldsymbol{x}$ returns a scalar value f. However, this doesn't necessarily imply that only one output of a simulation must be considered. It is common to aggregate multiple objectives into an overall performance measure by summing each output with a given set of weights. In Eq. (2), l is the number of objectives to be aggregated, and $w_j \in \boldsymbol{w}$ is a weight that scales the effect of $f_j(\boldsymbol{x})$ (an individual objective) on the aggregated objective. Furthermore, although each design corresponds to a single value of performance, there may be multiple designs that yield the same performance.

$$f(\boldsymbol{x}) = \sum_{j=1}^{l} w_j * f_j\,(\boldsymbol{x}) \tag{2}$$

It's clear from the equations above that formulating a problem for optimization requires assumptions and domain knowledge. For example, an equation or model is required to map each design point $\boldsymbol{x}$ to f. Additional equations are also required for the constraints $\boldsymbol{g}(\boldsymbol{x})$. Finally, $\boldsymbol{x}$ is bounded to give the algorithm a finite search domain. Due to these requirements, the problem formulation stage (defining the equations in (1) or choosing weights in (2)), is error-prone. In fact, Arora estimates that problem formulation accounts for 50% of the effort required to optimize a problem [58].

3.1.1. Optimization Space Terminology and Notation

To clarify some of the terms for discussing optimization, three spaces may be defined, as in Figure 17. These terms are common in optimization literature [67].

The ***design space*** $\mathcal{D}$ is the domain of all possible designs that may be generated throughout the entire optimization process. The tested designs are contained in the sample matrix $\boldsymbol{X}$, with m number of rows corresponding to individual designs (a.k.a. samples or points), and n number of columns corresponding to design variables (a.k.a. attributes or dimensions). Individual designs are denoted by $\boldsymbol{x}$ in general, or $\boldsymbol{x}^i$ when discussing a specific design. The design space is bounded by a vector of lower bounds ($\boldsymbol{D}_L$) and a vector of upper bounds ($\boldsymbol{D}_U$): $\boldsymbol{x} \in [\boldsymbol{D}_L, \boldsymbol{D}_U]$. These bounds are given by the user in the problem definition as a wide scoping region that may be worth exploring.

The ***search space*** $\mathcal{S} \subseteq \mathcal{D}$ covers the boxed area that the optimization algorithm is currently searching. Variables are bounded by a vector of lower bounds ($\boldsymbol{S}_L$) and a vector of upper bounds ($\boldsymbol{S}_U$)**.** In the search space, new points are sampled by an optimization method or by random sampling. The search space is intended to be flexible throughout the optimization process, as users chose to explore new regions.

The ***performance space*** $\mathcal{F}$ covers the range of function values which are generated from $f(\boldsymbol{x})$. The bounds of the performance space are generally unknown (especially prior to optimization). Indeed, it is the goal of optimization to find the lower bound of $\mathcal{F}$ (and corresponding design), representing minimum cost. A vector $\boldsymbol{f} \subseteq \mathbb{R}$, contains all of the scalar function values f^i, tested for each design $\boldsymbol{x}^i$.

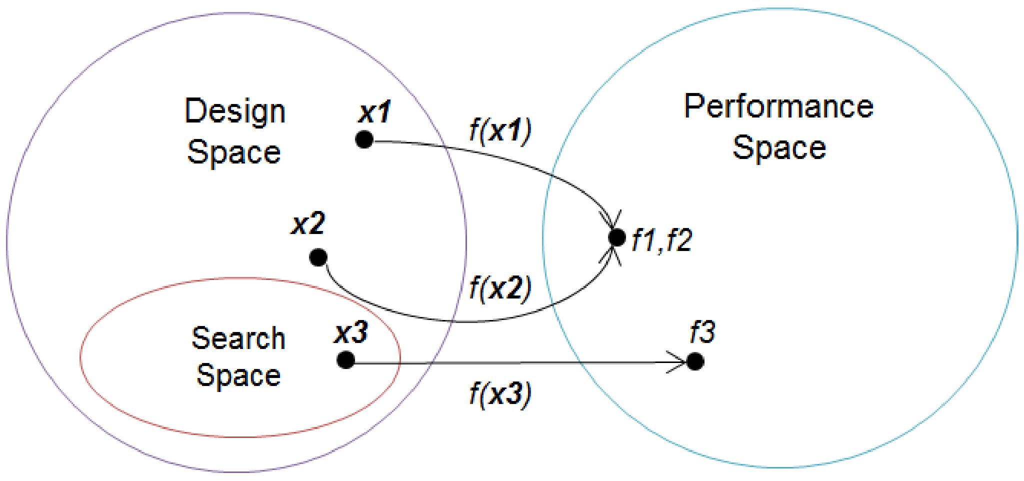


Figure 17 Design Space, Search Space and Performance Space

3.2. Regions and Clustering

In order to select areas for further investigation, ***regions*** may be defined in either the performance or design spaces. Selecting a region enables users to visually highlight points with similar properties. Regions may also grow or shrink as the optimization progresses, which gives a visual indication of convergence.

To illustrate the three types of regions in this work, a two-variable benchmark optimization problem, the *six-hump camel back* problem (SC) is introduced. It's named the six-hump camel problem because there are six local minima in total, including two global minima. This problem will be used as an example throughout this chapter and is defined in Eq. (3), with bounds $\boldsymbol{D}_L = [-2, -2]$ and $\boldsymbol{D}_U = [2\ ,2]$. A contour plot of the SC problem is given in Figure 18.

$$\min f\ (\boldsymbol{x}) = 4 * (x_1)^2 - \frac{21}{10} * (x_1)^4 + \frac{1}{3} * (x1)^6 + (x_1) * (x_2) - 4 * (x_2)^2 + 4 * (x_2)^4 \qquad (3)$$

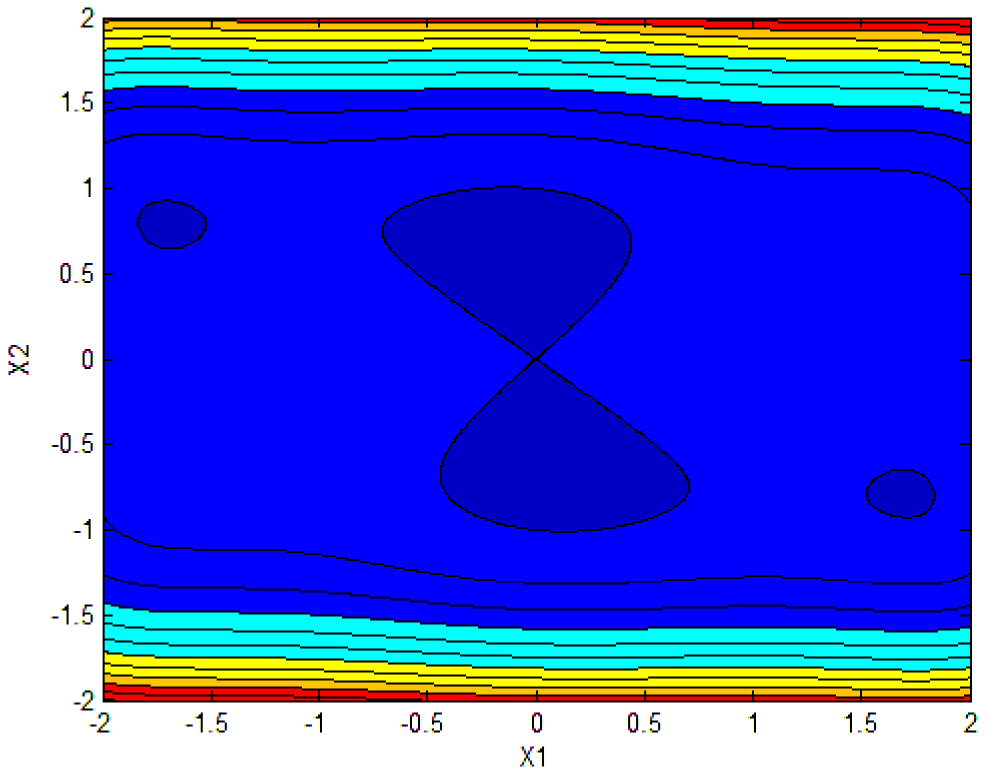


Figure 18 Contour plot of Six-Hump camel problem

3.2.1. Percentile Regions

The first type of region that may be defined is a ***percentile region*** ($\boldsymbol{K}$)**.** Percentile regions group points by relative performance. For example, a percentile region may correspond to the query: "Show me the top 15% of designs". Points with function values between percentiles k_l and k_u are selected. Formally, this can be written as Eq. (4):

$$\boldsymbol{K} = \{\boldsymbol{x} \in \mathbf{X} \mid k_l \leq k(f(\boldsymbol{x})) \leq k_u\} \tag{4}$$

Where $k(f(\boldsymbol{x}))$ is the relative ranking of each function value (as expressed by a percentile). For illustration, the top 15% of 500 random points, tested for the SC problem, are shown in black in Figure 20. The same region is also shown in parallel coordinates in Figure 21. The cyan area shows the boxed space encompassing these points. In Figure 20, the bounding function values are also shown (0.215 to -1.02).

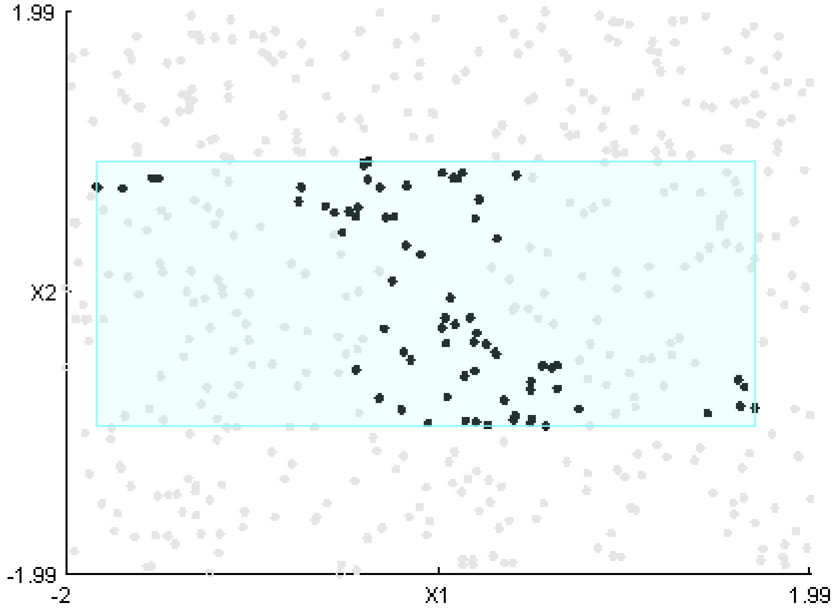


Figure 19 Percentile Region - Top 15% of the SC Problem (Scatterplot)

Percentile regions are important for two reasons. First, the limits of performance are generally unknown prior to optimization. Second, the concept of a "good performance" may also change as new samples are tested. Therefore, by using relative performance, a meaningful region can be defined without specifying numeric values.

This allows user to track top performing points throughout the optimization process, and observe as the algorithm converges.

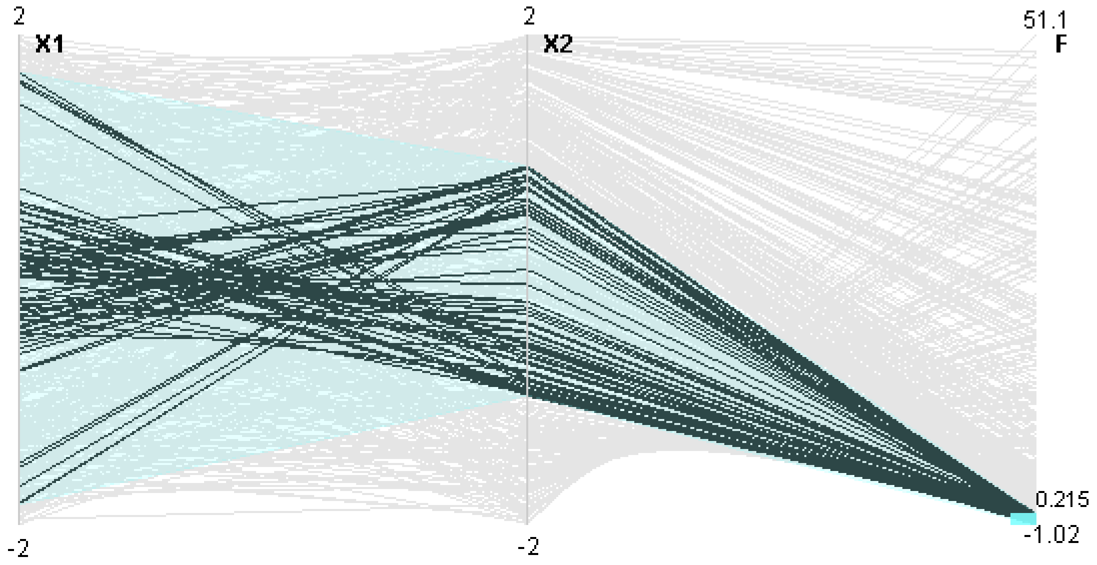


Figure 20 Percentile Region - Top 15% of the SC Problem (PCP)

3.2.2. Performance Regions

If a designer knows which range of performance they wish to explore (e.g. designs that outperform a threshold), they may limit the performance space to exact numeric values. A range which restricts designs by absolute performance is simply called a ***performance region*** ($\boldsymbol{P}$). A performance region is formalized by Eq. (5):

$$\boldsymbol{P} = \{\boldsymbol{x} \in \mathbf{X} \mid f_l \leq f(\boldsymbol{x}) \leq f_u\} \tag{5}$$

Where f_l and f_u are limits given to the performance space. For instance, the performance region defined by $-1.02 \leq f(\boldsymbol{x}) \leq -0.5$, is shown in Figure 21. Again, black points are in the region, while the remaining points are not. The limits of design variables corresponding to the points are shown in orange.

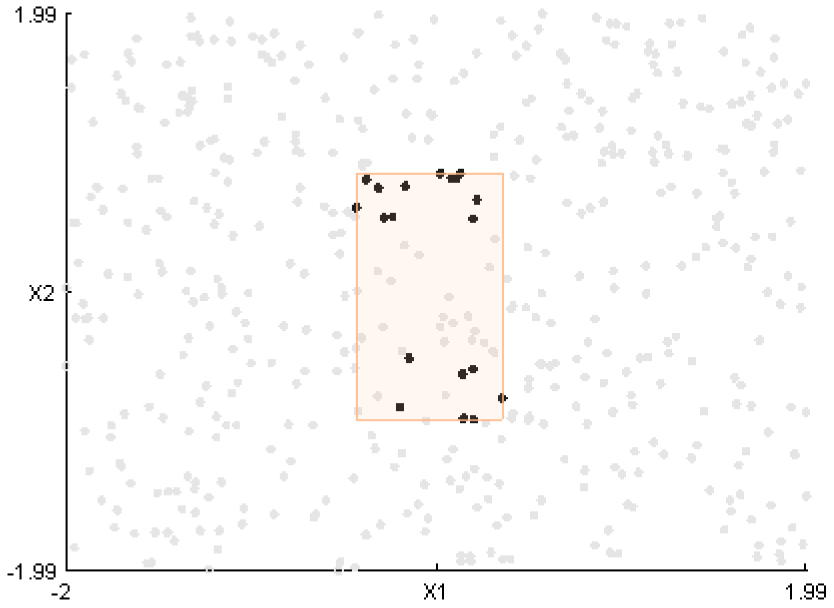


Figure 21 Performance Region - Objective between -1.02 and -0.5 of SC Problem

3.2.3. Design Regions

Design regions ($\boldsymbol{D}$) are defined in the design space, regardless of performance. For example, a design region may be as follows: "Show me all of the designs where $-1 \leq X_1 \leq 1$ and $-1 \leq X_2 \leq 1$". Design regions are formalized in Eq. (6):

$$\boldsymbol{D} = \{\boldsymbol{x} \in \mathbf{X} \,|\, \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u\} \tag{6}$$

In Eq. (6), $\boldsymbol{x}_l$ is a vector of lower limits for each variable, and $\boldsymbol{x}_u$ is a vector of upper limits for each variable. The space defined by $-1 \leq X_1 \leq 1$ and $-1 \leq X_2 \leq 1$ is shown in Figure 22 and Figure 23. Notice that all of the points in the purple area are highlighted in the X2 vs X1 graph, as the limitation is placed on $\boldsymbol{x}$ instead of f.

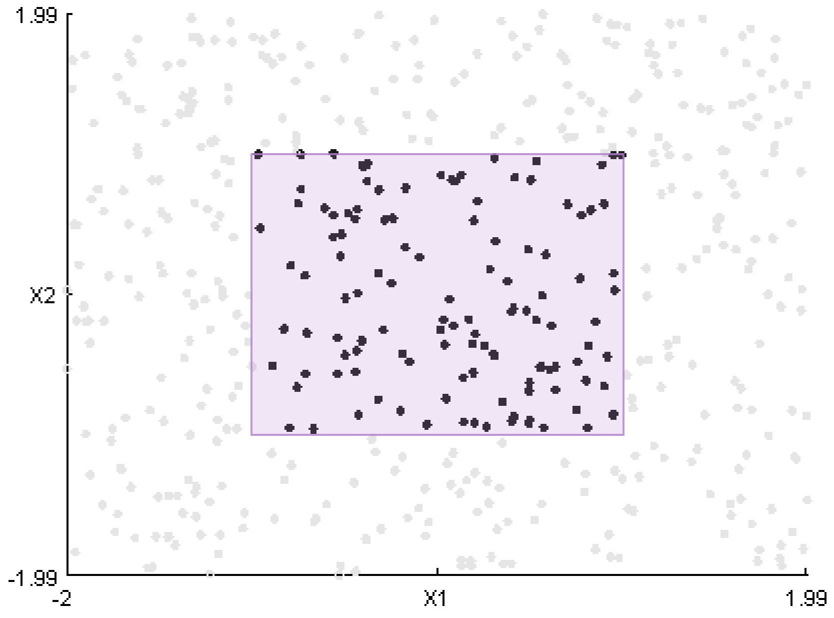


Figure 22 Design Region - x1: [-1, 1] and x2: [-1, 1] (Scatterplot)

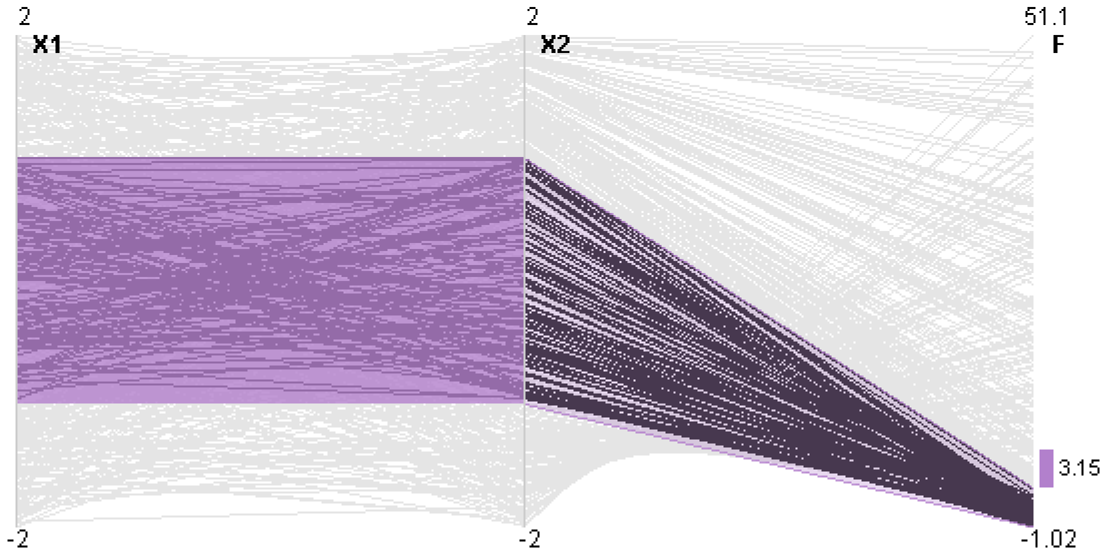


Figure 23 Design Region - x1: [-1, 1] and x2: [-1, 1] (PCP)

Design regions are simply bounds on the variable space. Therefore, they may be used as the search space bounds for further optimization. Additionally, localized models may be fitted and tested within these bounds.

3.2.4. Clustering

Interactive methods to define and adjust regions are discussed in Chapter 5. As an alternative, clustering automatically splits data to find distinct groups of similar points. In engineering design, clusters within a percentile or performance region means there are distinct design alternatives that yield similar performance.

Consider the percentile region shown in Figure 19. There are four areas which make up the top 15 percentile of points as shown in Figure 24. Each group represents a distinct set of designs that are in the top 15 percentile.

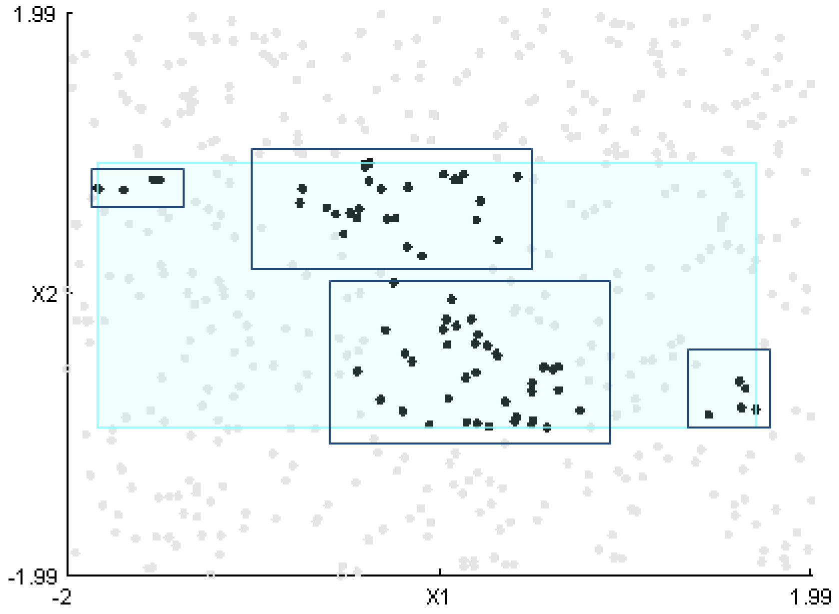


Figure 24 A possible clustering within the top 15% of points of the SC problem

Of course, clustering is subjective. For example, one could easily argue that the middle region in Figure 24 is a single cluster. Therefore, clustering should be made flexible using visual feedback. As a very brief introduction to cluster analysis, background is given for two popular methods: *k-means* and *single-link* clustering.

K-Means clustering

K-means clustering is a center-based clustering method that splits points into *k* number of groups, where *k* is typically provided by a user. The term *center-based* clustering means that each point is related to the cluster's center (a centroid). The process is summarized in Figure 25 and illustrated in Figure 26, where the colors indicate the clusters, and triangles indicate centroids. The initial centroids were chosen at random for this example. In practice, centroids may be chosen to be furthest from one another.

After initialization, each point is assigned to the *nearest* centroid. Nearness, or similarity, can be defined in many ways. In fact, there are dozens of definitions of similarity available (e.g. L-norms, Jaccard, Cosine, etc.) [70]. In this work, the squared Euclidean distance is employed. Data is also standardized in each dimension (between [0,1]) to ensure each design variable has equal weight in the distance computation. Squared Euclidean distance (a.k.a. L_2^2) is one of the most intuitive notions of distance for purely quantitative data. The L_2^2 distance is given by Eq. (7):

$$d(\boldsymbol{x}^i, \boldsymbol{x}^c) = \sum_{j=1}^{n} (\boldsymbol{x}_j^i - \boldsymbol{x}_j^c)^2 \tag{7}$$

In words, Eq. (7) sums the squared differences between the point $\boldsymbol{x}^i$ and centroid $\boldsymbol{x}^c$, in each of *n* dimensions. Once all points are assigned to their nearest centroid, the centroids are redefined as the means of their assigned points. Next, the points are re-assigned to the nearest of the new centroids. This iterative process continues until all centroids no longer change. Note that the choice of initial centroids will affect the result of the k-means clustering. Often, k-means is repeated with different centroids choices, and the result which minimizes the overall distance from points to their center is selected.

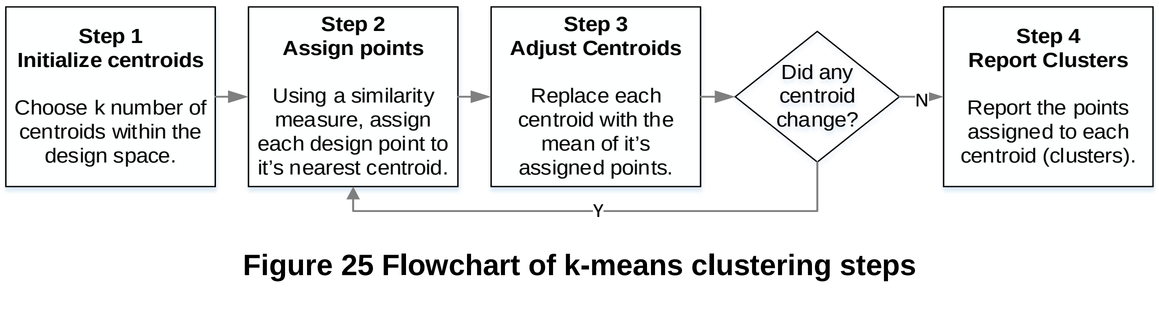

Step 1
Initialize centroids
Choose k number of centroids within the design space.
Step 2
Assign points
Using a similarity measure, assign each design point to it's nearest centroid.
Step 3
Adjust Centroids
Replace each centroid with the mean of it's assigned points.
Did any centroid change?
N
Y
Step 4
Report Clusters
Report the points assigned to each centroid (clusters).

Figure 25 Flowchart of k-means clustering steps

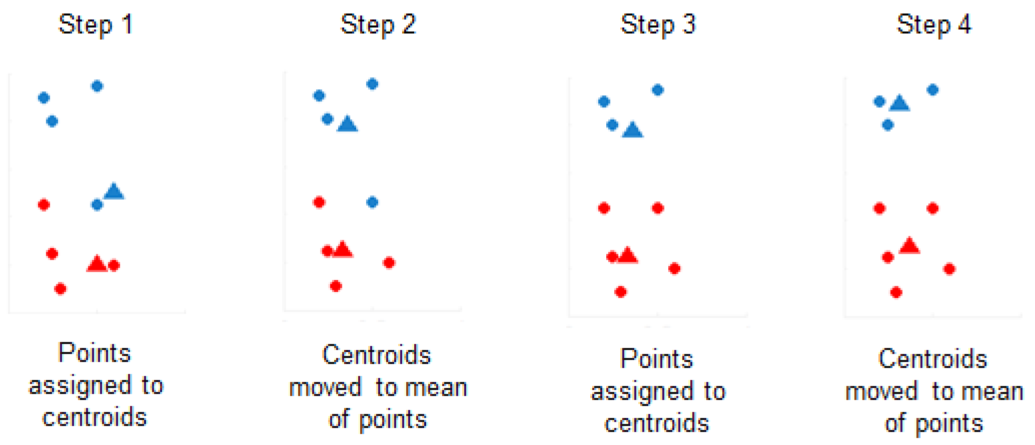

Step 1
Step 2
Step 3
Step 4
Points assigned to centroids
Centroids moved to mean of points
Points assigned to centroids
Centroids moved to mean of points

Figure 26 Visual example of k-means clustering steps

Agglomerative (single-link) clustering

Agglomerative clustering is a hierarchical approach that combines points from the ground up to build clusters. This approach is fundamentally different from center-based methods. The idea is to build small clusters, and progressively combine the nearest ones, until the distance between clusters is beyond a cut-off or all clusters have been merged. Therefore, the parameters to agglomerative clustering are a cut-off distance (ϵ) and a method to determine nearness between clusters (e.g. single-link).

There are many ways to define nearness between two clusters. One way, following the thinking of k-means, is to consider the distance between the means of each cluster. This is called the centroid proximity method. The *single-link* variation defines proximity as the distance between the two closest points from each cluster, as shown in Figure 27. The mathematical definition of single-link proximity p is given in Eq. (8).

$$p(\boldsymbol{c_a}, \boldsymbol{c_b}) = \min[\, d\left(\boldsymbol{c}_a^{i_a}, \boldsymbol{c}_b^{i_b}\right) \,] \tag{8}$$

$$1 \leq i_a \leq \mathrm{m_a} \;\; 1 \leq i_b \leq \mathrm{m_b}$$

Where $\boldsymbol{c_a}$ and $\boldsymbol{c_b}$ are two non-empty clusters, containing $\mathrm{m_a}$ and $\mathrm{m_b}$ number of points (respectively), and *d* is a distance function (e.g. L_2^2). Individual points in the clusters are denoted by $\boldsymbol{c}_a^{i_a}$ and $\boldsymbol{c}_b^{i_b}$ respectively.

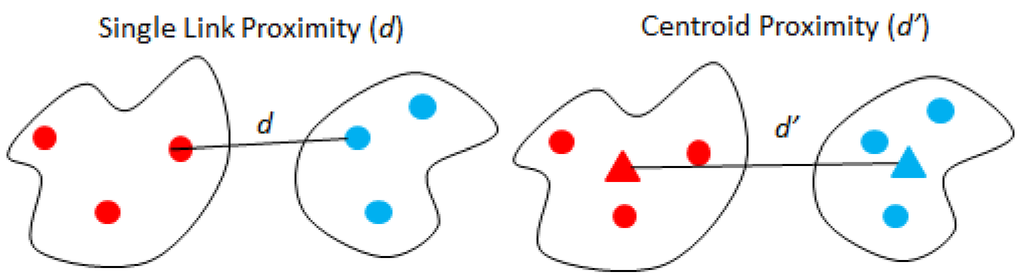


Figure 27 Single-link proximity versus centroid proximity (adapted from [70])

As an example, consider the data which was clustered using k-means above. This data is clustered via single-link clustering in Figure 28. Initially, clusters consist of single points. Therefore, their proximity is simply the distance between one another. In the first step, the two nearest points are merged to form a two-point cluster. Step 2 merges the next closest clusters. This continues until all clusters are merged, or the minimum cluster proximity exceeds a cut-off distance.

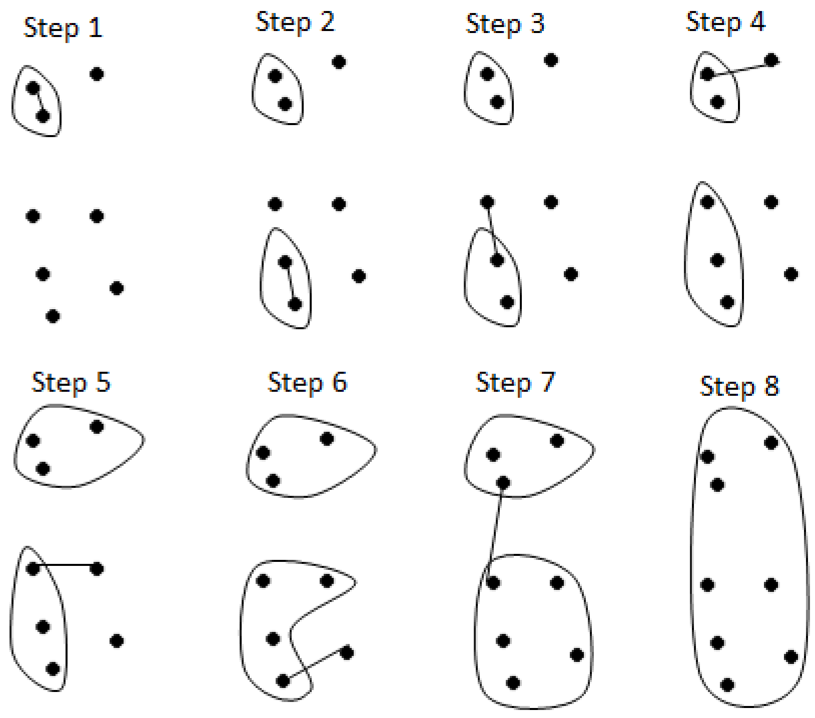


Figure 28 Single-link clustering example

Notice that the minimum proximity between clusters in each step (as shown by a line) increases monotonically for single-link clustering. Thus, the *cut-off distance* ϵ

specifies a sufficient distance between groups for them to be considered truly distinct. In practice, however, it is difficult to specify a meaningful cut-off distance .

Although, cluster quality can be evaluated using quantitative metrics (e.g. average distance of points to cluster center) there is no theoretical best value for cut-off. Choosing ϵ depends on the user's preferences (e.g. what is an acceptable distance for the clusters to be considered unique), the application, and the data. Therefore, users may adjust ϵ using a slider in the visualization framework, to automatically re-cluster in a flexible manner. Clustering via k-means or single-link was found to be sufficiently quick (~0.2 seconds) for datasets of 10,000 points on a typical PC using MATLAB. However, redrawing the graphs increases the overall response time to between 2-3 seconds for the same datasets. This delay is non-ideal for smooth interaction (<1 second response time) [71]. Nonetheless, clustering can be recomputed relatively quickly, and with visualization, *k* or ϵ can be selected on-the-fly.

SC Clustering Example

K-means and single-link were applied to the top 15% region of the SC problem as an example. The results are shown in Figure 29 for k=4 with k-means, and for $\epsilon = 0.07$ with single-link. Notice, the clustering from k-means does not match the expected "natural" clustering (in Figure 24). This is due to the random selection of the initial centroids. In this case, single-link's result is more in line with the proposed *natural* clustering, but required careful tuning to $\epsilon = 0.07$.Once split into clusters, the top performing region may further split to *design* regions for further exploration as distinct areas as explained in Chapter 5.

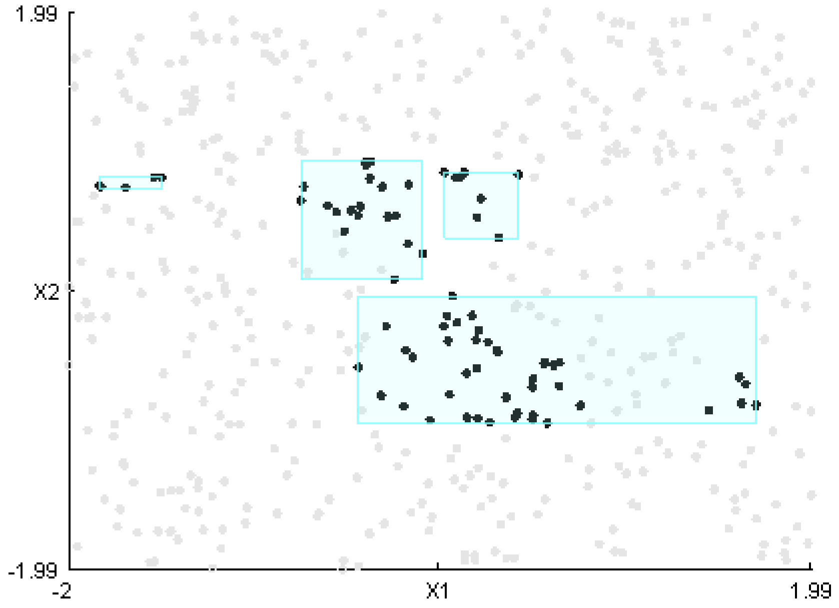


Figure 29 SC Top performing region (top 15%) via k-means clustering (k=4)

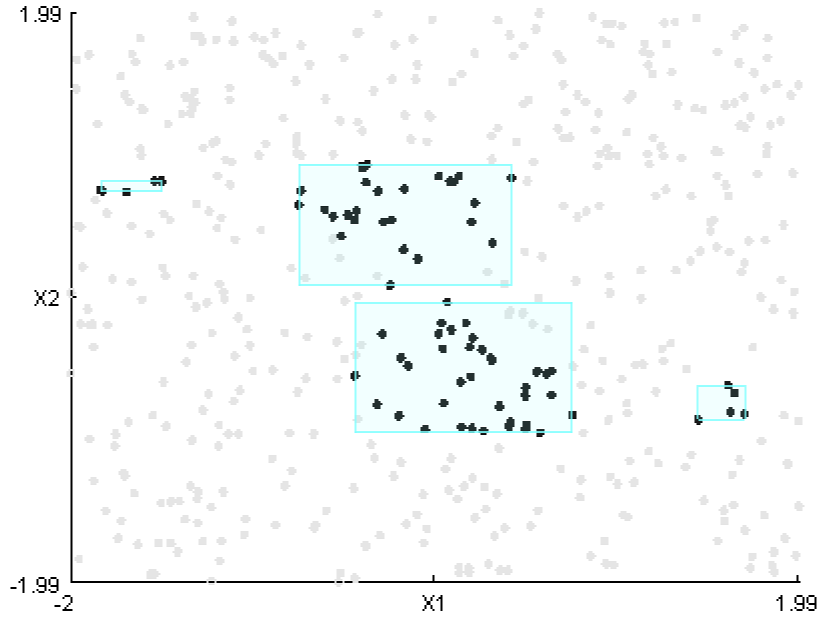


Figure 30 SC Top performing region split via single-link clustering (ϵ =0.07)

3.3. Interaction and Navigation

A critical component of visual data analysis is interaction. Interaction allows users to browse through sets of data, seek for specific values, or simply see what data is available.

Each time the user performs an interaction, they follow a cycle of actions as modeled in Figure 31. This model, Norman's action cycle, is common in human-computer interaction literature [17,72]. The process begins with a specific *goal*, such as determining if there are distinct design groups with high performance for an optimization problem. Next, the user decides on a *specific intent*. For the previous goal, this may include separating regions by cluster analysis in the top 15% of points. The user then plans *actions*. For example, to perform cluster analysis, a percentile region may be defined and then further split by single-link clustering. The visualization is refreshed once these actions are carried out. At this point, the user perceives, interprets, and evaluates the resulting visualization. The result often leads to a new goal (e.g. adding samples to the distinct regions) which closes the interaction loop.

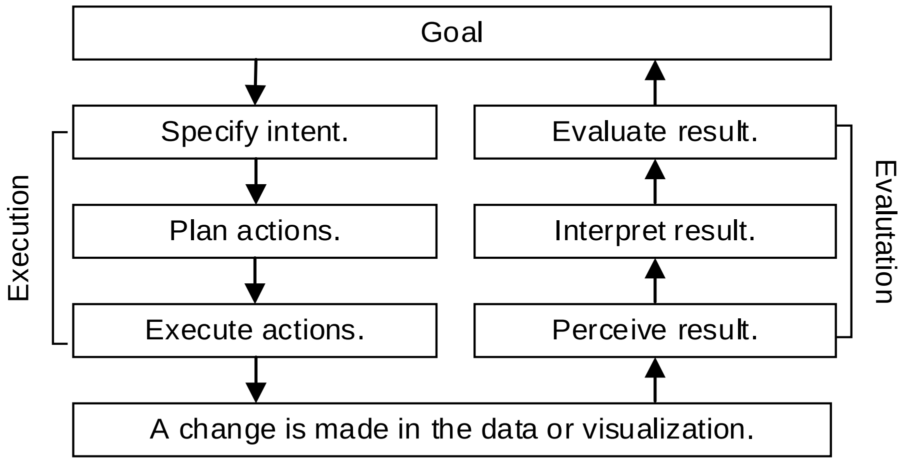


Figure 31 Norman's action cycle of interaction (modified from [17])

The value of interaction is more apparent on datasets with more than three attributes. Therefore, a four variable pressure vessel design (PV) optimization problem is used as an example. This problem is from Globally Optimal Design by Wilde [73], and has been used as a benchmark problem in engineering optimization literature [74,75]. The problem has four variables: tank radius (R), tank length (L), shell thickness (T_s) and head thickness (T_h). The objective is to minimize the aggregated system cost ($), including materials and manufacturing. There are three constraints in the original problem, based on ASME pressure vessel standards, and one objective cost function. Although the problem is often posed with discrete thickness values, it is defined here for continuous values as in reference [74] with Eqs. (9) to (12).

$$\min f\ (\boldsymbol{x}) = 0.06224\mathrm{T_s}RL + 1.7781T_hR^2 + 3.1661T_sL + 19.84T_s^2R \tag{9}$$

$$g1 = 0.0193R - T_s \leq 0 \tag{10}$$

$$g2 = 0.00954R - T_h \leq 0 \tag{11}$$

$$g3 = 1296000 - \pi R^2L - \left(\frac{4}{3}\right)\pi R^3 \leq 0 \tag{12}$$

The variable bounds are as follows (each variable is in inches):

$$R \in [25{,}150], T_s \in [1{,}1.375], L \in [25{,}240], T_h \in [0.625{,}1]$$

A diagram of the vessel design parameters is shown in Figure 32.

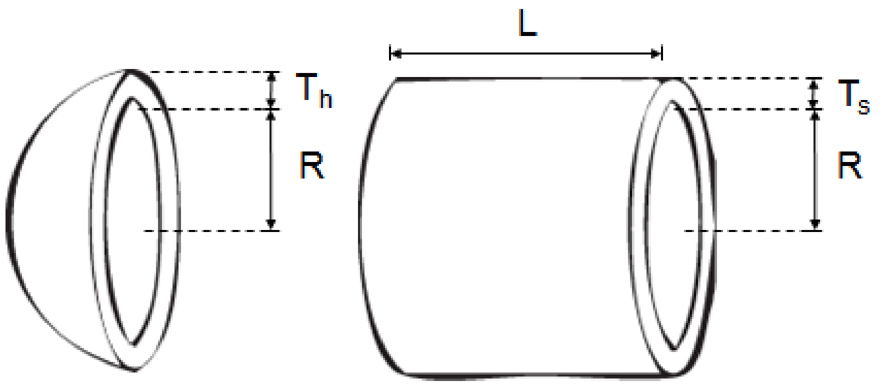


Figure 32 Diagram of the PV optimization problem variables

3.3.1. Navigation, Brushing and Overviews

Navigation

Navigation is the term for moving from one information space to another. This includes choosing which data is shown or hidden, and in which level of detail. In traditional design optimization the specific goal is to navigate through the design space to find the optimum. In this case, navigation is automatically executed by an optimization algorithm which biases the generation of new designs towards minimization of the objective function. This kind of target specific navigation is called *pursuit* [17], and can be partially or fully automated. A contrasting type of navigation is *exploration*, where the user manually navigates through the design space to improve their understanding of the dataset.

Brushing

Brushing, related to navigation, is the selection of data in one view, which becomes nearly instantly highlighted in other views. For example, consider Figure 33 and Figure 34 which show the top 50 designs out of 500 for the PV problem (all other designs are suppressed). In Figure 33 the shell thickness (T_s) is plotted against the head thickness (T_h). Although the 2D projection does not show all of the attributes of the data, it can be used to *brush* points in T_s vs. T_h (thickness variables).

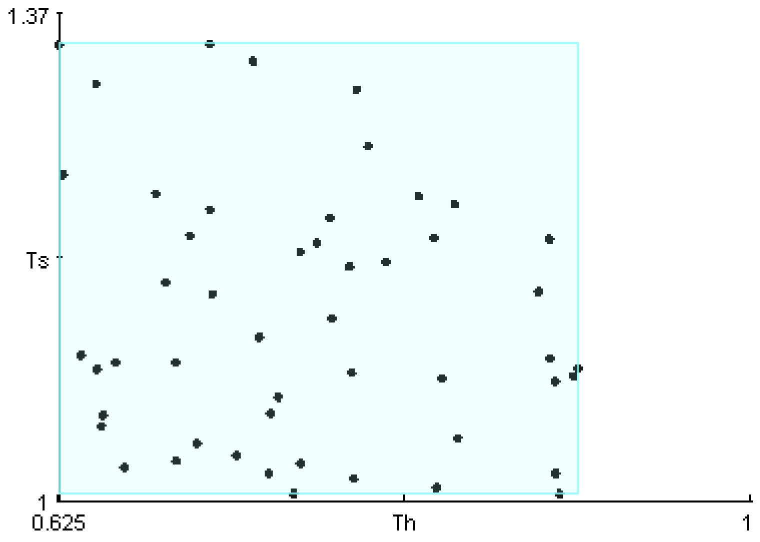


Figure 33 2D projection of PV problem (top 50 designs shown)

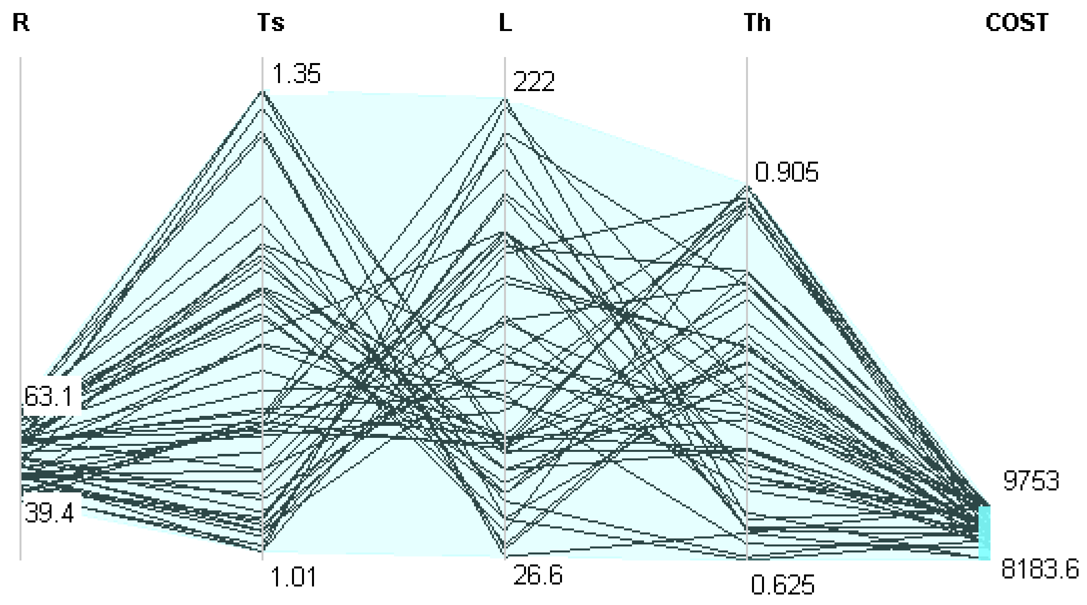


Figure 34 Parallel coordinates plot of PV problem (top 50 designs shown)

As an example, a selection is made in Figure 35 for thin low-cost vessels. The corresponding points are immediately shown in an adjacent plot (Figure 36). By brushing other 2D projections, the designer may select other attributes as well. Furthermore, picking points directly in 2D projections is less error-prone than in parallel coordinates where there is overdraw. Brushing is a very popular interaction technique, and was also employed by related optimization visualization work [21,57].

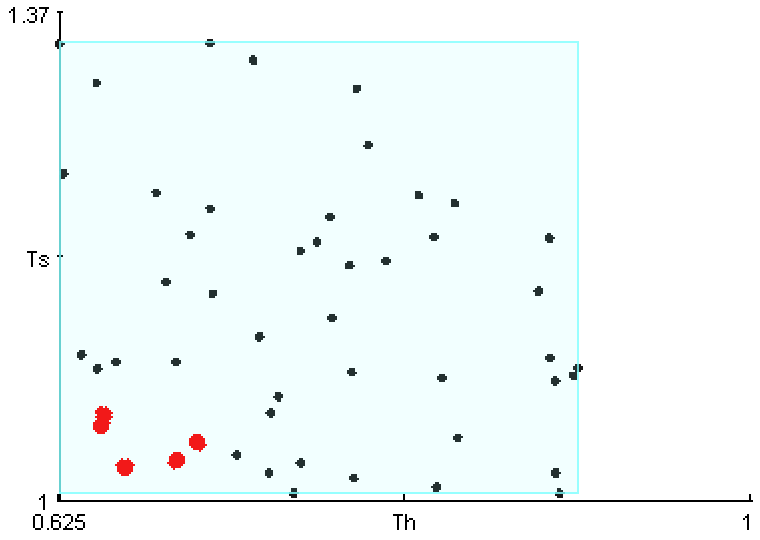


Figure 35 2D projection of PV problem (thin designs selected)

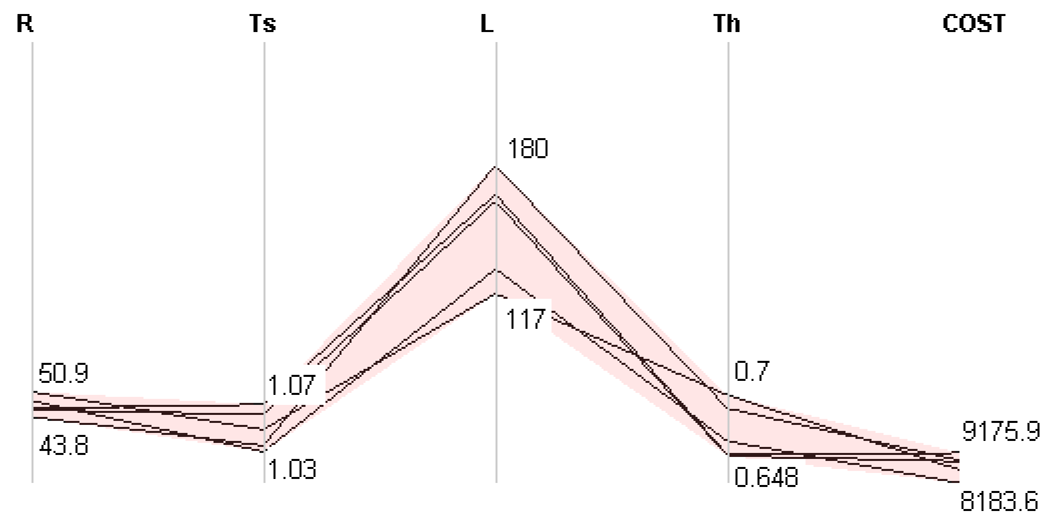


Figure 36 Parallel coordinates plot of PV problem (thin designs selected)

Overview

In order to know which 2D projections may be interesting, an overview of all of the 2D projections may be plotted. The value of an overview is not to identify specific numeric values as with the parallel coordinates, or to make selections, but to identify which projections are worth navigating to. Spence refers to this as a “see-and-go” approach as opposed to “go-and-see” [17]. Scatterplot matrices as overview was also discussed in detail by Elmqvist in [36] and employed in Stump et al.’s ATSV [21].

For example, in Figure 37, it is clear that the feasible region of R is restricted to a small range, due to defined constraints. There is also a negative correlation between R and L for the top performing designs, as shown in the L vs R subplot.

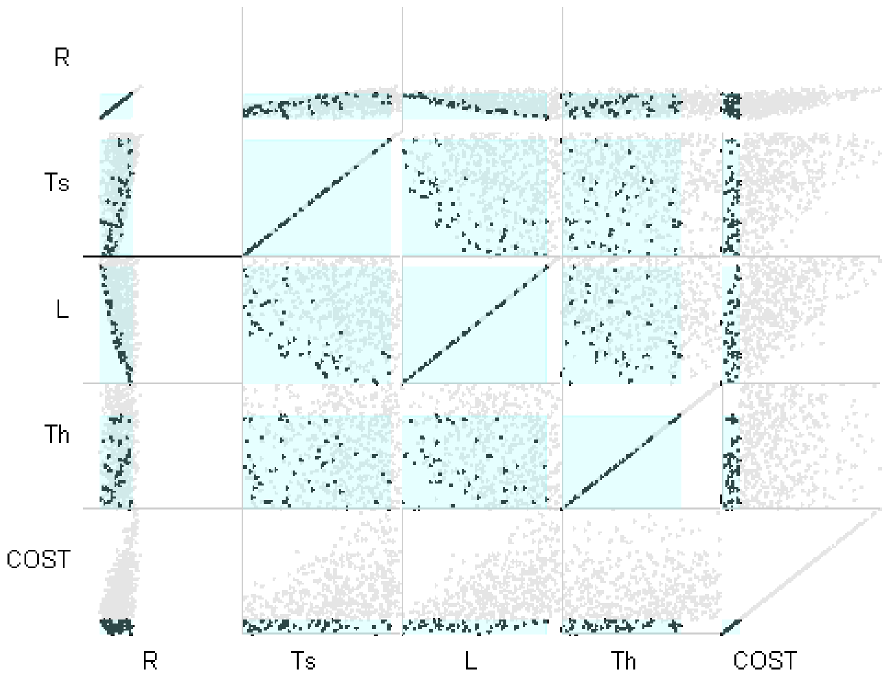


Figure 37 2D Projection overview of top 50 PV Designs

3.3.2. 2D Projection by Principal Components Analysis

A special 2D projection that arranges designs based on their overall variance is achieved using PCA. As mentioned in 2.2.3, PCA finds the orthogonal directions which maximize overall variance in the data. Therefore, by performing PCA on a selected region, the data will be separated to maximize the overall difference in designs. In other words, the corners of a PCA graph represent polarizing designs.

In Figure 39, the designs in the bottom-left corner are selected. These designs are long and slender with low shell thickness. In the other corner of the plot (Figure 40), the vessel is shorter and wider. This shows opposing possible pressure vessel designs (within the low cost percentile region).

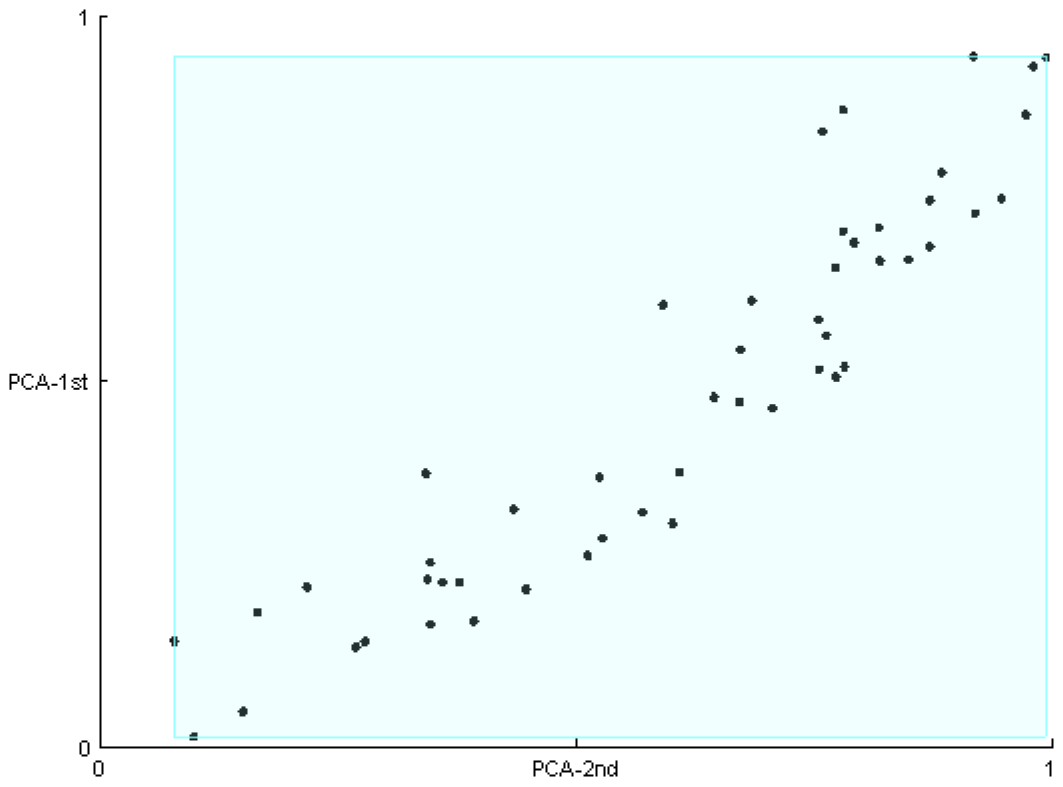


Figure 38 Two principal components of PV Problem (top 50 designs shown)

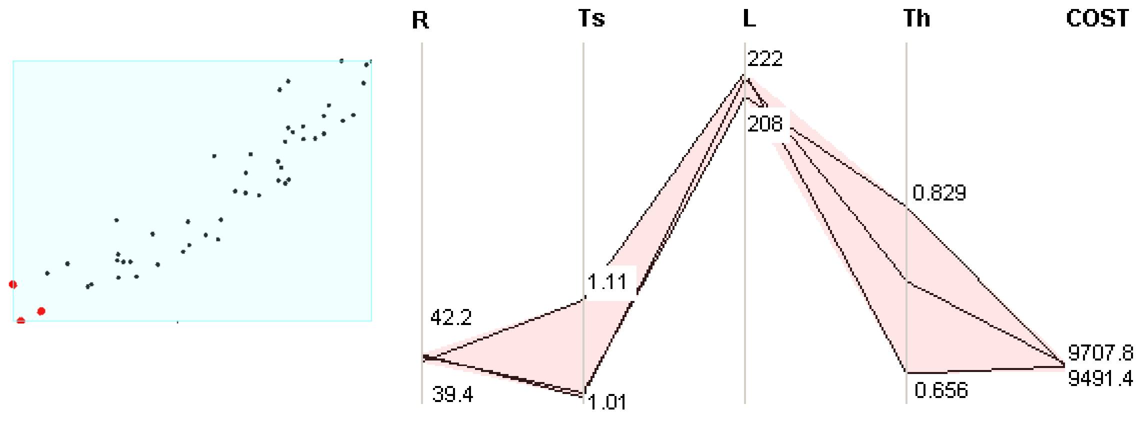


Figure 39 Designs selected in bottom-left of PCA plot (top 50 designs shown)

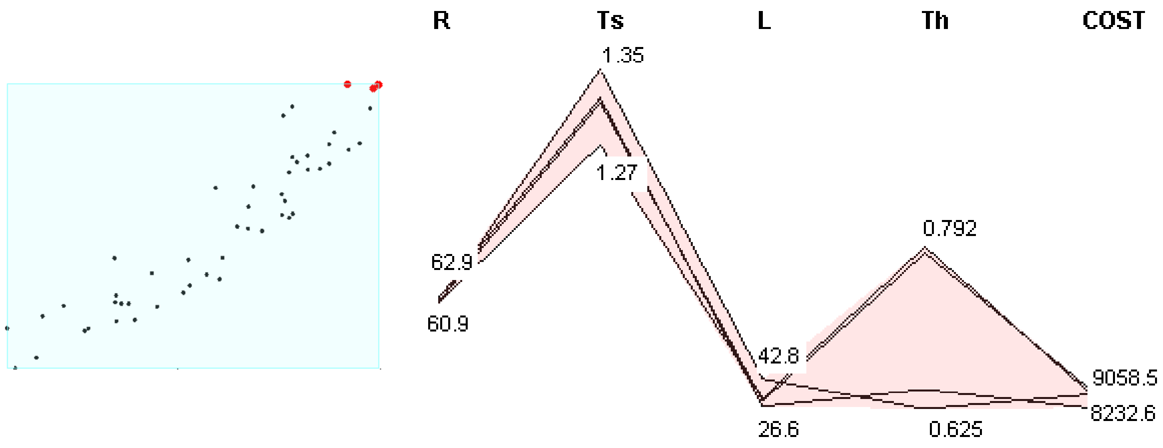
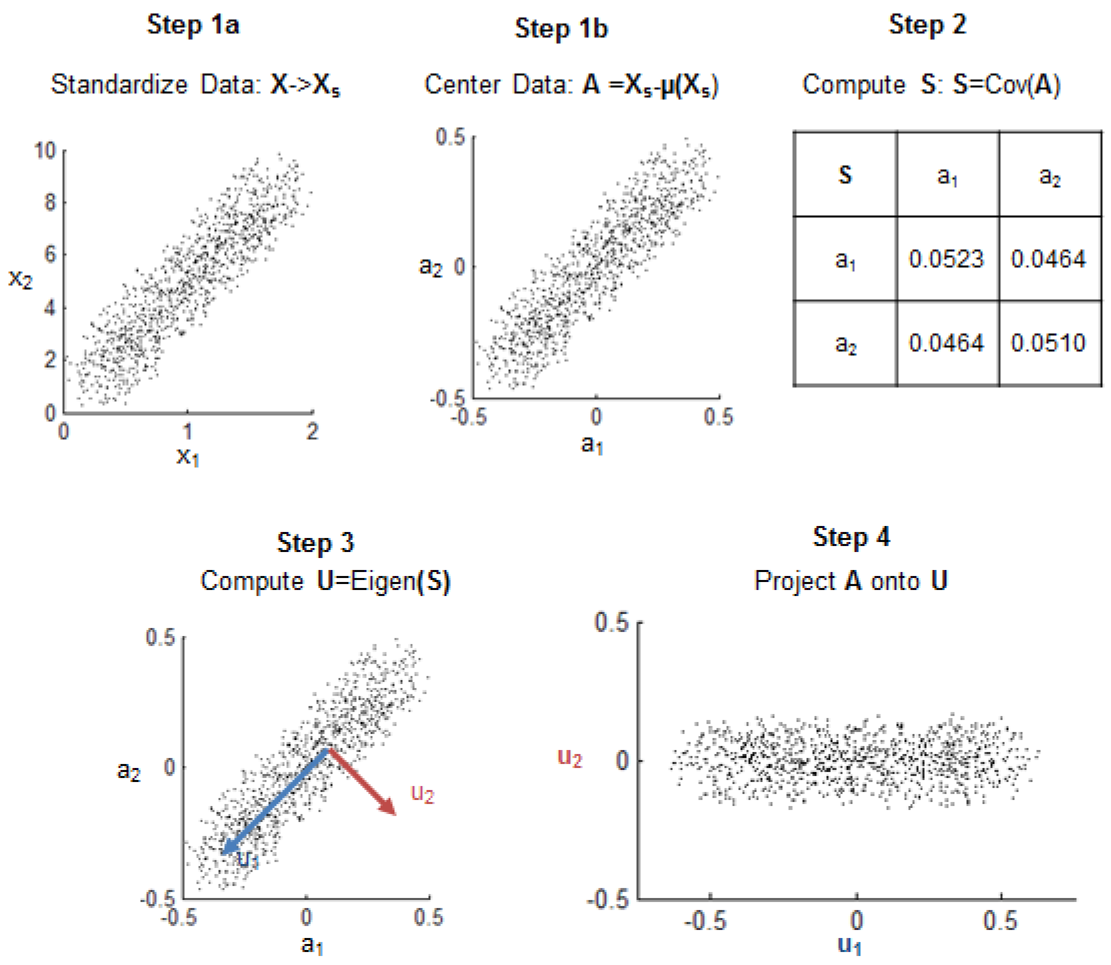


Figure 40 Designs selected in top-right of PCA plot (top 50 designs shown)

Procedure of PCA on optimization design data

The mathematical details of PCA can be found in Appendix. The first step is to standardize the data to remove the effect of varying units and ranges for different variables. The data is then centered to simplify computation of the covariance matrix. Next, the covariance matrix **S** is computed for the preprocessed data, which represents how each variable varies with respect to one another. The third step is to compute the orthogonal directions which maximize variance, by finding the eigenvectors $\boldsymbol{U}$ corresponding to the largest eigenvalues of **S**. The final step is to simply project the data onto the two largest eigenvectors of **S**. The steps are shown graphically in Figure 41 for a simple two-dimensional example.

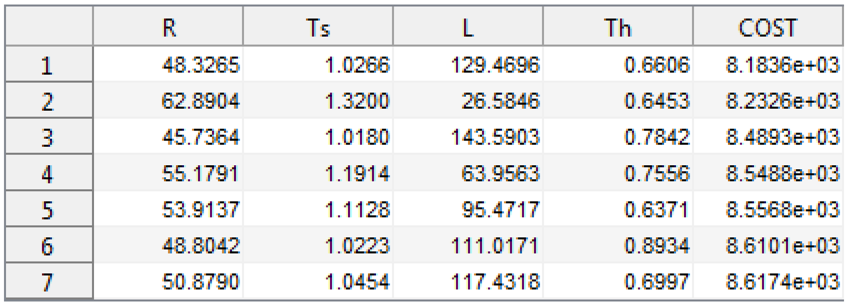


S	a_1	a_2
a_1	0.0523	0.0464
a_2	0.0464	0.0510

Figure 41 Graphical steps of Principal Component Analysis (Math in Appendix)

3.3.3. Data Table and Selection

An alternative to visual brushing is to simply use a sortable data table. If data is selected in the table, it may be brushed in the exact same manner as in the 2D projection (i.e. the points are highlighted in the 2D plots and plotted as a red region in the PCP graph as in Figure 36). It is also possible to highlight rows in the data table, as data is brushed in a graph. A sample data table is provided below for the PV problem, sorted by cost.



	R	Ts	L	Th	COST
1	48.3265	1.0266	129.4696	0.6606	8.1836e+03
2	62.8904	1.3200	26.5846	0.6453	8.2326e+03
3	45.7364	1.0180	143.5903	0.7842	8.4893e+03
4	55.1791	1.1914	63.9563	0.7556	8.5488e+03
5	53.9137	1.1128	95.4717	0.6371	8.5568e+03
6	48.8042	1.0223	111.0171	0.8934	8.6101e+03
7	50.8790	1.0454	117.4318	0.6997	8.6174e+03

Figure 42 Data table representation of PV Problem

3.4. Summary

This chapter began by describing the mathematical components of an optimization problem written in standard form. It was explained that multiple outputs of a simulation may be aggregated with weights to form a single objective problem. Next, the design, search, and performance spaces were defined as terms to distinguish between variable choices and their associated performance. Regions were also defined as three types: percentile region, performance region, or design region. These regions may be further split using clustering (e.g. k-means or single link) or by visual interaction (e.g. projection and brushing). Once a region is defined, it may be modeled locally using a metamodel for further investigation without additional sample data. Metamodeling is described in the following chapter.

Chapter 4. Regional Metamodeling

Metamodeling (a.k.a. surrogate modeling) is the process of modeling a computationally intensive model with an approximation of reduced computational cost. Simulations that require hours to compute may be replaced with simple approximations (metamodels) to be evaluated in milliseconds in exchange for fidelity. Metamodels range from simple Polynomial Regressions (PR) and Radial Basis Functions (RBF) to more complex High-Dimensional Model Representations (HDMR) [76]. Different families of models have been compared for their use in engineering design [77,78]. Consider Figure 43, which shows a 2^{nd} order polynomial regression model of Eq. (13) over a fixed interval. This chapter explains how local metamodeling is used in optimization and how it can approximate a selected design region.

$$f(\boldsymbol{x}) = x^3 + 0.5 * x;\ -0.5 \le x \le 1 \tag{13}$$

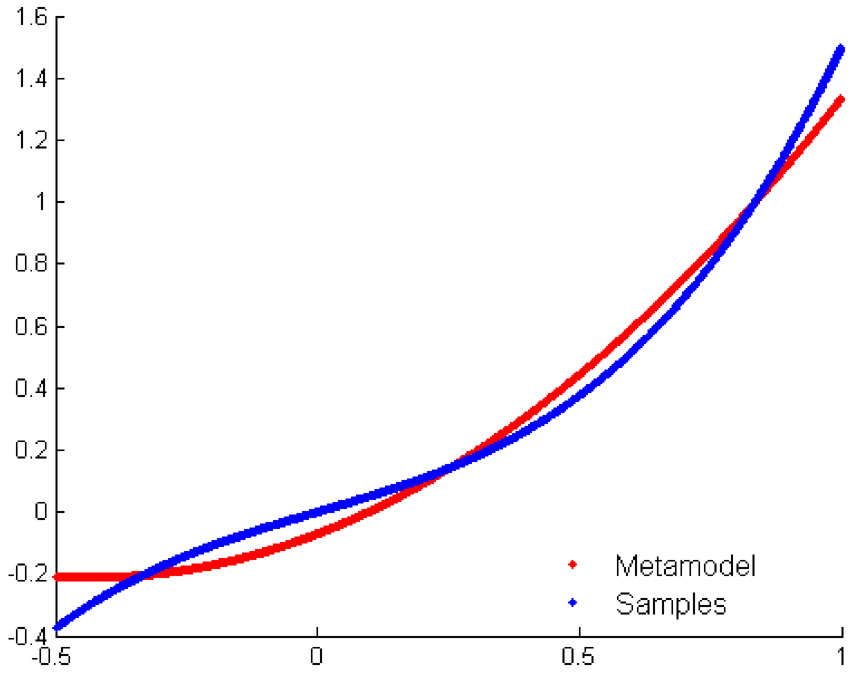


Figure 43 Single Variable Metamodel (2nd order PR approximating a 3rd order)

4.1. Metamodeling in optimization algorithms

As mentioned, in iterative optimization, an equation or simulation is evaluated by the optimization algorithm iteratively. This is considered “expensive” as the evaluation may take time.

In metamodel-based design optimization (MBDO), expensive samples train a metamodel, which is eventually evaluated in place of the expensive function. A measure of fitness, such as the root-mean-squared-error (RMSE), tests the model accuracy. Once the model is deemed accurate, often by a defined threshold, it can be optimized in place of the simulation. In fact, the model may only require accuracy in a sub-region of the design space, where points generate low cost values. Afterwards, a local optimization can be performed on the locally accurate metamodel.

In this thesis, a metamodel based algorithm called the Trust-Region Mode-Pursuing Sampling (TR-MPS) method, developed by Cheng and Wang [79], is used to generate new design points. The details of the method are beyond the scope of this work. The general process of MPS [74], from which TR-MPS is based, is shown in Figure 44. After initial sampling, a (linear piecewise/RBF) model is fit to the data. At this point, a discriminative sampling strategy is applied to generate a distribution of samples near the minimum. These points train a local quadratic metamodel. If the model is accurate (using the R-Squared metric), local optimization is performed. If not, additional samples are added and the model is updated. After the local optimization step, the minimum is verified with the expensive simulation, if the predicted minimum and the objective function value agree (as specified by a parameter), it is returned as an approximate global minimum. If the difference between the predicted minimum and simulation minimum differ significantly, additional samples are added and the loop continues.

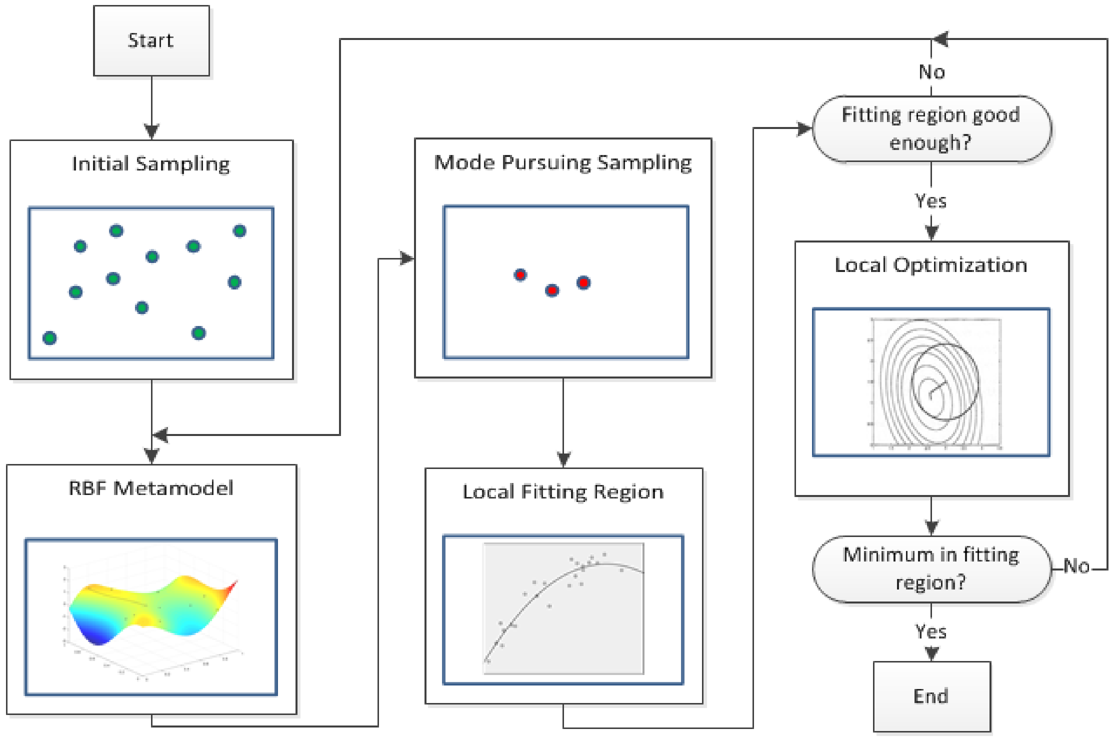


Figure 44 MPS Procedure (Figure provided by Cheng and Wang)

4.2. Metamodeling of a selected region

Aside from direct use in optimization (e.g. MPS), metamodeling can provide details about a user selected sub-region of the design space without additional expensive sampling. Eq. (14) is, one of the models available in the framework: a polynomial regression (PR), capped at second order for simplicity. This model is common in engineering optimization, especially in response surface methodologies (RSM). The model contains one constant term, n linear terms, n quadratic terms, and $\binom{n}{2}$ pairwise interaction terms. The set of models (which vary with $\boldsymbol{\beta}$) is the set of hypotheses $\boldsymbol{H}$. A particular hypothesis is denoted by h.

$$f(x) \approx h(x) = \beta_o + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \beta_{ii} x_i^2 + \sum\sum_{i<j} \beta_{ij} x_i x_j \tag{14}$$

The goal of model fitting is to find $\hat{h} \in \boldsymbol{H}$ which best approximates f. In particular, the goal is to reduce the sum of squared errors (SSE) between $h(x)$ and $f(x)$. It can be shown that the $\boldsymbol{\beta}$ that minimizes SSE (denoted by $\hat{\boldsymbol{\beta}}$) is given by Eq. (15).

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{f} \tag{15}$$

Note that $\boldsymbol{X}$ and $\boldsymbol{f}$ consist of the points in a selected design region *not* the global design space. If metamodeling over a global space, a 2nd order polynomial cannot be trusted to approximate $f(\boldsymbol{x})$ accurately. Determining when a region is sufficiently small for local modeling is the focus of *trust-region* research and is beyond the scope of this work. To compensate, model validation is discussed in the following section, which estimates the effectiveness of the local metamodel posteriori. A global method (e.g., HDMR) may also be used as an alternative.

4.2.1. Model validation using cross-validation

The validation of metamodels, beyond two dimensions is difficult to visualize [80]. A method to do so, was introduced by the author in [37]. Meanwhile an alternative to visualization is to use statistics to summarize the model's performance. Specifically cross-validation (CV) tests the suitability of a set of models ($\boldsymbol{H}$) for a set of data. CV uses a subset of the (regional) sample data to validate the model, instead of generating new random points. Ideally, new test data would be generated to estimate the model's performance, but this requires additional expensive sampling. Therefore, CV can be used as a quick indication of the success of metamodeling.

For clarification, training data $\boldsymbol{X_T}$ builds the model, validation data $\boldsymbol{X_v} = \{\boldsymbol{X}\}\backslash\{\boldsymbol{X_T}\}$ verifies the model and additional test data $\boldsymbol{X_A}$ evaluates the model's performance on completely new points. Error at the training points is called *in-sample* error, while *out-of-sample* error refers to the error at the test points. The purpose of cross-validation is to estimate the out-of-sample error without actually testing additional simulations, by reserving some of the given data for validation. Figure 45 shows the partitioning of data for cross validation.

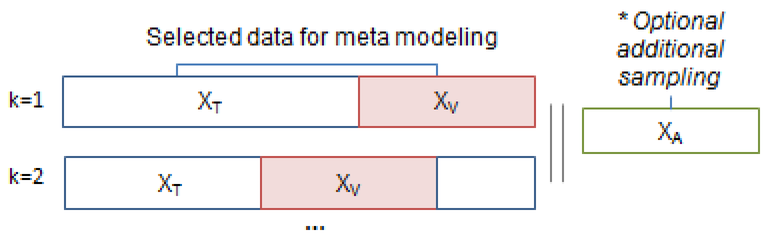

Selected data for meta modeling
* Optional additional sampling
k=1
X_T
X_V
X_A
k=2
X_T
X_V
...

Figure 45 Partitioning of data for cross-validation

In the case of k-fold cross-validation, m/k number of points are chosen for validation while $(m - m/k)$ points are used for training (m is the total number of regional samples). For example, if the region consists of 30 points, 3 are placed in $\boldsymbol{X_v}$ while the remaining 27 are placed in $\boldsymbol{X_T}$. A model ($\hat{h}_1$) is built from $\boldsymbol{X_T}$; the error is recorded, and the model is discarded. When computing the error, $\boldsymbol{X_v}$ is essentially out-of-sample as it was not used to build the model. Next, another 3 sample points, which haven't been in $\boldsymbol{X_v}$ previously, are selected to form a new $\boldsymbol{X_v}$. The remaining 27 points build another model ($\hat{h}_2$), which is tested with $\boldsymbol{X_v}$ and discarded. This process loops until all points have been used for validation (creating a total of k models). Afterwards, all of the points are placed into $\boldsymbol{X_T}$ to train a final model. The mean validation error for all of the models is reported. This is an estimate of the out-of-sample error of $\boldsymbol{H}$ as in Eq. (16).

$$E_{CV}(\boldsymbol{H}) = \frac{1}{k}\sum_{i=1}^{k} \mu[\,\boldsymbol{e}(\,\hat{h}_i(\boldsymbol{X_v}), f(\boldsymbol{X_v})\,)\,] \tag{16}$$

In Eq. (16) μ is the mean operator, and $\boldsymbol{e}$ is the error of the validation set $\boldsymbol{X_v}$. In this work, error is defined as the root-mean-squared error (RMSE) as in Eq. (17). The RMSE of cross-validation ($\boldsymbol{X_e} = \boldsymbol{X_v}$) is the RMSECV, while the RMSE on additional test points ($\boldsymbol{X_e} = \boldsymbol{X_A}$) is simply RMSE. Additionally, the NRMSE or NRMSECV are the RMSE and RMSECV divided by the local function value range as in Eq. (18). If cross-validation is successful, RMSECV will be similar to RMSE (which can be confirmed using additional samples if desired).

$$e(\,\hat{h}_i(\boldsymbol{X_e}), f(\boldsymbol{X_e})\,) = RMSE = \sqrt{\mu\left([f(\boldsymbol{X_e}) - \hat{h}_i(\boldsymbol{X_e})]^2\right)} \tag{17}$$

$$NRMSE = \frac{RMSE}{\max[f(\boldsymbol{X})] - \min[f(\boldsymbol{X})]} \tag{18}$$

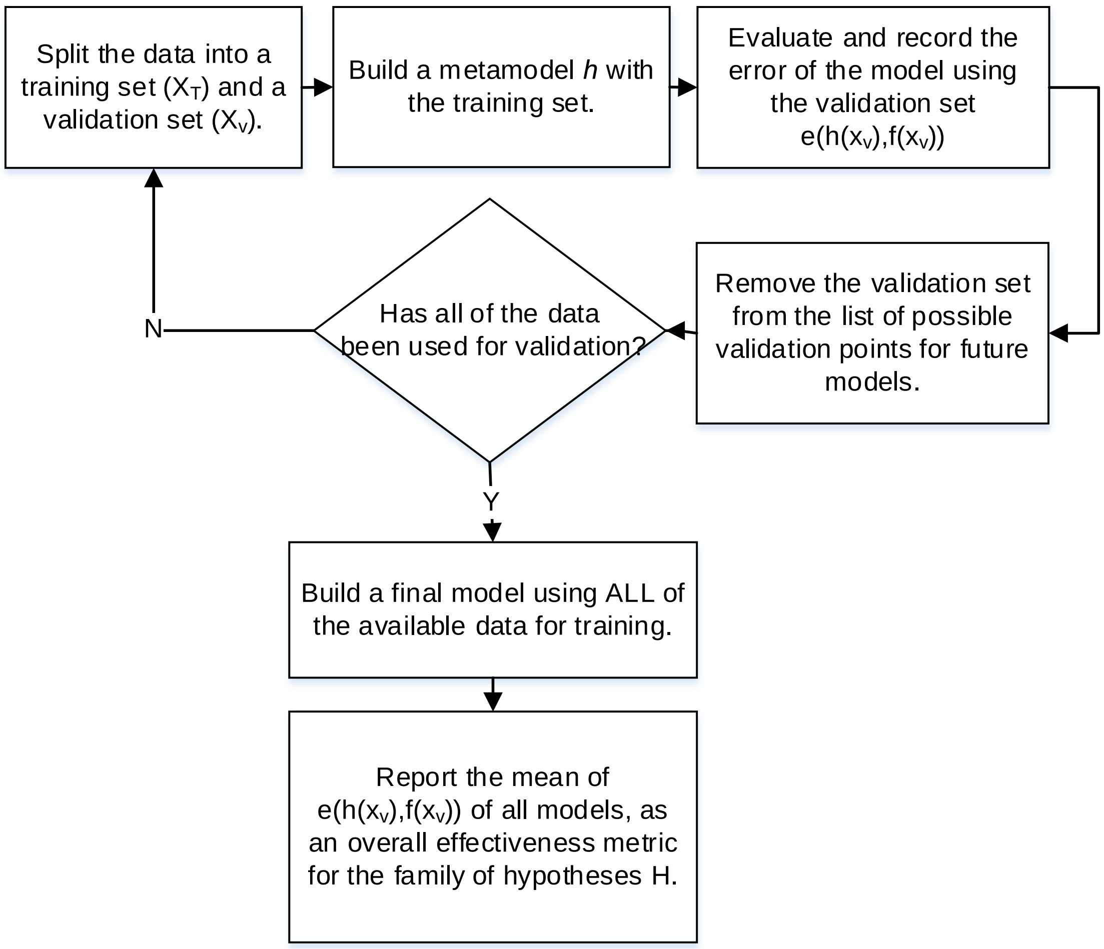

Split the data into a training set (X_T) and a validation set (X_V).
Build a metamodel h with the training set.
Evaluate and record the error of the model using the validation set e(h(x_V),f(x_V))
Remove the validation set from the list of possible validation points for future models.
Has all of the data been used for validation?
N
Y
Build a final model using ALL of the available data for training.
Report the mean of e(h(x_V),f(x_V)) of all models, as an overall effectiveness metric for the family of hypotheses H.

Figure 46 Cross validation process

4.2.2. Model validation visualization

In the framework, the metamodel is shown with a simple visualization, adjacent to a table of error measures. The points in the design region which bound the top fifty PV designs are shown in Figure 47. The points are sorted and plotted by performance, overlaid on the metamodel approximations. The important feature of this plot is the (lack of) difference between the simulation data and metamodeling data at the training and testing points.

It should also be noted that the number of points in the design region which bound the top performing designs is relatively large (305 points are contained in the box that bounds the top 50 designs).

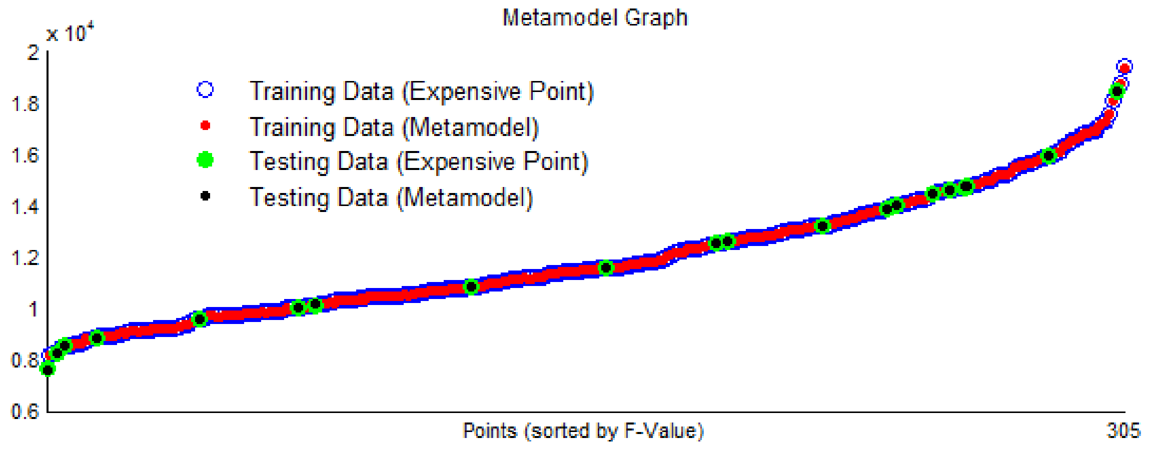


Figure 47 Metamodel for design region which bounds the top 50 PV designs.

From Figure 47, it is clear that the 2nd order model is accurate for this data. The red (metamodel) points are overlaid directly on the expensive points. There is no noticeable difference between the simulation and metamodel at the training points, or at the test points. This is confirmed by the table of error metrics in Table 2. The RMSE and NRMSE were computed with an additional 20 random test points (shown in green on Figure 47). The value of k for k-fold CV is 10.

Another statistic presented in Table 2 is R-Squared. R-Squared also measures goodness of fit using only $\boldsymbol{X_T}$, as in Eq. (19). It is computed for the final model where $\boldsymbol{X_T} = \boldsymbol{X}$**.** The downside to R-Squared is that it increases with the number of variables (and model complexity), even if they have no effect on the response.

$$R - Squared = 1 - \frac{\sum(f(\boldsymbol{X_T}) - \hat{h}(\boldsymbol{X_T}))^2}{\sum(f(\boldsymbol{X_T}) - \mu(f(\boldsymbol{X_T})))^2} \tag{19}$$

Table 2 Metamodel error metrics for top performing PV design region

R-Squared	1.00
RMSECV	15.89
NRMSECV	0.001

RMSE	15.79
NRMSE	0.001

4.2.3. Predicting local extremes

If the metamodel has a low NRMSE (<0.1), local optimization can be performed on the metamodel to predict the best and worst case performance of the region. This is executed using MATLAB's constrained optimization solver: *fmincon*. Additionally, the expensive simulation may be executed for the predicted min and max to see if they match. The results are summarized in Table 3 for the PV problem. The predicted cost extremes are close to the actual costs. Furthermore, the minimum cost design has significantly lower cost than the best case found from the 500 random samples in $\boldsymbol{X}$ (the best case was $8183.6). The results also show the massive range in cost ($7k to $21k+) within the box that bounds the top 50 designs. This suggests that a tighter region is required if the goal is to find ranges for variables that result in low cost designs.

Table 3 Predicted and simulated performance extremes of PV top design region

	COST [$]	R [“]	Ts [“]	L [“]	Th [“]
Min (Predicted)	7005.3	52.15	1.007	82.13	0.625
Max (Predicted)	21674	63.06	1.351	222.4	0.9054
Result @ Predicted Min	7018.5				
Result @ Predicted Max	21759				

4.2.4. Predicting local sensitivity

A second use of the local metamodel is to predict the sensitivity of variables in a local region by checking the magnitude of coefficients in the metamodel. The coefficients are shown in a bar chart, color coded by sign (red is negative and blue is positive). From Figure 48 the terms T_s and T_h, related to thickness, appear to be most influential in the top performing region. However, this “sensitivity” information is really measuring the weight of terms in the metamodel. The model may simply be compensating for missing higher order relationships, by increasing the weight of a quadratic term for example. Therefore, the bar chart should not be interpreted as the overall importance of individual variables (even locally), especially if the model fit is poor. Rather, it is simply the importance of terms in the 2nd order approximation.

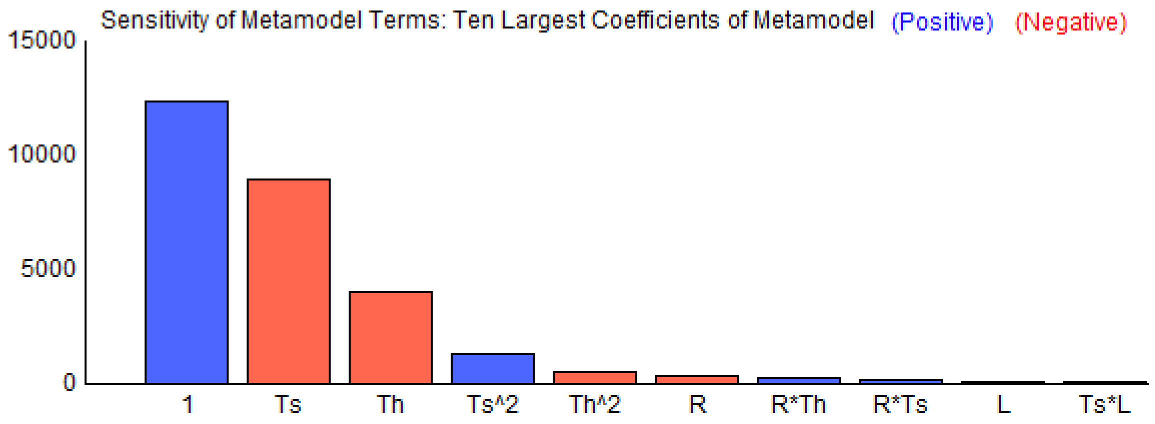


Figure 48 Coefficients of terms in 2nd order polynomial of PV top design region

4.2.5. High Dimensional Model Representation (HDMR)

As mentioned above, polynomial regression is often sufficient for modeling local-regions where the function is relatively quadratic or linear. However, in many cases, a global model is required. High dimensional model representation (HDMR) is a global metamodeling technique which splits the objective function into a sum of smaller component functions. The component functions represent the independent and cooperative contributions of each design variable or set of design variables. Some variations include Cut-HDMR [81], RS-HDMR [82,83], RBF-HDMR [84] and PCA-HDMR [85]. For brevity, this work only brefily touches on the mathematics of HDMR. A more detailed explaination is found in [86]. The general form of HDMR is shown in Eq. (20).

$$f(\boldsymbol{x}) = f_0 + \sum_{i=1}^{d} f_i(x_i) + \sum_{1\le i<j\le d} f_{ij}(x_i, x_j) + \cdots + \sum_{1\le i<j<k\le d} f_{i_1,i_2\ldots i_l}(x_{i_1}, x_{i_2}, \ldots, x_{i_l}) + \cdots + f_{12\cdots d}(x_1, x_2, \cdots, x_d) \tag{20}$$

Each component function in Eq. (20) contributes to $f(\boldsymbol{x})$, the HDMR of the system. In this expression, f_o is the zeroth order term, which is irrespective of $\boldsymbol{x}$. $f_i(x_i)$ are the first-order terms, representing the effects of each variable acting alone on *f*, without correlating with other variables. $f_{ij}(x_i, x_j)$, the second order terms, represent the correlations between any two variables. $f_{12\cdots d}(x_1, x_2, \cdots, x_d)$, the *d*th order terms, represent the correlations between *d* variables. Fortunately, due to the nature of engineering problems, design variables are typically selected to be highly independent [86]. Therefore, higher correlation terms of order *l* (*l*<*d*) tend to have a negligible effect on *f*($\boldsymbol{x}$). In this work, l is fixed to two, meaning only the second order correlations are considered. The number of components function bases is also two, as in [85].

PCA-HDMR

In 2012, Hajikolaei and Wang developed a method that uses principal component analysis (PCA) to determine weights for orthogonal basis (component) functions [85]. This method entitled PCA-HDMR, was shown to be significantly more accurate than standard random sampling (RS) HDMR. In general, adding more samples towards building a PCA-HDMR increases its accuracy [85]. Moreover, PCA-HDMR does not require a uniform distribution of sample points. This means it may be used for metamodeling with biased optimization data. In contrast, RS-HDMR demands uniformly distributed points for building the model; if newly added sample points are not uniform, the quality of the RS-HDMR deteriorates.

Use of PCA-HDMR as an alternative to polynomial regression

In this work, PCA-HDMR is provided as an alternative model to polynomial regression. If enough samples exist to build a HDMR model, HDMR may be superior to second order polynomial regression. The number of points required to build a PCA-HDMR is given by NC in Eq. (25), from [85]; where L and s are the order of correlation terms and number of basis functions (respectively), and n is the number of variables. Generally, more samples are required to build a PCA-HDMR model, in comparison to 2nd order PR, but PCA-HDMR may be more accurate for non-quadratic functions.

$$NC = \sum_{i=1}^{L} \binom{n}{i} s^i \tag{21}$$

The results of three test functions from [85], Eqs. (22) to (24), are presented. Each model was built with 1000 points,generated randomly in the design space. In each case, the value for NC is less than 1000. Twenty additional test points were used to compare the RSME and RMSECV.

Example Problem 1: Five variable polynomial (NC=50)

$$f(\boldsymbol{x}) = (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 \quad s.t. \quad x_{1\ldots5} \in [-2,2] \tag{22}$$

Table 4 Comparison of PR and HDMR: Five variable polynomial

2nd Order Regression		PCA-HDMR	
R-Square	0.86	R-Square	0.98
RMSECV	64.0	RMSECV	28.3
NRMSECV	0.087	NRMSECV	0.038
RMSE	47.6	RMSE	27.4
NRMSE	0.065	NRMSE	0.037

Example Problem 2: Ten variable sum (NC=200)

$$f(\boldsymbol{x}) = \left[\sum_{i=1}^{10} i^3 (x_i - 1)^2\right]^3 \quad s.t. \quad x_{1\ldots10} \in [-3,3] \tag{23}$$

Table 5 Comparison of PR and HDMR: Ten variable sum

2nd Order Regression		PCA-HDMR	
R-Square	0.87	R-Square	1
RMSECV	1.95e+12	RMSECV	1.14e+12
NRMSECV	0.050	NRMSECV	0.034
RMSE	1.83e+12	RMSE	1.07e+12
NRMSE	0.047	NRMSE	0.0314

Example Problem 3: Twenty variable sum (NC=800)

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{\boldsymbol{i}=\mathbf{1}}^{\mathbf{10}} [\mathbf{100}(\boldsymbol{x}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}+\mathbf{10}})^{\mathbf{2}} + (\boldsymbol{x}_{\boldsymbol{i}} - \mathbf{1})^{\mathbf{2}}] \quad s.t. \quad x_{1\ldots 20} \in [-3{,}5] \tag{24}$$

Table 6 Comparison of PR and HDMR: Twenty Variable sum

2nd Order Regression		PCA-HDMR	
R-Square	0.87	R-Square	1
RMSECV	3.59e-11	RMSECV	2.66e-10
NRMSECV	2.37e-15	NRMSECV	9.60-15
RMSE	3.53e-11	RMSE	9.98e-11
NRMSE	2.33e-15	NRMSE	3.59e-15

Discussion

From the results above, PCA-HDMR is comparable to 2nd order PR and often yields better performance (as in Table 4). However, PCA-HDMR requires more samples than a simple 2nd order model, which only requires $2n + \binom{n}{2} + 1$ points. Additionally, PCA-HDMR may over fit the data if the chosen order L exceeds that of the underlying cost function. Therefore, it is recommended to validate the model using additional test points, regardless of which model family is used. If additional test data is costly, cross-validation may be used as an approximation of the RMSE and NRMSE.

4.3. Summary

Metamodels are built to approximate expensive simulations, often in a small region of the design space. If a metamodel is accurate, it may be used in place of the expensive simulation for intensive sampling tasks like local optimization. This chapter presented PR and HDMR as metamodeling techniques for selected design regions. The model is then cross-validated to estimate out-of-sample performance, with minimal additional expensive sampling. The results are plotted to show performance and the importance of terms in the model. The following chapter combines the discussed techniques from Chapter 3 and Chapter 4 into a prototype application for interactive optimization.

Chapter 5. Optimization Visualization Framework

Chapter 3 presented techniques to define regions in optimization data and methods for navigation. Chapter 4 presented techniques for local metamodeling and validation. This chapter shows how these techniques may be combined into a cohesive framework for visualization of optimization data.

5.1. Framework Layout

A prototype visualization framework was implemented in MATLAB (due to the availability of algorithm codes). The layout is shown in Figure 49.

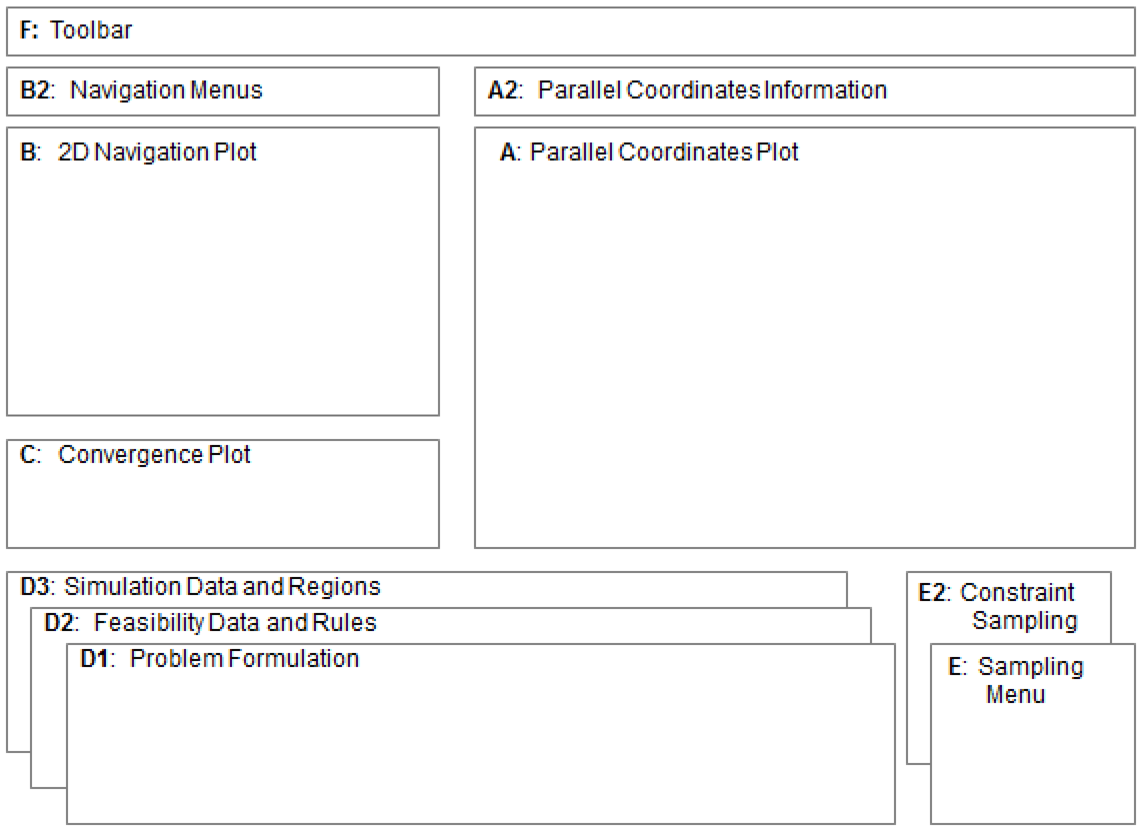
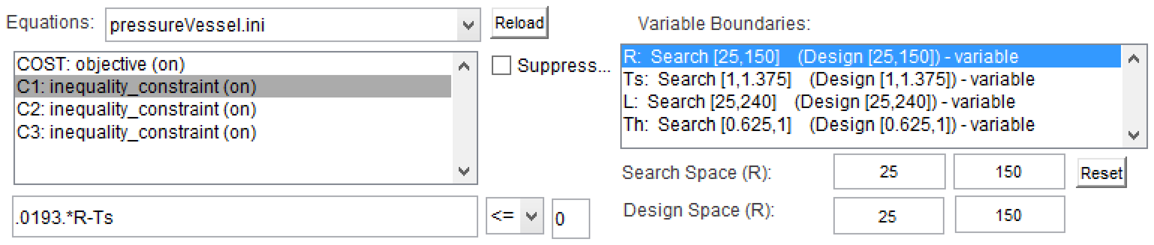


Figure 49 Optimization visualization framework layout

5.1.1. Problem Formulation (D1)

The first step in optimization is problem definition. Problems are defined in a problem definition file containing the objective, constraints, variable names and bounds. Once loaded into the framework, the equations, design and search spaces can be modified. Constraint equations may be changed or suppressed at any time before, during, or after optimization. Data that was previously feasible is hidden from the user if it becomes infeasible due to changes in constraints.



Equations: pressureVessel.ini

COST: objective (on)
C1: inequality_constraint (on)
C2: inequality_constraint (on)
C3: inequality_constraint (on)

.0193.*R-Ts <= 0

R: Search [25,150] (Design [25,150]) - variable
Ts: Search [1,1.375] (Design [1,1.375]) - variable
L: Search [25,240] (Design [25,240]) - variable
Th: Search [0.625,1] (Design [0.625,1]) - variable

Search Space (R): 25 150
Design Space (R): 25 150

Figure 50 Problem formulation menu

5.1.2. Feasibility Data (D2)

The second step (if the problem is constrained) is to check the necessity of the various constraints, and to identify the feasible region. A novel algorithm to identify constraint redundancy for black-box problems is presented in Chapter 6. In addition, constraint data can be visualized. To check feasibility, it is not necessary to execute the objective function. Consequently, constraints are sampled and visualized separately from the simulation data.

A plot of the overall feasible region for the PV problem is shown in Figure 51 (with 1200 constraint checks). Individual constraints may also be plotted to highlight the effect of the constraint acting alone. The feasible region is shown on the 2D navigation plot in Figure 52. Meanwhile, an adjacent bar chart plot specifies the percentage of sampled points that were feasible as a measure of overall restrictiveness. Last, an optional overview of all of the 2D projections can be shown by clicking the “Show Overview” button in the Toolbar (Section F) as in Figure 54.

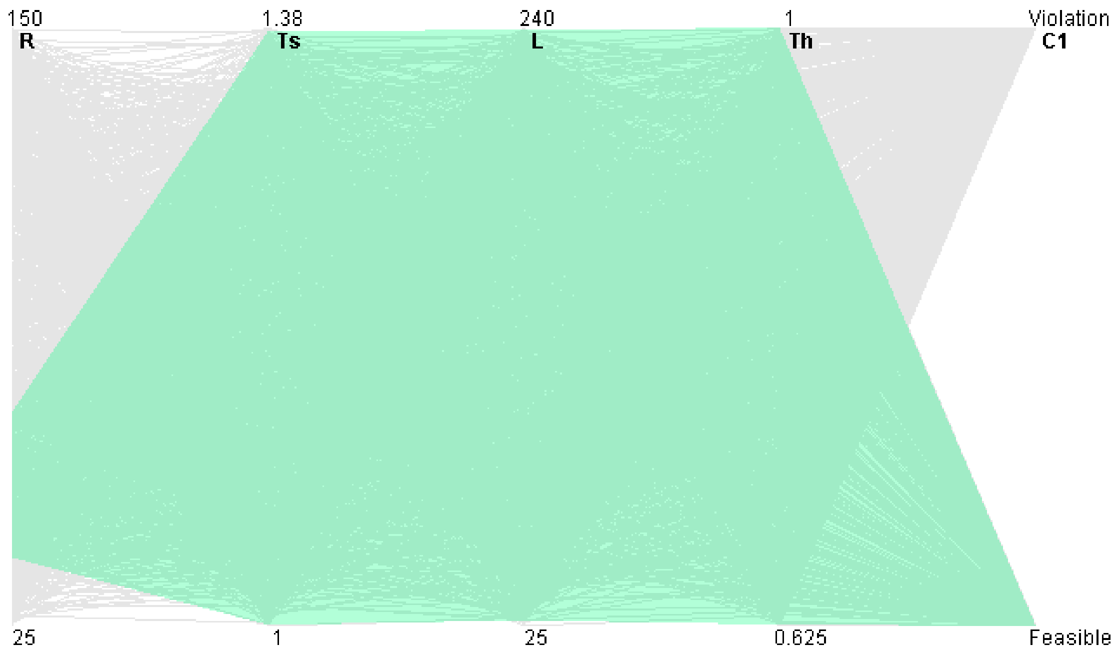


Figure 51 Feasible Region of the PV Problem

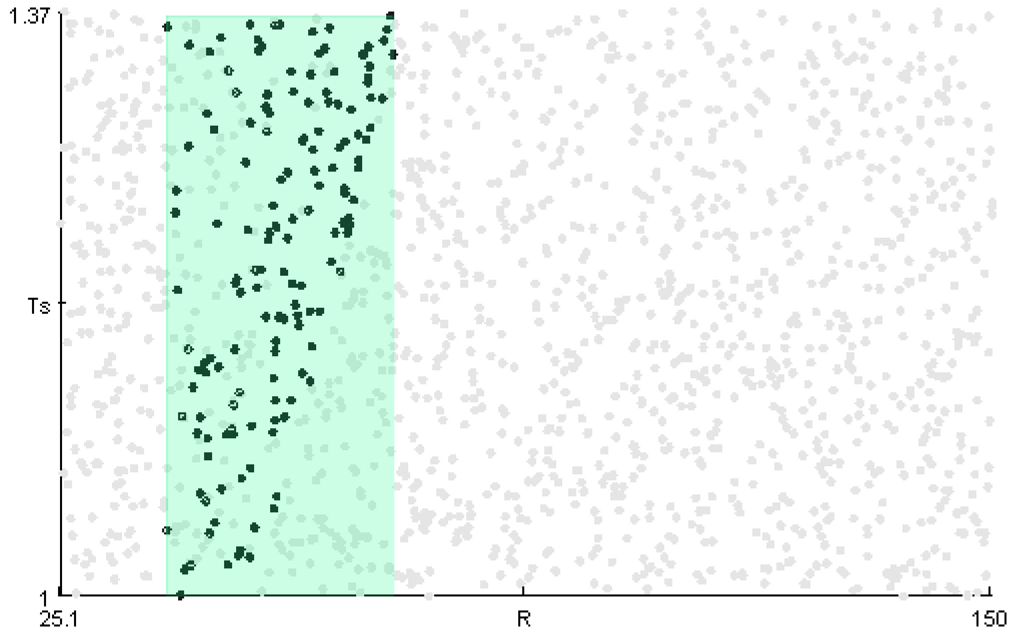


Figure 52 Overall feasibility as 2D projection

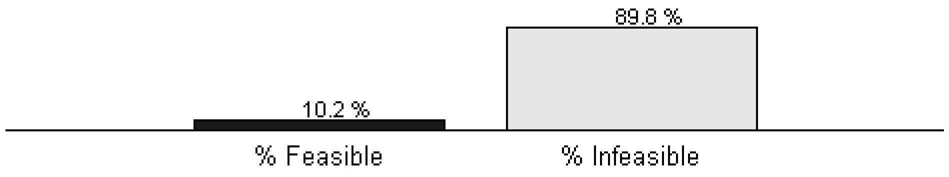


Figure 53 Overall feasibility as a bar chart

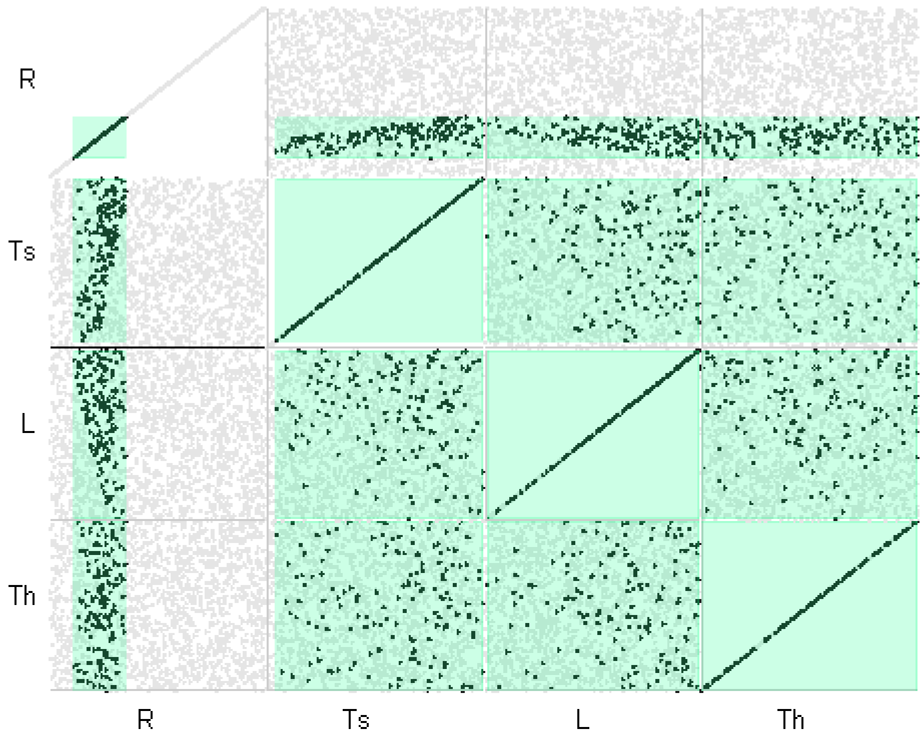


Figure 54 Feasible region as a 2D projection overview (scatterplot matrix)

5.1.3. Simulation Data, Convergence and Regions (D3)

The third step is to generate data by calling the expensive simulation on a set of points. There are two methods to generate data. The first method is to simply generate and evaluate a specified number of random samples within the search bounds. This is ideal for exploratory analysis of the search space, to widely cover the space. The main downside to random sampling is that it wastes time simulating points which are far from the optimum.

Therefore, the second method of data generation is to collect data while running an optimization method such as TR-MPS. TR-MPS aggressively biases samples towards the optimum, wasting few samples. In this case, the data is updated and plotted iteratively as in Figure 55. The downside to generating data from optimization is the lack of coverage in the overall search space due to the bias towards the optimum.

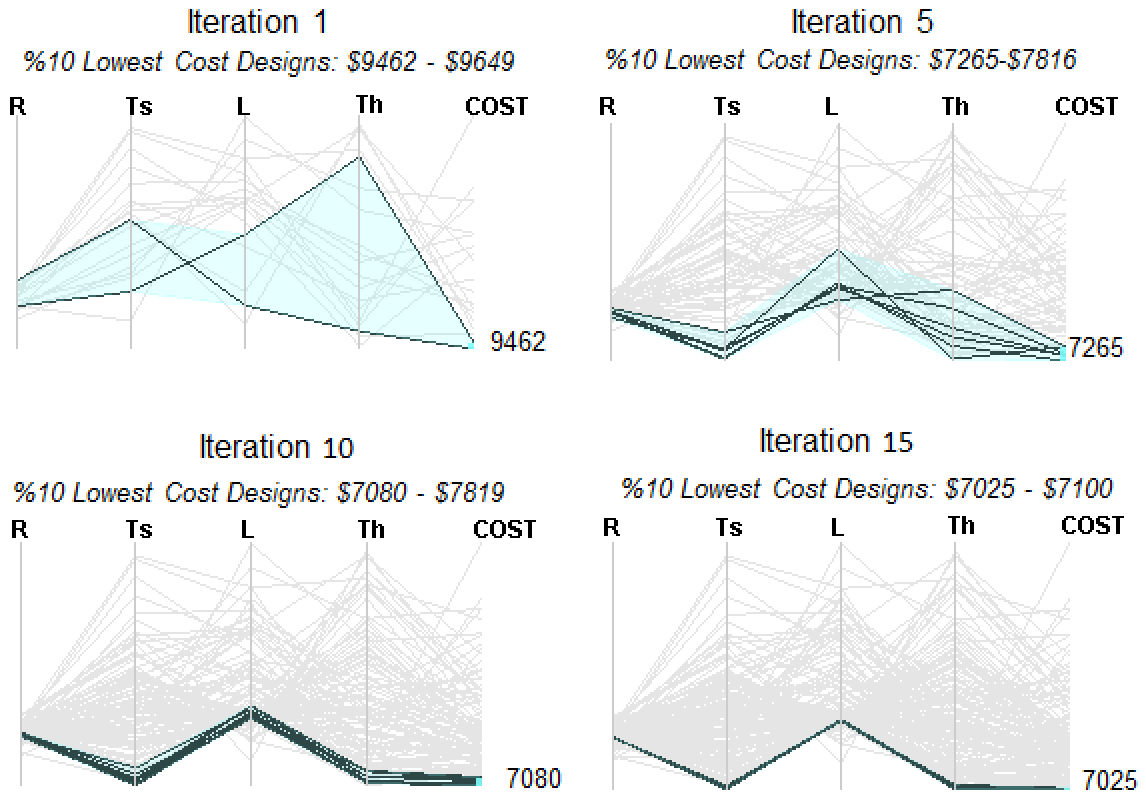


Figure 55 PV Problem top 10 % lowest cost region shown iteratively (with TR-MPS)

From Figure 55, a few observations can be made about the optimization process. First, as expected, the cost function is being effectively minimized. Second, the top designs for the PV problem form only one natural cluster, unlike the SC problem, where multiple design regions performed well. Third, the region containing the top 10% of lowest cost designs is converging. This is expected as TR-MPS continues to add additional samples near the top performing points. By iteration 15, (with 186 total function evaluations) optimization has roughly converged to the following design region:

$$R^{*}: [51.7,51.9] \;\; T_{s}^{*}: [1,1.01] \;\; L^{*}: [84.4,86.5] \;\; T_{h}^{*}: [0.625,0.633] \rightarrow COST\$: [7025,7100]$$

Convergence

Another common way to visualize convergence is to simply plot the function value vs iteration as in Figure 56 (Area C in the layout). In this plot, the global minimum found so far is graphed each iteration to show the amount improvement from continuing the optimization. The colors of points also indicate which iterations introduced points in the selected region. In the case of Figure 56, it is clear that the lowest cost designs were introduced in the latter iterations (colored in cyan). Similarly, if a point is brushed in the 2D navigation or data table, the iteration that introduced the brushed point is highlighted as shown in Figure 57. Figure 57 shows that the thicker pressure vessel designs were tested in earlier iterations of optimizations and then dismissed.

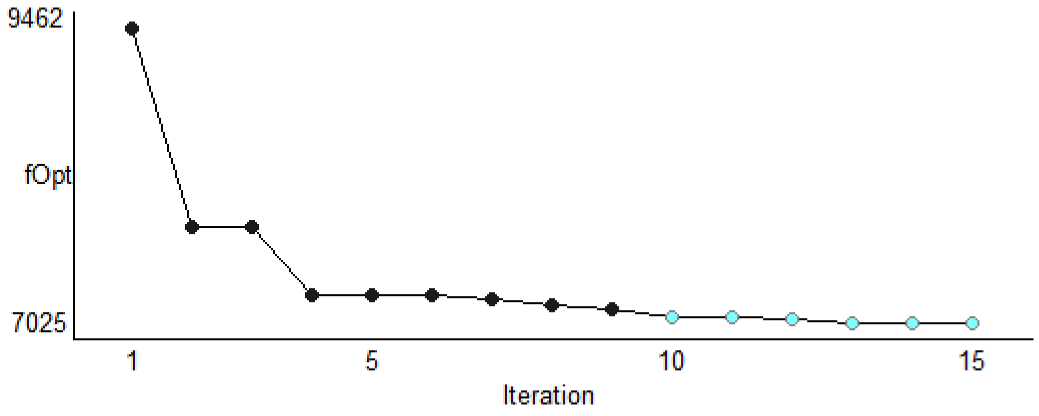


Figure 56 Iterative convergence plot of lowest cost design

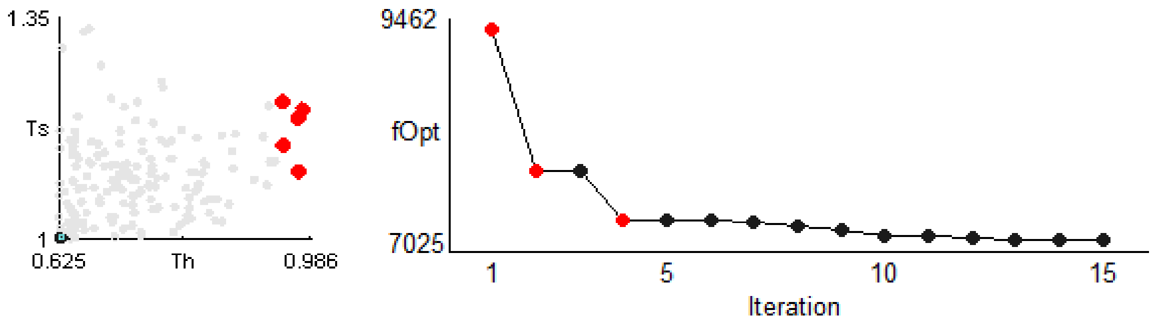


Figure 57 Iterations of selected points (thick pressure vessel designs)

Reviewing past iterations

In addition to viewing the convergence of function values, users may also navigate to past iterations from the Navigation Menu (B2). Here, the effectiveness of the optimization's sampling strategy is reviewed by comparing the designs that were introduced each iteration (highlighted in red). For example, in Figure 58, the new points introduced in the last iterations are mostly near the optimum, whereas the initial sampling is randomly distributed. This is indicative of TR-MPS's aggressive sampling in later stages.

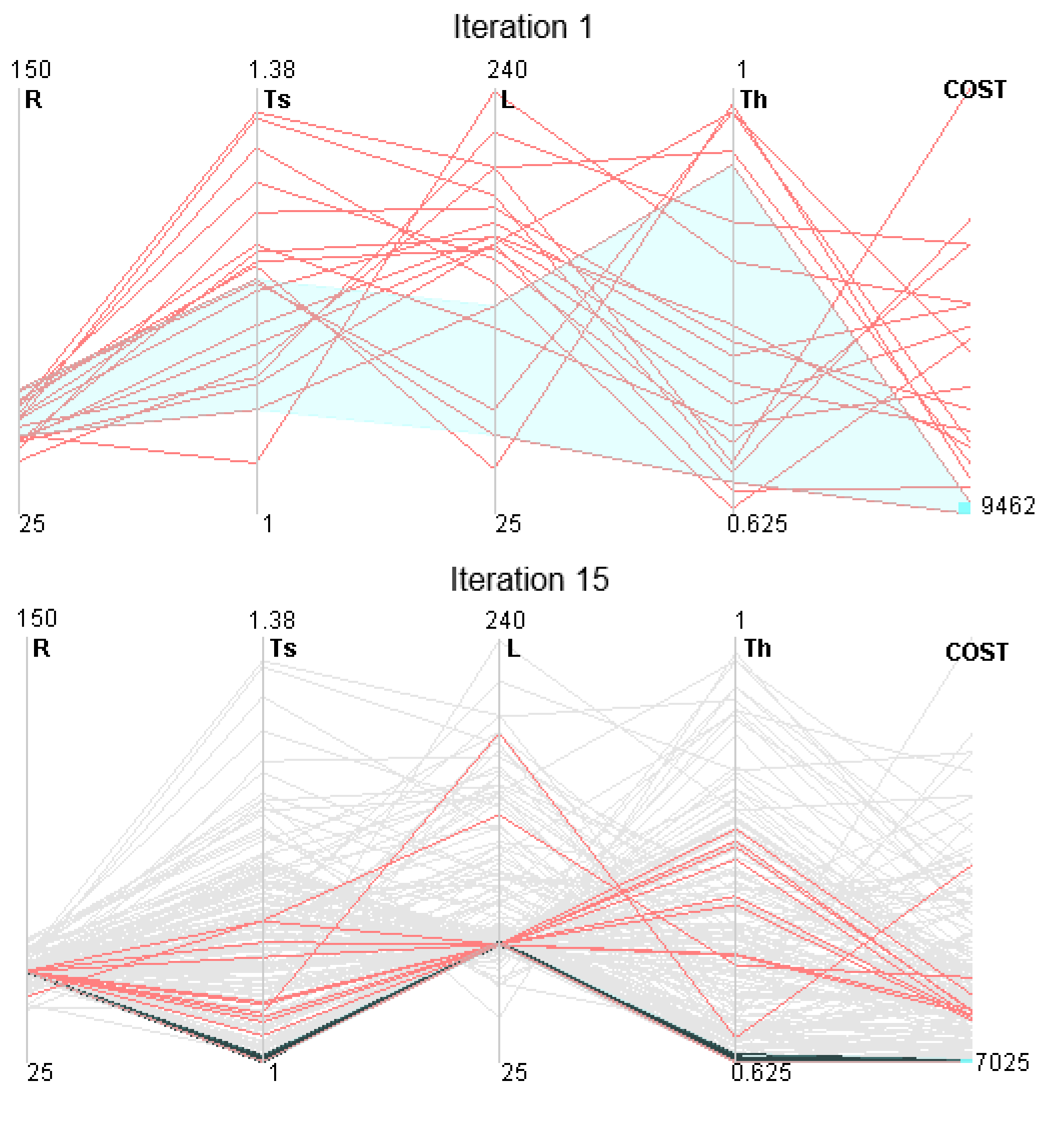


Figure 58 Designs introduced by iteration for PV problem optimization

Regions

As explained in Chapter 3, three types of regions may be defined to select areas of optimization spaces: (1) percentile regions, (2) performance regions, and (3) design regions. The regions are selectable and editable from a list as in Figure 59. Regions can also be combined by clicking on “In all tracked regions” or “In any tracked region”.

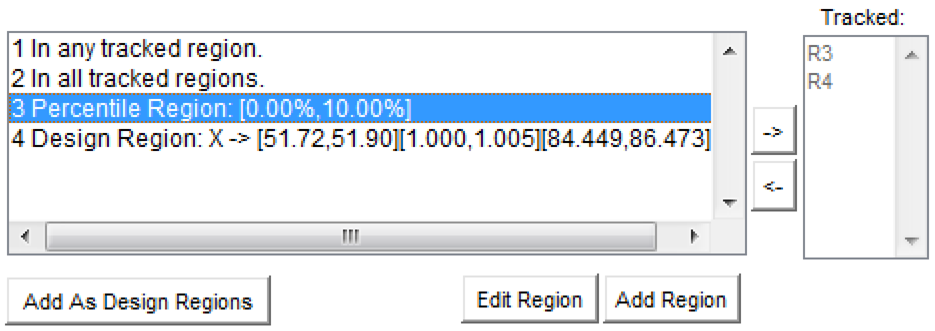

Tracked:
1 In any tracked region.
2 In all tracked regions.
3 Percentile Region: [0.00%,10.00%]
4 Design Region: X -> [51.72,51.90][1.000,1.005][84.449,86.473]
R3
R4
->
<-
Add As Design Regions
Edit Region
Add Region

Figure 59 Sample list of regions

Regions are defined using the *Add Region* button, which opens an interface for region definition. Similarly, *Edit Region* allows regions to be modified. Regions can also be interactively adjusted using sliders (placed in the D3 area of the layout), for continuous interaction. The value of reducing interaction cost to facilitate exploration has been emphasized by Spence [68] and Lam [72].

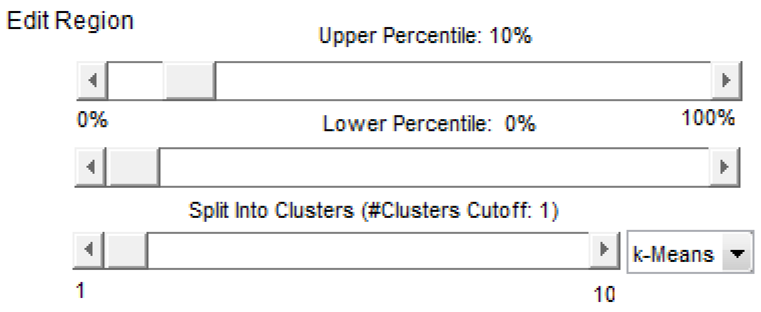


Figure 60 Region adjustment sliders

Two additional interactive methods are available to specify design regions. First, users may choose to automatically create a design region from a percentile or performance region. Upon clicking *Add as Design Regions* the corresponding design bounds (shown in cyan or orange) will be added as a design region. If the region is split to clusters, a design region will be formed for each cluster. This is shown in Figure 61 for the SC problem after clustering.

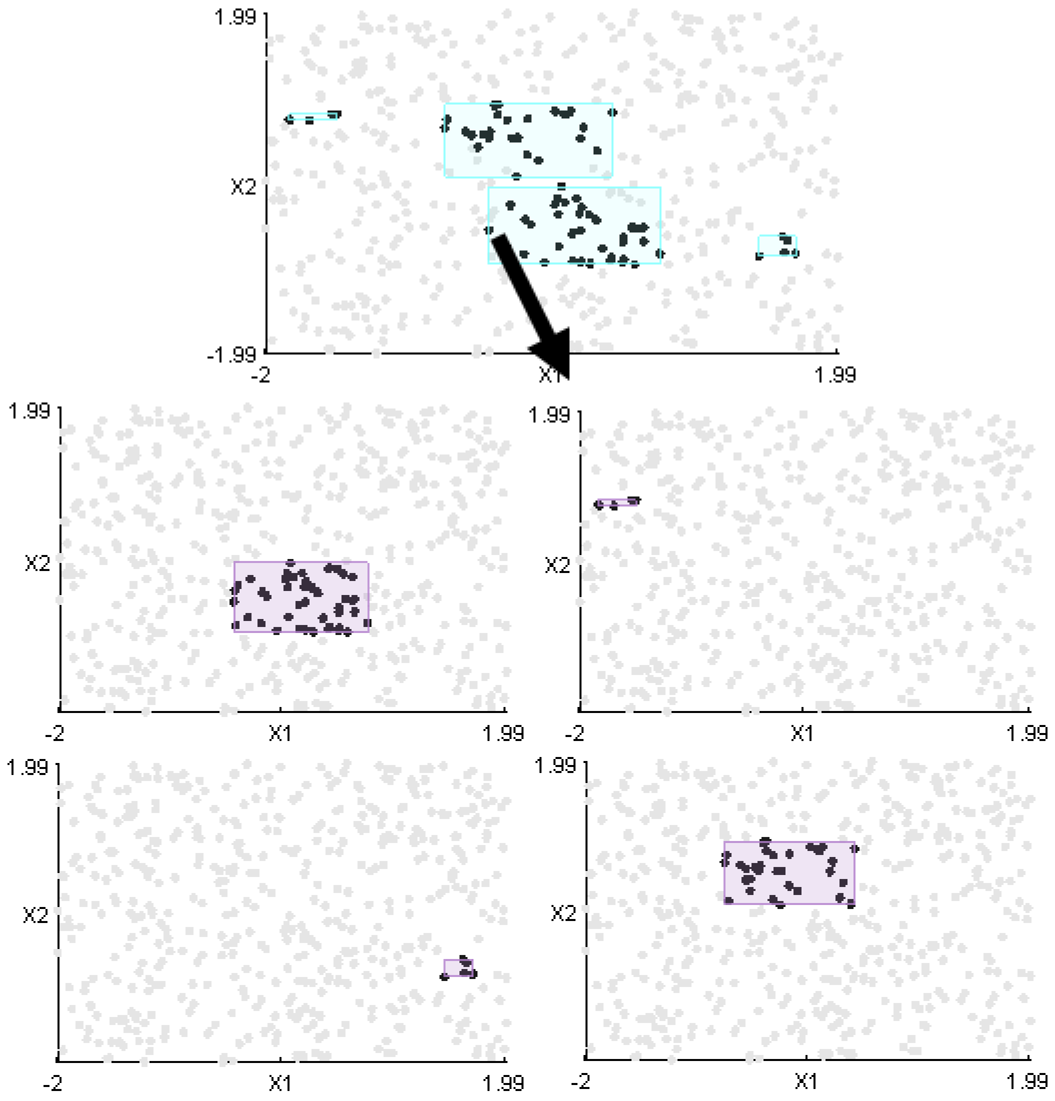


Figure 61 Top performing region of SC problem split into 4 design regions

A second method is to select points via brushing on a projection, and then click the “Add as region” button from the toolbar (F) in Figure 62. Design regions can also be set as search bounds using a *Set as Search Bounds* button.

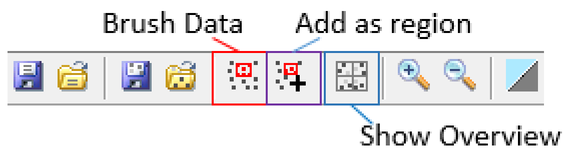


Figure 62 Toolbar for optimization framework

5.2. Summary

This chapter presented a software prototype that integrates the techniques described in previous chapter. It was shown that users may modify problems directly within the software. Next, the feasible space is plotted in parallel coordinates based on constraint sample data. Furthermore, methods to display convergence were also discussed such as tracking regions during optimization, highlighting iterations on convergence plot, or navigating to past iterations. Finally, ways to define design regions were explained. In the following chapter, emphasis is placed on determining which constraints may be suppressed or removed from the problem formulation. To do so, a novel technique incorporating data mining is introduced.

Chapter 6. Constraint Mining

As explained in Chapter 3, constraints are often defined to eliminate solutions that are not physically feasible or practical. Unfortunately, in some cases, designers may inadvertently specify constraints which are redundant. Common causes and consequences of superfluous constraints have been discussed by Karwan et al. [11]. In particular, they complicate the problem formulation, and may impact the performance of the optimization method. In this chapter, a new probabilistic method, including a priori association analysis, is proposed to identify and present constraint redundancies as rules to assist in problem formulation. The method has been accepted for publication in Engineering Optimization (pending minor revisions) [87].

6.1. Constraint mining overview

In general, the response of optimization constraints can be treated as Boolean variables: a one indicates a violation and a zero indicates satisfaction. From this data, relationships among constraints can be identified and presented to designers. This chapter first discusses past approaches related to redundant constraint identification with a focus on set-covering. Next, a new sequential process is presented. The method first checks if constraints co-occur using Jaccard similarity, before looking for implication rules in the remaining constraints with a priori association analysis [88,89]. Subsequently, constraints that do not co-occur, and are not implicit, are evaluated to see if they uniquely restrict the design space. Example problems, including the pressure vessel (PV) problem from Chapter 3, are tested and discussed.

6.2. Constraints in black-box engineering design

As explained in Chapter 3, in engineering design, inequality constraints are typically written as a vector of functions: $\boldsymbol{g}(\boldsymbol{x})$. In practice, $\boldsymbol{g}(\boldsymbol{x})$ may consist of nonlinear functions or even complex engineering simulations such as Finite Element Analysis (FEA) or Computational Fluid Dynamics (CFD). For such cases, it may be impossible to identify redundant constraints algebraically as done in [11,90–92]; no assumptions can be made about the properties of $\boldsymbol{g}$ (e.g. linearity, convexity, continuity etc.). Instead, like the objective function in previous chapters, constraints can also be treated as black-boxes. In this situation, random designs may be evaluated for each constraint, and the results may be summarized with 0 indicating a constraint satisfaction and 1 representing violation. From this basic information, constraints may be analyzed as a Boolean dataset.

6.3. Related redundancy identification methods

Redundant constraint identification has been the subject of extensive research since Boot's pioneering work in 1962 [93]. Although progress has been made, most work has focused on classifying and reducing constraints for linear programming (problems where the objectives and constraints are linear algebraic expressions). For example, Brearley et al.'s method [94] uses the coefficients of the constraints' terms to see if it is possible to satisfy them within the chosen design space. Telgen [11] developed a deterministic approach, similar to the simplex method in linear programming, using a minimum ratio test and simplex tableaus. Unfortunately, the majority of deterministic constraint reduction techniques require linear algebraic functions. A survey of redundancy checking for linear programming can be found in [90], with details in [11].

The set-covering approach, introduced by Boneh [95,96], is the most general method, and the most suitable for black-box optimization. Set-covering represents the feasibility of constraints for a given sample design $\boldsymbol{x^i}$, as a binary vector $\boldsymbol{e_i}$. If a constraint g_j is violated at the point $\boldsymbol{x^i}$, $e_{ij} \epsilon \boldsymbol{e_i}$ is 1. Otherwise, if the constraint is satisfied, e_{ij} is 0. For example, for two constraints g_1 and g_2, $\boldsymbol{e_i} = (0,1)$, if g_1 is satisfied and g_2 is violated for the design $\boldsymbol{x^i}$. If any constraint is violated for a design ($\sum \boldsymbol{e_i} \geq 1$), then $\boldsymbol{e_i} \in \boldsymbol{E}$. The main theorem of the set-covering approach states that if any design violates one or more constraints, at least one of those constraints is necessary. As notation, a vector $\boldsymbol{y}$, may be defined to summarize which constraints are necessary or redundant, based on the given data. Specifically, $y_j \epsilon \boldsymbol{y}$ is 1 if g_j is necessary, and y_j is 0 if g_j is redundant. With this notation, Boneh represents the main theorem of the set-covering approach as Eq. (25). In words, Eq. (25) states that for each observation $\boldsymbol{e_i \epsilon E}$, at least one of the violated constraints is non-redundant ($\boldsymbol{e_i} \cdot \boldsymbol{y} \geq 0$).

$$\boldsymbol{Ey} \geq (1,1,1 \ldots)^T \tag{25}$$

Given a sample of observations $\boldsymbol{E}$, Eq. (25) can be solved for $\boldsymbol{y}$ to identify redundant constraints. The solutions of $\boldsymbol{y}$ are not necessarily unique. Therefore, Boneh suggests pursuing the constraint subset which minimizes computational cost, and presents an algorithm to do so: the Set-Covering algorithm. The method in this chapter differs from set-covering in that its main concern is not finding irreducible sets of constraints (solutions of $\boldsymbol{y}$). Instead, the proposed method returns information regarding *why* constraints are redundant, which may provide insight about the formulated problem. Specifically, the chapter introduces methods to determine if constraints co-occur, are implicit with respect to another constraint, or are covered by other constraints. These relationships are defined and formalized.

6.4. Constraint redundancy rule definitions

According to Karwan et al.'s survey [11], the consensus is that a constraint is redundant if its removal has no effect on the feasible space. A more detailed taxonomy of redundancy types is also presented in [11]. In practice, information explaining relationships between constraints can be helpful for a designer who is trying to find errors in their problem formulation. Therefore, definitions are proposed in this work to not only indicate if a constraint is redundant, but also indicate *why*.

6.4.1. Redundant due to co-occurrence

Co-occurring constraints are constraints which are very often violated simultaneously. Although the constraints may be conceptually different, they have the same effect on the feasible region. This relationship means that any constraint C_k from a set of constraints ($\boldsymbol{C}$) can be chosen as a representative, while the others are suppressed or removed from the problem formulation. Co-occurrence is considered to be the most informative type of redundancy. If constraints *always* co-occur they are *duplicates*. A relaxed definition of co-occurrence is given in 6.5.1 based on Jaccard similarity.

Definition 1 (Redundant due to co-occurrence): Let $\mathbf{C}$ be a set of inequality constraints. If $\mathbf{C}$ co-occurs, then a constraint $C_k \in \boldsymbol{C}$ can be chosen as the representative of $\boldsymbol{C}$ and the remaining constraints $\mathbf{C}\backslash\{C_k\}$ are redundant.

6.4.2. Redundant due to implication

Redundancy due to implication means that a constraint is dominated by another constraint, within a given probability. Consider the data in Table 7. In the rows that P has a value of one, Q also has a value of one, but not vice versa. Therefore, we can write $P \rightarrow Q$. Another way to think of this is that P is dominated by Q; or if there is P, then there is Q; or P is redundant due to Q by implication. This rule is considered to be the second most informative type of rule, as co-occurrence automatically infers implication, but not vice-versa. Implication rules are generated efficiently using a priori association analysis, from data mining, as explained in 6.5.2.

Table 7 Implication example

P	Q
1	1
0	1
1	1
0	0

Definition 2 (Redundant due to implication): Let C_A and C_C be non-co-occurring inequality constraints. If $C_A \rightarrow C_C$, then C_A is redundant due to C_C by implication.

6.4.3. Redundant due to covering

If a constraint can be removed without affecting the feasible region, it is also redundant. To uniquely affect the feasible region, a case must exist where the constraint is violated while all others are satisfied. If no such case exists, the constraint is called redundant due to covering (i.e. C_k is *covered* by other constraints). Redundancy due to covering is equivalent to the general definition of redundancy in set-covering [96] and Boot's definition of a triviality [93].

Co-occurrence and implicitness is more informative to describe redundancies, as these rules include which constraint(s) make the constraint redundant (by co-occurrence or by association). The constraints which cover C_k, however, are not recorded (for computational simplicity). Consequently, covering is the least informative rule, and only examined after checking for co-occurrence and association.

Definition 3 (Redundant due to covering): Let $\boldsymbol{C_T}$ be the set of all constraints which are non-co-occurring and non-implicit. If no infeasible observation is made feasible by the removal of a constraint $C_k \in \boldsymbol{C_T}$, C_k is redundant.

6.5. The constraint mining method

An overview of the sequence for mining constraint relationships is described in Figure 63. Each step is described in detail in the following sections. The first four steps are general, and can be applied to rule mining for any Boolean dataset. The remaining steps are specific to the application of constraint analysis for engineering design.

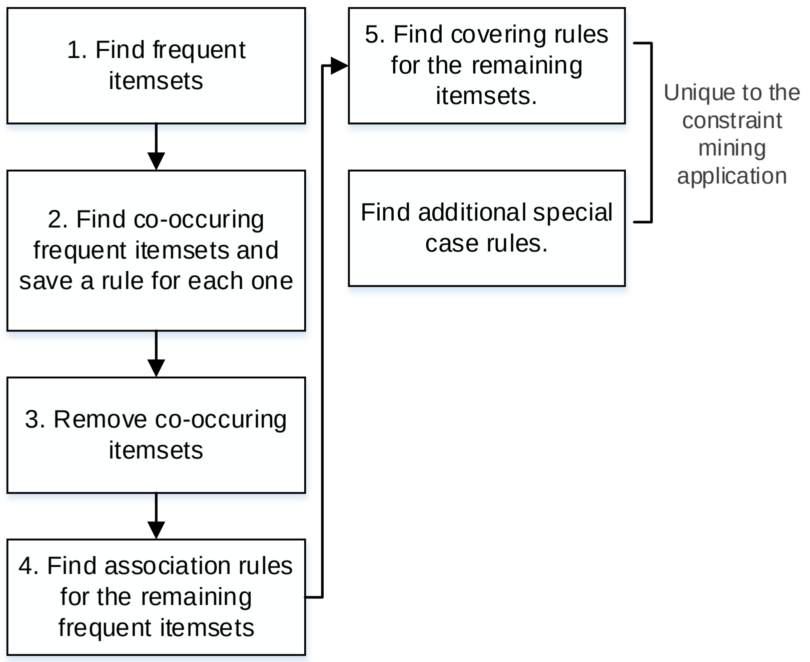

1. Find frequent itemsets
2. Find co-occuring frequent itemsets and save a rule for each one
3. Remove co-occuring itemsets
4. Find association rules for the remaining frequent itemsets
5. Find covering rules for the remaining itemsets.
Find additional special case rules.
Unique to the constraint mining application

Figure 63 Overview of the constraint mining method

6.5.1. Identifying frequent itemsets and co-occurring items

In association analysis literature, columns of Boolean matrices are referred to *items*. This stems from the original use of association analysis, to analyze consumer market basket data (e.g. *70% of people will buy butter if they buy bread*). Similarly, an *observation* refers to a row (e.g. the constraints results for a design $\boldsymbol{x^i}$). An item *occurs* if its value is 1. Meanwhile, a group of items is called an *itemset*. An example is given in Table 8 for five observations (designs) and six items (constraints). In design 1, for instance, the itemset $\{C_1, C_2, C_3, C_5, C_6\}$ occurs.

Table 8 Example binary representation of constraint violations (1: Violation)

Observation	C_1	C_2	C_3	C_4	C_5	C_6
1	1	1	1	0	1	1
2	1	1	1	0	0	1
3	0	0	1	0	1	1
4	1	0	0	0	0	0
5	0	1	0	1	0	0

Step 1 – Filter items by support and find frequent itemsets

As explained by Tan, the number of possible association rules for a dataset grows exponentially with its number of items (d): $nRules = 3^d - 2^{d+1} + 1$ [70]. For example, in Table 8, with 6 items, there 602 possible rules of the form: $\boldsymbol{C_A} \rightarrow \boldsymbol{C_C}$. Therefore, in Agrawal's a priori association analysis method, itemsets which have few occurrences are eliminated [88,89]. The occurrence an itemset $\boldsymbol{C}$ is quantified by $Support$ *s* as in Eq. (26):

$$s(\boldsymbol{C}): \frac{\sigma(\boldsymbol{C})}{N} \tag{26}$$

In Eq. (26), $\sigma(\boldsymbol{C})$ is the count of observations such that *all* items $\boldsymbol{C}$ occur, and N is the total number of observations. For example, from Table 8, the support of the itemset $\{C_1, C_2\}$ is s$(\{C_1, C_2\}) = \frac{2}{5}$. An itemset is considered ***frequent*** if $s(\boldsymbol{C}) \geq minsup$, where $minsup$ is a threshold chosen by the user (given in Section D2 of the framework). A low $minsup$ means rarer items will be considered, but also increases computational effort. Although frequency-based filtering is important for consumer market databases with thousands items, in most engineering problems the number of constraints is much lower, as constraints are defined for physical properties. Therefore, assuming the number of constraints is roughly ten or less, $minsup$ may be set to zero. In other words, all constraints can be considered frequent, even if they are rarely active, without significant computational cost. In 6.5.6, it is explained that support for individual constraints is an informative statistic on its own.

In the a priori algorithm, frequent itemsets are built from the bottom up, by combining only the frequent itemsets found *a priori*. For example if the items C_1 and C_2 are frequent, they form: $\{C_1, C_2\}$. The list of frequent itemsets then becomes: $\{\{C_1\}, \{C_2\}, \{C_1, C_2\}\}$. Support is anti-monotonic. Accordingly, only the frequent *k* sized sets are needed to generate the *k+1* sized sets. Efficient implementation details can be found in most introductory data mining books. A flowchart is presented in Figure 64. It is important to note that if $minsup = 0$, all of the possible itemsets are frequent.

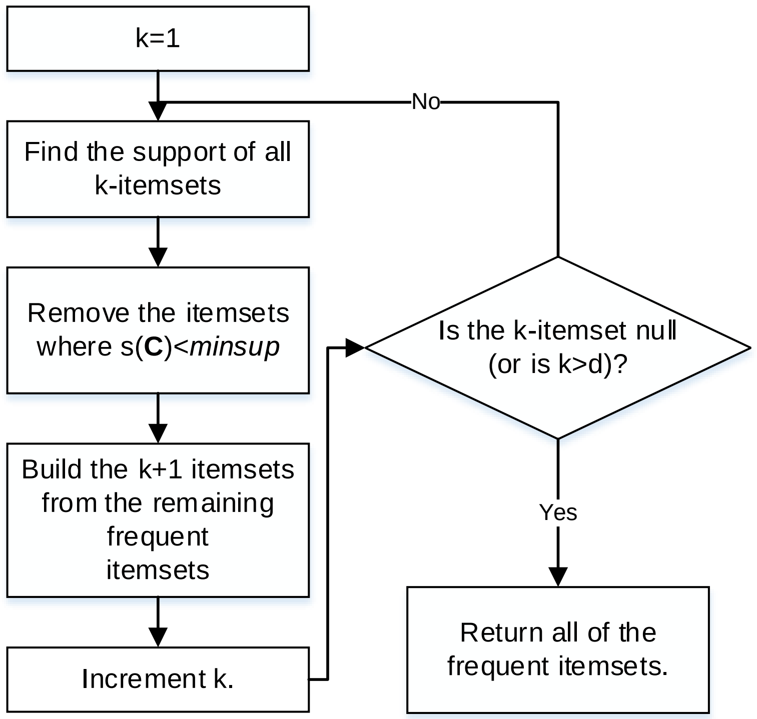

k=1
No
Find the support of all k-itemsets
Remove the itemsets where s(C)<minsup
Is the k-itemset null (or is k>d)?
Build the k+1 itemsets from the remaining frequent itemsets
Yes
Increment k.
Return all of the frequent itemsets.

Figure 64 Generation of a frequent itemset of size k in the a priori method

Step 2 - Use the Jaccard measure to identify co-occurring sets

Once frequent itemsets are generated, they can each be checked for co-occurrence. This stage is a contribution of this work, and not part of Agrawal's standard a priori method. Jaccard (j) measures the conditional probability of *all* items occurring, if it is known that *any* occurs. Mathematically, $j(\{a,b,c\}) = P(a\wedge b\wedge c \mid a\vee b \vee c)$. Jaccard for an itemset $\boldsymbol{C}$ can be written as in Eq. (27):

$$j(\boldsymbol{C}) = \frac{\sigma(\boldsymbol{C})}{\gamma(\boldsymbol{C})} \tag{27}$$

In Eq. (27), γ is the count of observations where *any* item in $\boldsymbol{C}$ occurs. In the proposed method, if $j(\boldsymbol{C})$ is greater than a co-occurrence threshold $minjacc,$ the itemset is called co-occurring or bundled. Although the bundle pattern has been discussed before [97], it wasn't defined using Jaccard. The meaning of *bundle*, in this work, is conceptually similar to [97] in that it indicates co-occurrence, but differs mathematically.

Definition 4 (Co-occurrence/Bundling) *An itemset $\boldsymbol{C}$ co-occurrs (is bundled) if j($\boldsymbol{C}$) $\geq minjacc$. A bundled set of constraints is denoted by $\boldsymbol{C_B}$.*

It is important to emphasize that $j(\boldsymbol{C})$ is fundamentally different from $s(\boldsymbol{C})$. Support (s) measures the likelihood of a group of constraints to be violated. Jaccard (j) measures the similarity of constraints in a group. For example, in Table 8 $s(\{C_3, C_6\})$ is $\frac{3}{5} = 0.6$; while, $j(\{C_3, C_6\})$ is $\frac{3}{3} = 1$. If $j = 1$ for a set, the constraints are *duplicates*. By reducing $minjacc$ from 1 to a slightly lower value, constraints that are nearly duplicates are identified.

Step 3- Remove co-occurring frequent itemsets

If a set of items is co-occurring, a rule is saved for those items (e.g. "*Constraints C3 and C6 co-occur.*"). Furthermore, it can be shown that the subsets of co-occurring sets also co-occur, based on the fact that j is anti-monotonic with the addition of items.

Theorem 1 Given a co-occurring itemset $\mathbf{C_B}$ and a subset $\mathbf{C_B}^{-} \subseteq \mathbf{C_B}$: j($\mathbf{C_B}^{-}$) $\geq j(\mathbf{C_B})$. Therefore, if $\mathbf{C_B}$ is co-occurring, $\mathbf{C_B}^{-}$ is co-occurring.

Proof:

$$\sigma(\mathbf{C_B}) \leq \sigma(\mathbf{C_B}^{-})\ by\ the\ anti-monotonic\ property\ of\ support.$$

$$\gamma(\mathbf{C_B}^{-}) \leq \gamma(\mathbf{C_B})\ by\ the\ definitions\ \gamma(\mathbf{C_B}) = |any(\mathbf{C_B})|\ and\ \ \mathbf{C_B}^{-} \subseteq \mathbf{C_B}.$$

$$\therefore minjacc \leq \frac{\sigma(\mathbf{C_B})}{\gamma(\mathbf{C_B})} \leq \frac{\sigma(\mathbf{C_B}^{-})}{\gamma(\mathbf{C_B}^{-})}\ \blacksquare$$

To remove co-occuring subsets efficiently, the algorithm in Figure 65 can be used. Starting with the largest frequent itemset (of size k_{max}), $j(\boldsymbol{C})$ is computed. If $\boldsymbol{C}$ is co-occurring, a rule is saved, and all of its subsets (including $\boldsymbol{C}$ itself) are removed. Once each set of size *k* has been evaluated, *k* is decremented. The process stops if all sets of size *k* co-occur, or if $k = 1$. In other words, the method first checks if the largest frequent group of items co-occurs, then moves to smaller and smaller item groups (until single item groups remain).

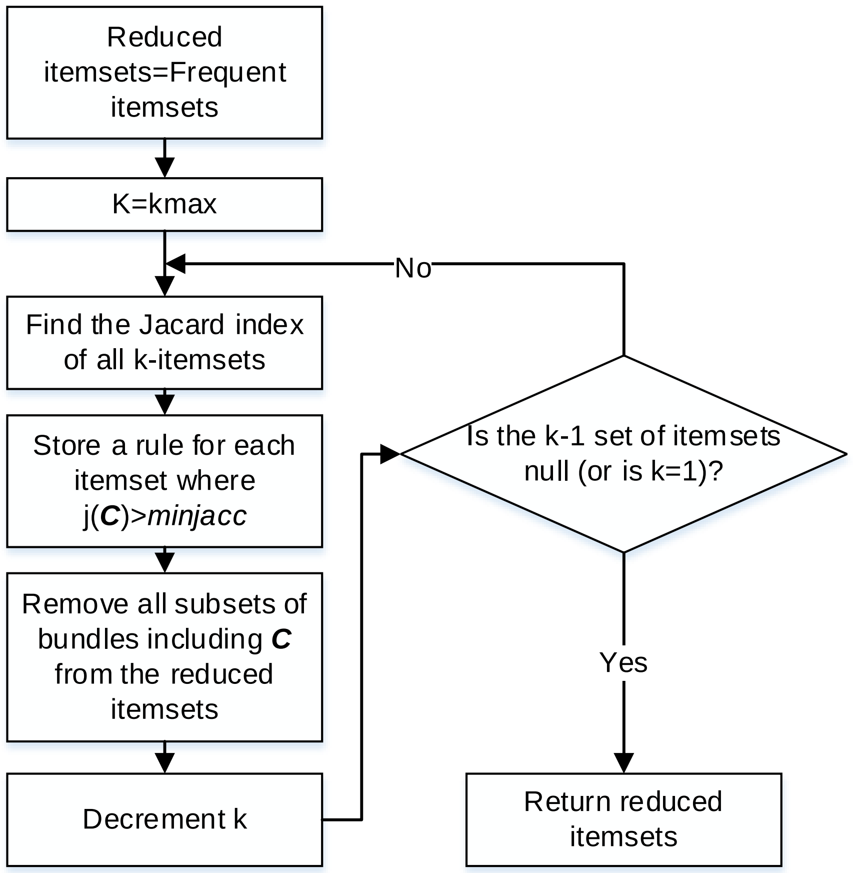

Reduced itemsets=Frequent itemsets
K=kmax
No
Find the Jacard index of all k-itemsets
Store a rule for each itemset where j(C)>minjacc
Remove all subsets of bundles including C from the reduced itemsets
Decrement k
Is the k-1 set of itemsets null (or is k=1)?
Yes
Return reduced itemsets

Figure 65 Identification of bundles and itemset reduction

6.5.2. Generating association rules (Step 4)

Association rules (rules of the form $\boldsymbol{C_A} \rightarrow \boldsymbol{C_c}$) are a well-researched area in data mining popularized by Agrawal [88,89]. Since their introduction, a significant amount of research has been published that presents efficient algorithms [98,99], new patterns [100–102], compact representation of itemsets [103] and new measures of rule interestingness [104]. Tan provides an excellent review of the field in the bibliographical notes of [70]. Applications have also expanded beyond market baskets to finding protein interactions [105] and associations between carbon levels and ocean climates [106]. This work is the first time the method has been applied to optimization constraints.

Association rules are generated and evaluated by *confidence* in Agrawal's algorithm. Confidence $c(\boldsymbol{C_A} \rightarrow \boldsymbol{C_c})$ represents the conditional probability of $\boldsymbol{C_C}$, given $\boldsymbol{C_A}$

(based on the sample data). It is defined by Eq. (28), where $\boldsymbol{C_A}$ and $\boldsymbol{C_c}$ are frequent itemsets:

$$c(\,\boldsymbol{C_A} \rightarrow \boldsymbol{C_c}) \colon \frac{\sigma(\boldsymbol{C_A} \cup \boldsymbol{C_c}\,)}{\sigma(\boldsymbol{C_A})} \tag{28}$$

In Eq. (28), $\boldsymbol{C_A}$ is the *antecedent* set or simply the left-hand-side (LHS). Meanwhile, $\boldsymbol{C_c}$ is the *consequent* set or the right-hand-side (RHS). Rules are generated by testing confidence against a threshold: $minconf$. The appropriate choice for $minconf$, like $minsup$ or $minjacc$, also depends on the application. For example, in design engineering, it is important to avoid suppressing constraints which are not truly redundant. Therefore, as a rule of thumb, $minconf$ should be nearly or exactly 1. Furthermore, it is enforced that $minjacc \geq minconf$, by definition. If an itemset did not meet the threshold to be co-occurring, then it may still generate association rules. Conversely, if the set co-occurs, then it was accounted for in Step 3 and will not be further analyzed for associations.

In Agrawal's method, association rules are generated for each individual itemset. The process starts by assigning an individual item as the consequent, while the remaining items in the set form the antecedent. Each item is selected as the consequent once, forming a set of single item RHS rules. The RHS of rules with high confidence ($\geq minconf$) are then combined to form two item RHS rules. Consequents which generate high confidence rules continue to merge and form new rules until all high confidence rules have been generated for the itemset.

Confidence has useful properties. In particular, it is anti-monotonic as items are shifted from the LHS to the RHS. For instance, in the set $\{C_1, C_2, C_3\}$, the rule $\{C_1, C_2\} \rightarrow C_3$ will have higher (or equal) confidence when compared to $\{C_1\} \rightarrow \{C_2, C_3\}$ or $\{C_2\} \rightarrow \{C_1, C_3\}$. This is explained in detail in [70]. Similarly, if two rules have the same LHS; but, one RHS is a subset of the other's, the rule with the smaller RHS will have higher confidence. For example, $c(\{C_1\} \rightarrow \{C_2\}) \geq c(\{C_1\} \rightarrow \{C_2, C_3\}\,)$. A simple proof is provided below for completeness.

Theorem 2. Given an antecedent $\boldsymbol{C_A}$, consequent $\mathbf{C}_{\boldsymbol{C}}$, and subset $\mathbf{C}_{\boldsymbol{C}}^{-} \subseteq \mathbf{C}_{\boldsymbol{C}}$**:**

$$c(\boldsymbol{C_A} \rightarrow \mathbf{C}_{\boldsymbol{C}}) \geq c(\boldsymbol{C_A} \rightarrow \mathbf{C}_{\boldsymbol{C}}{}^{-})$$

Proof:

$$c(\boldsymbol{C_A} \rightarrow \mathbf{C}_{\boldsymbol{C}}) = \frac{\sigma(\boldsymbol{C_A} \cup \boldsymbol{C_c}\,)}{\sigma(\boldsymbol{C_A})} \leq \frac{\sigma(\boldsymbol{C_A} \cup \mathbf{C}_{\boldsymbol{C}}{}^{-}\,)}{\sigma(\boldsymbol{C_A})} \;\; multiply\ both\ sides\ by\ \sigma(\boldsymbol{C_A}),$$

$$\sigma(\boldsymbol{C_A} \cup \boldsymbol{C_c}\,) \leq \sigma(\boldsymbol{C_A} \cup \mathbf{C}_{\boldsymbol{C}}{}^{-}\,)\ which\ is\ true\ by\ the\ anti-monotonic\ property\ of\ \sigma \;\; \blacksquare$$

6.5.3. A brief discussion regarding the number of rules and the benefit of co-occurring sets

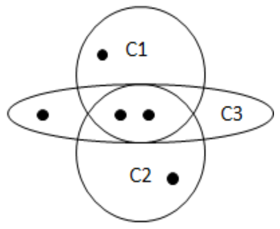
The support-confidence framework reduces the number of possible implications by eliminating rare items or weak rules respectively. Unfortunately, datasets with similar high support items still creates many rules. This is a fundamental drawback of the standard a priori algorithm. For example, for items C_1, C_2 and C_3 the corresponding frequent itemsets may be $\{\,\{C_1\},\{C_2\},\{C_3\},\{C_1C_2\},\{C_1C_3\},\{C_2C_3\},\{C_1C_2C_3\}\,\}$. In total, there are 18 possible implication rules for these three items. Meanwhile, a single rule stating that "$\{C_1,C_2,C_3\}$ co-occur" conveys the same information. In fact, this rule is not only more compact, it is simpler to understand. This is why co-occurring sets are considered more informative than associations, and are tested first.

6.5.4. Further rule reductions when considering item removal

Restricting association rules to single item antecedents

This section shows that for constraint redundancies, association rules are only needed for one item antecedents. Consider the example in Table 9. Having $\{C_1,C_2\}$ as the LHS forms a stronger implication of C_3 than C_1 or C_2 individually. The rule is suggesting, in words, that if there is a violation in both C_1 and C_2, then C_3 is also violated, which is obvious from the Boolean values in Table 9.

Table 9 Example of increasing confidence with additional items in the antecedent

C_1	C_2	C_3	$Conf(\{C_1, C_2\} \rightarrow C_3)$	$Conf$ $(C_1 \rightarrow C_3)$	$Conf$ $(C_2 \rightarrow C_3)$	
1	1	1	$\frac{\sigma(\{C_1,C_2,C_3\})}{\sigma(\{C_1,C_2\})} = \frac{2}{2} = 100\%$	$\frac{\sigma(\{C_1,C_3\})}{\sigma(C_1)} = \frac{2}{3} = 67\%$	$\frac{\sigma(\{C_2,C_3\})}{\sigma(C_2)} = \frac{2}{3} = 67\%$	
1	1	1				
0	0	1				
1	0	0				
0	1	0				

In reality, $\{C_1, C_2\} \rightarrow C_3$ states that the intersecting region of $\{C_1, C_2\}$ is redundant due to C_3. Yet, if the goal is to remove constraints, it is not important if $\{C_1, C_2\}$ implies C_3. Neither C_1 nor C_2 can be removed, as C_1 and C_2 restrict other areas as well. In order to remove either C_1 or C_2 (due to C_3), individual rules $C_1 \rightarrow C_3$ or $C_2 \rightarrow C_3$ are required. In this case, the confidence of these rules is only 67%. Thus, neither C_1 nor C_2 is redundant due to C_3. This example shows that although antecedents with more than one item may form strong rules, the rules cannot be acted on, and do not need to be considered.

Restricting association rules to single item consequents

Another way to limit the number of rules is to limit consequents to contain exactly one item. Assume that an implication rule is of high confidence with a single item LHS (C_A) and multi-item RHS ($\boldsymbol{C_C}$). All single item RHS rules $C_A \rightarrow C_C$, where $C_c \in \boldsymbol{C_c}$, are also of high confidence, by Theorem 2. For example, if $\{C_1 \rightarrow C_2C_3\}$ exceeds the confidence threshold, then $\{C_1 \rightarrow C_2\}$ and $\{C_1 \rightarrow C_3\}$ do as well. Furthermore, the sets $\{C_1, C_2\}$ and $\{C_1, C_3\}$, that generate $\{C_1 \rightarrow C_2\}$ and $\{C_1 \rightarrow C_3\}$ (respectively), are guaranteed to exist in the frequent itemset if $\{C_1, C_2, C_3\}$ exists, due to the anti-monotonic property of support. Therefore, if $\{C_1 \rightarrow C_2C_3\}$ exists with high confidence, then the rules $\{C_1 \rightarrow C_2\}$ and $\{C_1 \rightarrow C_3\}$ will also exist with high confidence.

In terms of constraint removal, the intuitive interpretation is that if a constraint is redundant due to the area where multiple constraints intersect, it is also redundant due to each intersecting constraint. This fact, combined with the limitation to single item antecedents, limits association rules to item *pairs* (which do not co-occur). Only considering pairs in the association rule generation phase greatly improves efficiency, and limits the number of rules returned.

6.5.5. Finding covered constraints (Step 5)

The next step is to find remaining redundancies for constraints that do not co-occur and are not implicit. This can be accomplished by looking for individual constraints which are never violated alone. By Boneh's main theorem [95], if an observation violates only one constraint, that constraint is absolutely necessary. Conversely, if a constraint is never violated alone, it has no unique effect on the feasible space and is redundant. Consider the case in Table 10. The union of violations in C_1 and C_3 covers the observations violated C_2. As shown by the observations in grey, if C_2were removed from set of constraints, the feasible points (with no violations) would remain unchanged for the given data. This makes C_2 redundant (by covering).

Determining if a constraint is covered is fundamentally different than identifying co-occurrence or implication. Namely, to check covering, additional scans of the data are needed. However, note that observations do not need to be counted as when checked for co-occurrence or implication. Therefore, checking if a constraint is covered only requires the set of unique observations (as in the set-covering approach [95]). This reduced set of observations is shown in Table 11, generated from Table 10 by eliminating the repetitions.

Covering is only checked for constraints which are not already explained by co-occurrence or implication ($\boldsymbol{C_T}$). A simple scan to find a case where C_k is violated alone is executed for each $C_k \in \boldsymbol{C_T}$ (using the reduced observations). If no such case exists, the constraint does not uniquely impact the overall feasibility of the problem, and a rule is saved. In Table 11, there is no case where C_2 violation occurs alone. Thus, C_2 is covered.

Table 10 Example of covered constraints

C_1	C_2	C_3
0	0	0
1	0	0
1	1	0
1	1	0
0	1	1
0	1	1
0	0	1

Table 11 Example of covered constraints (reduced observation set)

C_1	C_2	C_3
0	0	0
1	0	0
1	1	0
0	1	1
0	0	1

6.5.6. Additional information from constraint mining

In addition to finding bundles, association rules, and covered constraints two more types of information are reported: infrequent constraints and infeasible problems.

Infrequent constraints

It is often useful for the designer to understand and review constraints which are mostly passive, especially if they are costly to evaluate during optimization. Infrequent constraints are those which are rarely (or never) violated in the sample of observations (as quantified by support). By tuning $minsup$, the restrictiveness of constraints can be easily identified.

Definition 5 (Infrequent constraint) An individual constraint C_i is infrequent if $s(C_i) \leq minsup$.

Infeasible problems

An infeasible problem does not contain any feasible area. If a problem is infeasible, it may require re-formulation (e.g. making a restrictive constraint an objective) or an optimization algorithm that specializes in highly constrained problems. To search for overall feasibility, the *any* operator (γ) is used on each observation $\boldsymbol{o_i}$ in the unique set of observations. If any $\gamma(\boldsymbol{o_i})$ is zero (meaning $\boldsymbol{o_i} = \mathbf{0}$), a feasible point exists, the search is stopped, and the problem is considered feasible. If no feasible point exists, the problem is deemed infeasible.

Definition 6 (Infeasible problem) A problem is infeasible if there is no observation $\boldsymbol{o_i}$ in the unique set of observations such that $\boldsymbol{o_i}=\mathbf{0}$.

6.6. Collecting data and parameters

6.6.1. Collecting constraint data

To collect data for mining, constraint checks are randomly generated in an iterative fashion. The maximum number of iterations (nI) and the maximum number of constraint checks (nT) are given (in Section E2 of the layout). The number of constraint check samples for each iteration (nS) is as in Eq. (29):

$$nS = \frac{nT}{nI} \tag{29}$$

The parameter nT should be chosen based on the computational intensity, number of variables and number of constraints. For problems with computationally inexpensive constraints, thousands of samples can be tested. For computationally expensive constraints, fewer may be budgeted. Each iteration, nS random samples are evaluated by $\boldsymbol{g}(\boldsymbol{x})$, and the result is added to the table of Boolean observations. Rules are then mined using the method described in this paper. The stopping criteria for the iterative process are a number of iterations (I_s) have occurred without a change in rules, or that a maximum number of total observations have been sampled. Figure 66 shows the data generation process, where nI_s is the maximum number of iterations that may occur without change before stopping.

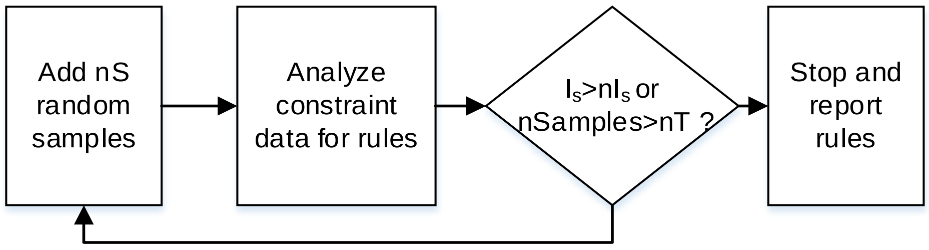

Add nS random samples
Analyze constraint data for rules
I_s>nI_s or nSamples>nT ?
Stop and report rules

Figure 66 Data generation flowchart

It is important to note that there is no theoretical guarantee on the number of observations required to establish correct rules. In fact, even if the conditional probability of a rule is 100%, based on the limited sample data, it may change if additional data is added. This is an inherent limitation of probabilistic methods. Therefore, if constraints are suppressed based on the rules found, it is suggested to confirm that the final solution is feasible by testing it against all constraints (including those which were suppressed). The data may also be visualized as shown in Chapter 5.

6.7. Examples and Results

6.7.1. Two-Variable Mathematical Examples

Problem Definition

In this section, constraint mining is tested on two-dimensional mathematical examples which can be easily visualized as in Figure 67. The arrows in Figure 67 indicate the direction of feasibility. The first problem has seven constraints, including one nonlinear constraint. The second problem is from Telgen's chapter in [11] (p.57), including 8 linear constraints. The problems are defined algebraically in Table 12 and Table 13, respectively. In Example 1, it is clear that $g3$ and $g7$ are multiples of one another. Furthermore, $g2$ is a tighter restriction than $g5$. In Example 2, the patterns are less obvious from the equations, but can be seen visually in Figure 67. Namely, $g2$ has no influence on the feasible space, due to the combination of $g6$ and $g3$.

Table 12 Constraint mining Example 1

$$\begin{aligned} g1:&\quad -x1 - x2 \le 0 \\ g2:&\quad x1 + x2 \le 5 \\ g3:&\quad x1 \le 3 \\ g4:&\quad -x2 \le 0 \\ g5:&\quad x1 + x2 \le 6 \\ g6:&\quad -x1 + x2^2 \le 0 \\ g7:&\quad 2(x1) \le 6 \end{aligned}$$

Table 13 Constraint mining Example 2

$$\begin{aligned} g1:&\quad x1 - x2 \le 2 \\ g2:&\quad 2(x1) + x2 \le 7 \\ g3:&\quad x1 \le 2 \end{aligned}$$

$$g4: \quad -x1 + 2(x2) \leq 4$$
$$g5: \quad 2(x2) \leq 5$$
$$g6: \quad x1 + x2 \leq 4$$
$$g7: \quad -x1 \leq 0$$
$$g8: \quad -x2 \leq 0$$

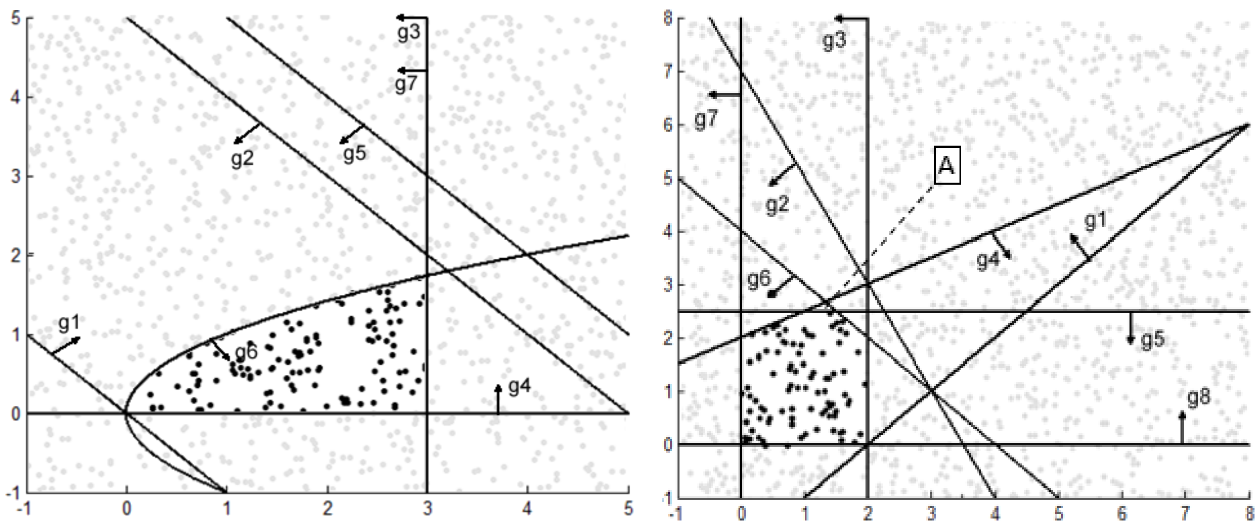


Figure 67 Example 1 (left) with 1200 samples and Example 2 (right) with 1600 samples (dark is feasible).

Constraint mining results

The results from constraint mining are presented in Table 14 and Table 15. The settings assume that a single contradictory case is sufficient to discredit a rule; therefore, $minsupp = 0$ and $minjacc = minconf = 1$. In Example 1, 1200 samples were tested. In Example 2, 1600 were tested. The number of iterations without a rule change (nIs) was set to 5, and nT = was set to 2000 with $nI = 10$ $(nSi = 200)$. The results from Table 14 are correct by inspection of Figure 67.

The results from Table 15 illustrate the probabilistic nature of the method. Although the first two rules are correct, the third rule is false. Indeed, there is an area (**A** in Figure 67) that is uniquely restricted by $g5$, meaning $g5$ is not covered. Unfortunately, no sample was generated in that region during random sampling. Consequently, from the constraint data, $g5$ has no case where it is violated and others $(g6\,\&\,g4)$ are not.

Thus, it is covered based on the collected observations. This illustrates the pitfall of using a probabilistic method.

Table 14 Example 1 Rules

Rule	Rule Conditional Probability	Rule Type
‘g3 and g7 co-occur’	100%	Co-occurrence
‘g5 is redundant to g2 by implication.’	100%	Implication
‘g1 is covered due to the union of other constraints’	100%	Covering

Table 15 Example 2 Rules

Rule	Rule Conditional Probability	Rule Type
‘g1 is covered due to the union of other constraints’	100%	Covering
‘g2 is covered due to the union of other constraints’	100%	Covering
‘g5 is covered due to the union of other constraints’	100%	Covering

6.7.2. Pressure Vessel Design Example

The PV problem, from Chapter 3, was also tested using constraint mining. For this problem, 1200 samples were generated, with $minsup = 0.1$, $minjacc = minconf = 0.9$. The data generation parameters are the same as in the previous examples with the exception that nT was raised to 4000, to account for the larger design space of 4 variables. The resulting list of rules is shown in Table 10.

Table 10: Rules for continuous pressure vessel design optimization

Rule	Rule Conditional Probability	Rule Type
'Violations in g2 are redundant due to g1 by implication.’	99.8%	Implication

Surprisingly, it was found that violations in g2 imply g1 in ~99% of samples. Therefore, constraint g2 could be suppressed (with the risk that the optimum falls into the 0.2% of cases where the rule is false). Optimization was subsequently performed with and without g2 using TR-MPS. The best design found in both cases is $f^* = 7006.8$, at $R^* = 51.8$”, $T_s^* = 1$", $L^* = 84.6$" and $T_h^* = 0.625$". This suggests that g2, really has no effect at the optimum.

To verify association rules beyond 2D, the underlying constraint check data can be visualized as in Chapter 5. The constraints g1 and g2 are shown below as scatterplot matrices. It is clear that the restriction of g1, covers the same points as g2 but is stricter on the variable R.

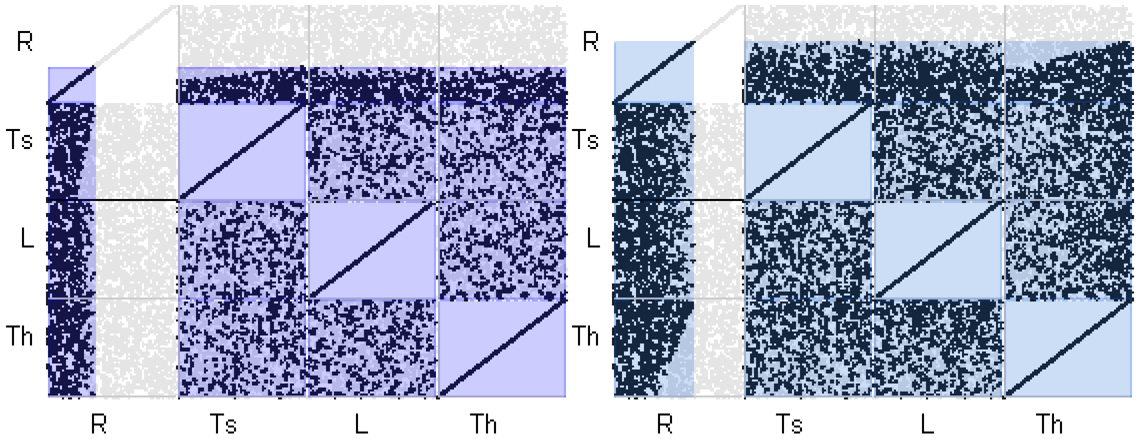


Figure 68 Constraints: g1 (left) and g2 (right) of the PV problem (dark is feasible).

6.8. Summary

This chapter presented a systematic method to find redundant constraint groups in black-box optimization problems. Association analysis, from data mining, was applied for constraint redundancy identification, which is a new application for the method. Rather than directly applying association analysis, this work developed a sequential method and theorems for constraint redundancy identification. In specific:

1. A new method is proposed to find constraints that co-occur, using Jaccard similarity, before performing association analysis on the remaining frequent itemsets. This prevents the generation of many unnecessary association rules.

2. Additional limitations were added to a priori rule generation. In particular, it was shown that implication rules with more than two items do not provide additional information for constraint redundancy identification.

3. Additional redundancies (due to covering) are checked on the remaining itemsets using a reduced set of observations as in the set-covering approach.

4. The result of the proposed method is a set of readable rules that the user may choose to act on, as opposed to a reduced set of constraints as in other redundancy identification methods.

The method was first applied to mathematical problems. It was found that the rules summarize the relationship among constraints. However, it was also pointed out that, due to the method's probabilistic nature, the accuracy of the rules depends on the sample points. An incorrect rule was mined due to a lack of sampling in a particular region for one test example. The method was then applied to the PV problem. Surprisingly, it was found that a design constraint for this benchmark problem is likely redundant. The rules were validated using multivariate visualization. Although the presented method cannot guarantee correctness, constraint rules provide design engineers with new information of how their constraints restrict the design space. Ultimately, this provides a starting point for comparing constraints visually, and may lead to improved problem formulations. The following chapter focuses on a case-study of applying the developed interactive optimization framework.

Chapter 7. Robotic Automotive Assembly Station Optimization – Case Study

In this chapter, the black-box optimization visualization framework is applied to the optimization of automotive assembly station planning. First, the specific optimization goal is described in standard form. Next, the results from an application specific tool are compared to the results using the general optimization visualization framework. Although the application specific visualization is more intuitive; this application illustrates how a general framework may be applied to a real industrial application in the absence of additional visualization.

7.1. Robotic automotive assembly optimization problem description

In robotic automotive assembly, fixtures and clamps hold sheet metal parts in position while welding as shown in Figure 69. These fixtures contain locating pins which slide into holes and slots (which restrict the parts from translating or rotating) [107]. The holes and slots are the other half of the locating system called “locators”. In practice, parts are also clamped down (at various positions on the part). Clamps are omitted in the following examples for simplicity, but may also be modeled and optimized using the same approach. Furthermore, in this analysis parts are assumed to be rigid.

To move the parts on and off the fixtures (by robotic arms), there is a slight clearance between the pin and the hole or slot. Unfortunately, a clearance also means that the parts may “jiggle” before being clamped, creating variance in part assemblies. Furthermore, the location and sizes of holes and slots will vary slightly each time a blank is manufactured, according to design specifications, typically on the order of 0.1mm for large automotive parts. Overall, it is desirable to place the locators in a configuration such that the variance in part assemblies is minimized. This can be achieved by simulating many assemblies in CAD (i.e. DCS Analyst), with differing locating schemes, and comparing the variance of alternative fixture configurations.

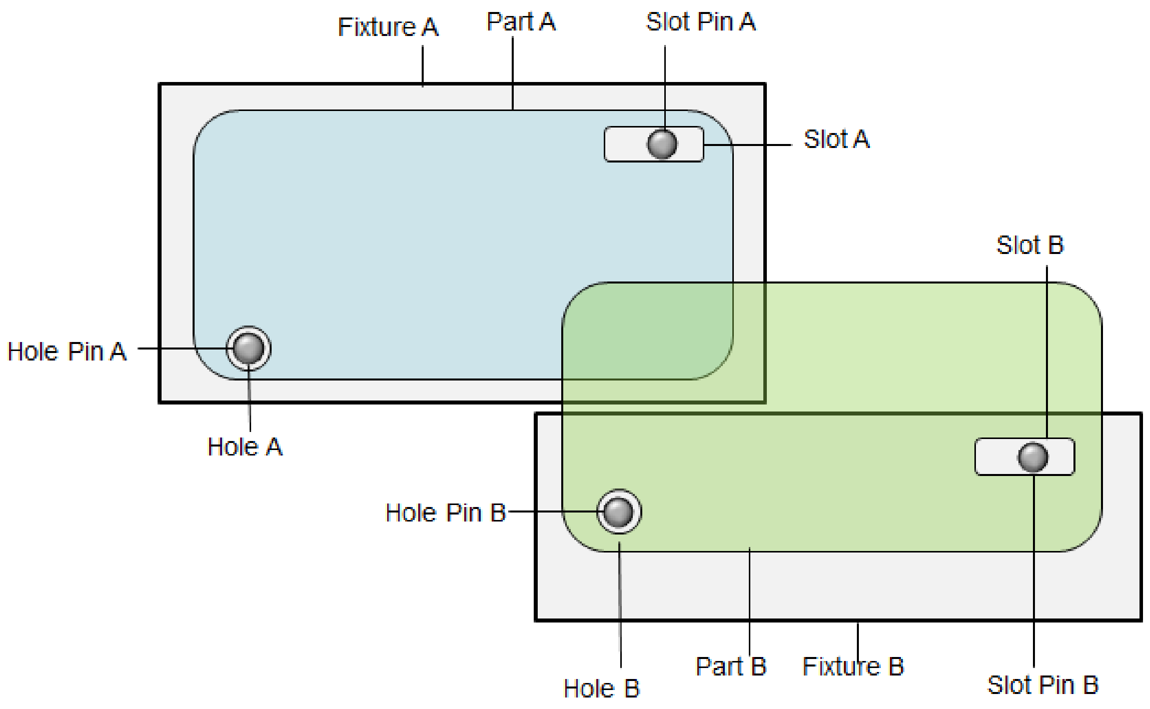


Figure 69 Possible locator configurations for two plates

Two methods to quantify the overall variance of assembly simulations are shown in Figure 70. In Option 1, the variance is computed on a single corner-to-corner distance measure. In Option 2, the distance from nominal is measured at multiple key product characteristics (KPC) points on the parts which are then squared and aggregated (as in Eq. (2) with equal weights) as an overall measure of assembly variance. Key product characteristic points are points which must be near their nominal position to ensure the safety and quality of the vehicle.

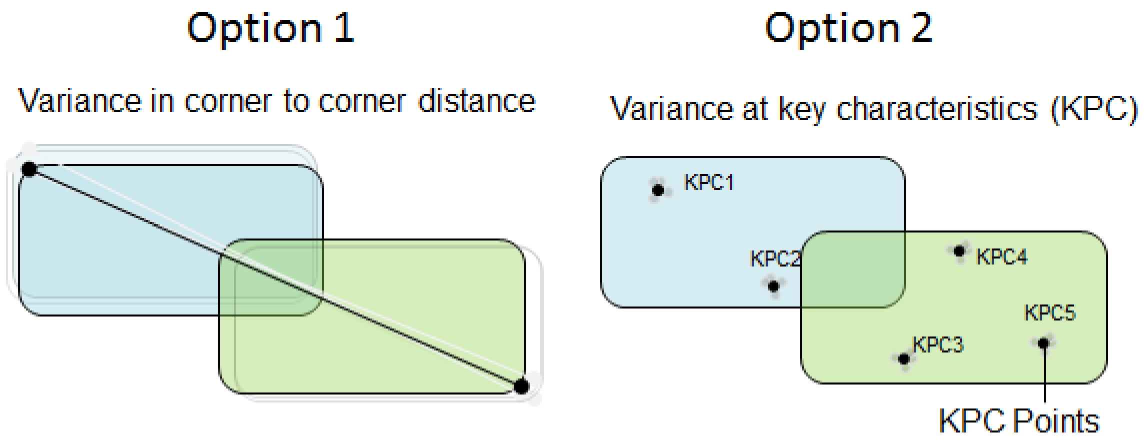

Option 1
Variance in corner to corner distance
Option 2
Variance at key characteristics (KPC)
KPC1
KPC2
KPC4
KPC5
KPC3
KPC Points

Figure 70 Possible measures to quantify variance

7.2. Optimization problem formulation

The objective of optimization is to minimize the variance in part assemblies by changing the placement of locators on fixtures. Figure 71 shows the coordinate system for parts and locators. For a three part assembly, with two-locators per part (hole and slot), there are 12 variables in total (X-Z locations for each locator). In reality, there is also a Y (out of page direction) coordinate to consider. However, the Y coordinate may be automatically computed based on the X-Z location and can be considered as a dependent variable.

In this problem, the design space is specified by a discrete list of possible positions for each locator on the part. The design variables are offsets for each X and Z coordinate of the locator, starting from an initial location. The objective function is to minimize the sum-of-squares of 14-KPC Point variances (variance is computed over 2000 assemblies for each configuration, simulated by DCS Analyst). Although there are constraints in practice (e.g. avoiding overlapping locators), they are omitted for this simplified example, creating an unconstrained optimization problem. The objective function is explicitly written as in Eq. (30).

$$\min f(\boldsymbol{x}) = \sum_{i=1}^{14} V_i^2 \tag{30}$$

In Eq. (30), V is proportional to the variance of the Euclidean distance of a KPC point (*i*) from its nominal position. Specifically, V is 6 times the standard deviation of 2000 assembly simulations in DCS Analyst. The variables $x_1 \dots x_{12}$ are the x-z offsets of locators. Figure 71 shows a simplified model of the assembly optimized. The assembly consists of three parts (labeled A, B and C) forming a structural sub-assembly of the side-panel for a larger vehicle.

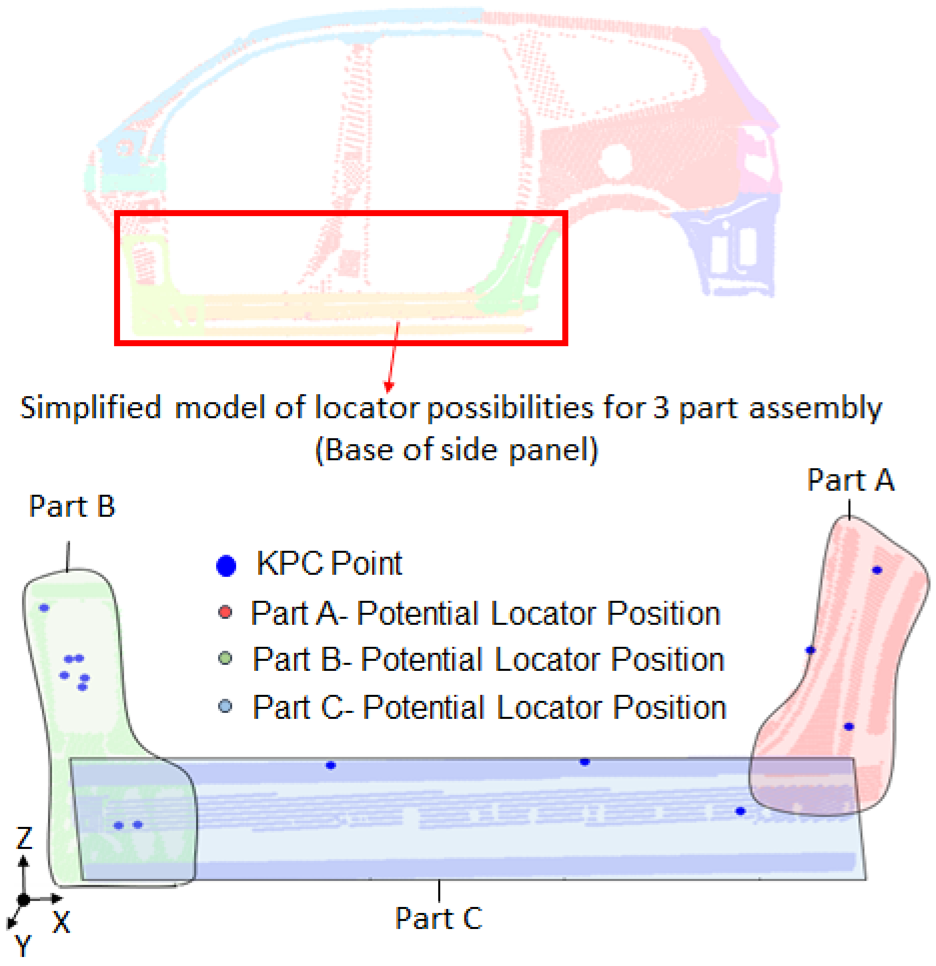

Simplified model of locator possibilities for 3 part assembly
(Base of side panel)
Part B
Part A
● KPC Point
● Part A- Potential Locator Position
● Part B- Potential Locator Position
● Part C- Potential Locator Position
Z
X
Y
Part C

Figure 71 Base of automotive side panel with potential locator positions

The initial locator positions (with 0 offsets) are as follows in Figure 72. The objective value with this configuration is considered to be the "baseline" (i.e. the minimum performance that is acceptable) with a value of $0.32mm^2$.

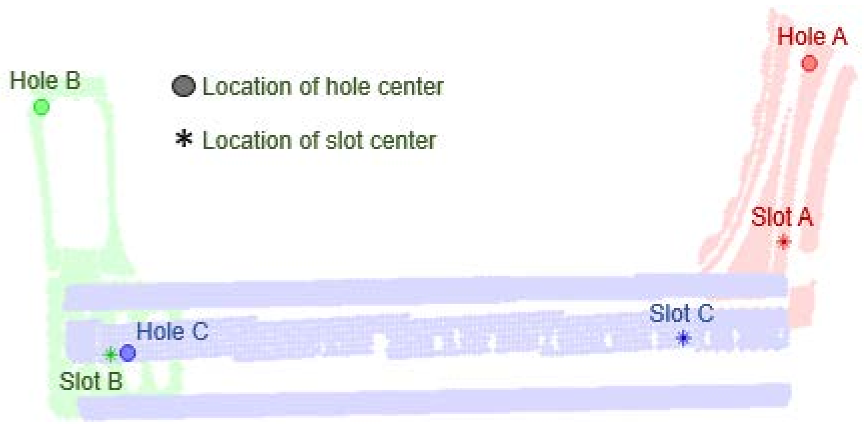


Figure 72 Baseline configuration of locators

7.3. Results and discussion

In Figure 73, all of the points with variance lower than the baseline are highlighted using a performance region after a TR-MPS optimization with 5000 samples (~3 hours runtime). From the results, the objective can only be slightly reduced by offsetting holes and slots (from 0.32mm^2 to 0.297 mm^2). Perhaps more interesting, is the clear outlier. This outlier (with a value of 964 mm^2) was selected using the data table, then exported and plotted with an application specific visualization (also developed by the author). The result is shown in Figure 74. Figure 74 illustrates the reason for the outlier: the locators of Part C are concentric, allowing the part to rotate freely around the center of the hole/slot. Indeed, a constraint which ensures locators do not overlap is clearly necessary. With this information a constraint may be added in the optimization process to eliminate the outliers from the set of feasible data.

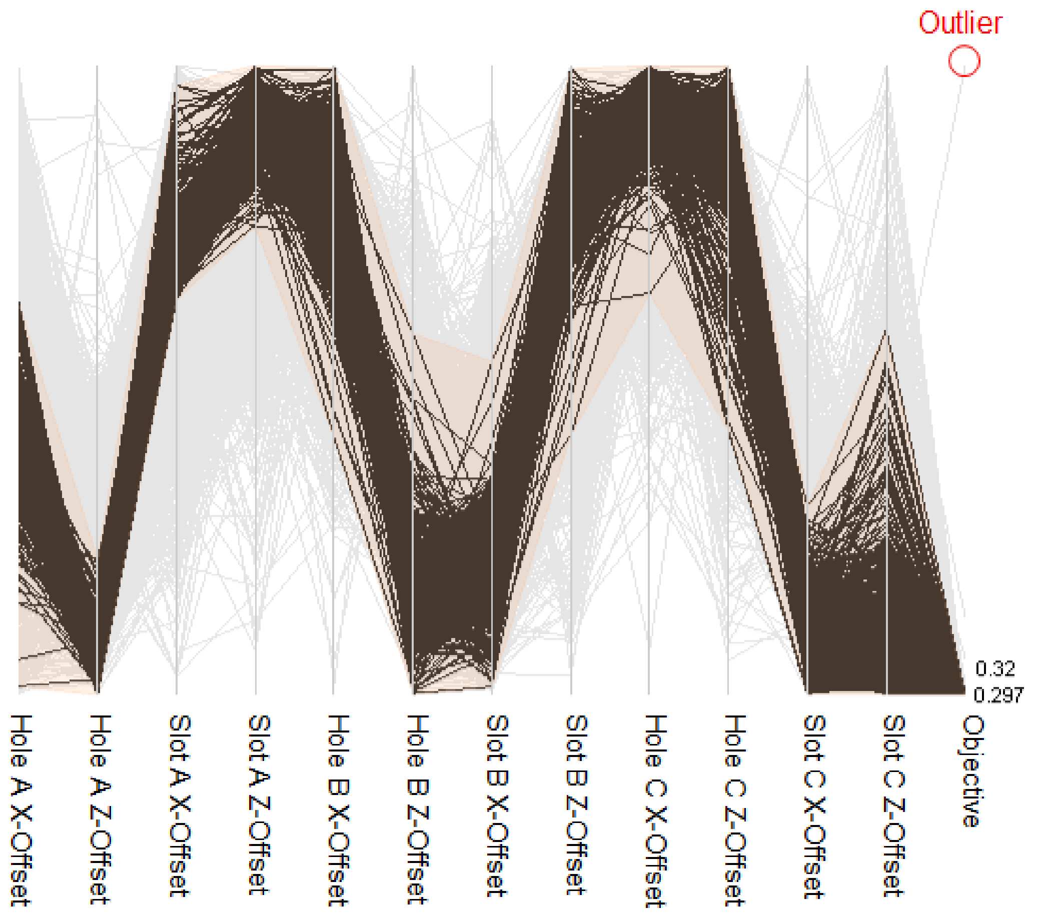


Figure 73 Locator offsets resulting in a lower variance than the baseline

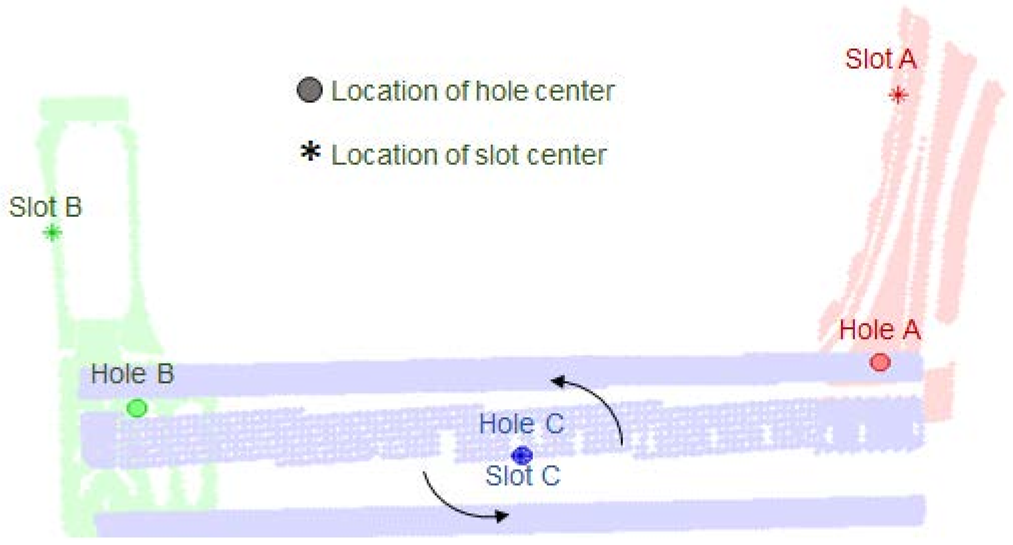


Figure 74 Outlier configuration (Part C is free to rotate)

Figure 75 shows the optimal design, also selected from the sorted data table. For Part A, the results suggest to completely lower the hole, as indicated by the maximum negative Z offset. The slot, on the other hand is moved up, as shown by the large positive offset in Z. For Part B, the hole is moved to the bottom right (large positive X offset and negative Z offset) and the slot is moved to the top left (large negative X offset positive Z offset). In Part C, the Hole is moved to the top right and the slot to the bottom left. Overall, from Figure 75, the suggested offsets can be readily determined. However, the designer must interpret the meaning of each numeric value, and how it relates to the initial locator scheme. In other words, the results are not immediately clear, and require some thought.

Alternatively, the results may be plotted as in Figure 76. Here, it is instantly clear how the recommended locator configuration is arranged on the parts. The results can be understood pre-attentively (without cognitive effort). Namely, the locators are spread to the corner of the parts for this particular assembly. On the other hand, this visualization required knowledge about the optimization problem. Each offset had to be added to an initial X-Z location, representing the final position of a particular locator. In practice, a general visualization framework may be useful to generate data and identify sub-regions of the data. The region can then be exported into an application specific tool if available.

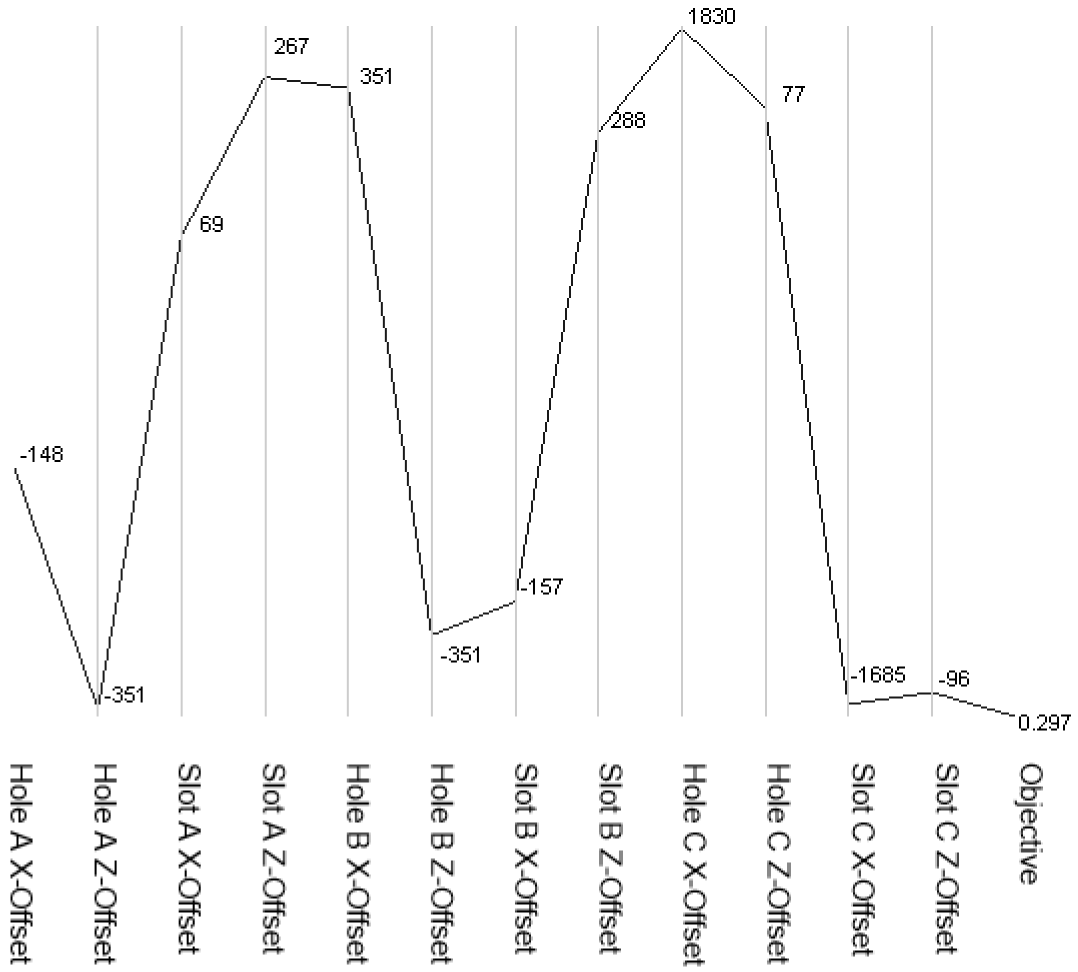


Figure 75 Locator offsets which generate the lowest assembly variance

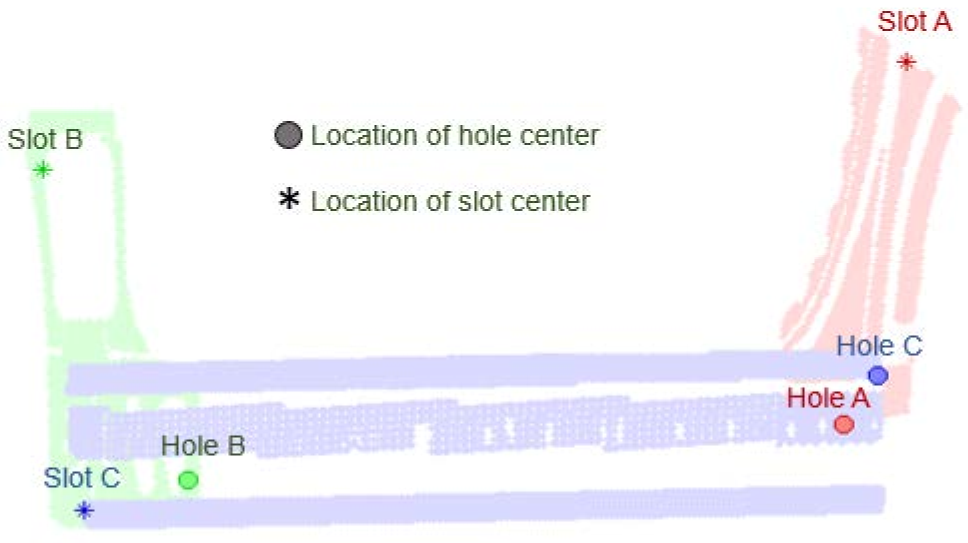


Figure 76 Application specific visualization locator offsets at minimum

The design region bounding the points from 0.297 to baseline (the orange area in Figure 73) was also added as a design region using the ("Add as design region button"). The worst sampled performance in the design region is $0.45mm^2$. In an attempt to further investigate the extremes of this region, a metamodel was also built. However, a second-order polynomial regression is clearly not suitable for metamodeling this region as shown in Figure 77. This is confirmed by an NRMSECV of 0.12 and an R-Square of 0.47. Therefore, the simple metamodel cannot predict a worst or best case performance for the region. PCA-HDMR is even further off with a NRMSECV of 23.

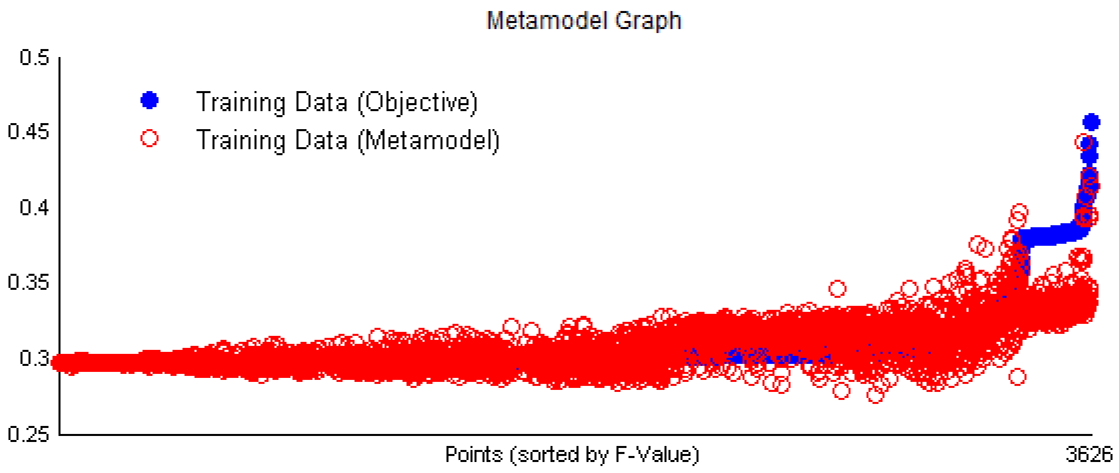


Figure 77 Metamodel of region bounding the points which are lower than baseline

Aside from design data, additional information about the optimization process may be gained from the convergence chart shown in Figure 78. For example, there was no improvement in the last 168 TR-MPS iterations (the optimum was found in iteration 357). This suggests that the algorithm has converged.

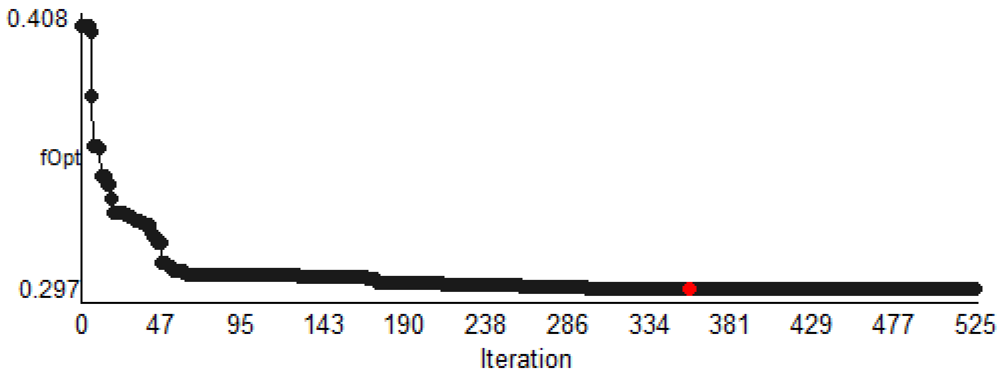


Figure 78 Three part assembly optimization convergence chart

7.4. Summary

This chapter presented an automotive assembly station optimization project, as an application for visualization in support of design optimization. It was shown that the framework introduced in this thesis is able to reveal some helpful information about the optimization problem. For example, after finding and investigating an outlier it became clear that a constraint preventing the overlap of locators is necessary. On the other hand, this chapter also shows that using an application specific visualization is ideal. For instance, in this case, the locator offsets can be added to the initial positions and directly plotted on the parts as glyphs. Nonetheless, in the absence of further information about the problem or geometry, parallel coordinates and general information visualization may be a starting point for further investigation.

Chapter 8. Summary and Future Work

8.1. Summary

This thesis presents a framework to visualize data in optimization problems with the goal of providing more transparency about the optimization problem and process. By visualizing data during optimization, the designer may interject and make adjustments to their problem, such as directing the search between new search bounds or adjusting constraints. New features that were developed in this integrated design optimization framework that separate this work from others are listed as follows.

In this work, an emphasis is placed on defining sub-regions of the design or performance spaces to track optimization progress. Clustering may further split the region into sub-spaces automatically. Additionally, users may use an adjacent two-dimensional projection or data table to select data manually. Principal-component-analysis was applied to generate a two-dimensional projection that captures the most variance in the selected designs. Once regions are defined, they may be modeled with a polynomial regression or HDMR, which can replace the expensive simulation if the model is valid in the region. Model validity is estimated using cross-validation with the option of adding additional test points. This local model may be used to predict the performance extremes of the selected region using a local optimization on the metamodel. Besides showing the convergence of the design space, the current optimum value is plotted against the iteration; and the iterations which generated the selected data are highlighted. Furthermore, users may navigate to past iterations to see when data was introduced, and how the sampling strategy biases later samples.

Aside from visualization, an algorithm, incorporating association analysis, is presented to automatically identify redundancies in constraints which can be confirmed visually. This algorithm finds relationships between constraints as rules that may be used to suppress constraints, simplifying the problem formulation and potentially reducing the number of constraint checks during optimization.

An application was presented for the optimization of robotic automotive assembly station fixtures. This example showed how visualization may assist in identifying missing information from the problem formulation. For example, it is clear that constraints must be added to ensure that locators do not overlap. It was also shown that although a general visualization tool is beneficial, an application specific visualization may be far more efficient if available.

8.2. Future Work

Although this work progresses multivariate visualization in support of optimization by visualizing convergence in design data, many improvements may still be made. For instance, a key limitation is that the visualization techniques may become incomprehensible with many variables (e.g. more than 20). For example, if a problem has 100 variables, it is difficult to follow a single polyline in parallel coordinates. Similarly the scatter plots in a scatterplot matrix become too dense to read and too costly to compute. To overcome this, a more in depth analysis of the data, prior to visualization, is required. For instance, using a global metamodel, such as HDMR, a subset of variables, that significantly influence the output, may be grouped. Alternatively dimensionality reduction techniques, such as PCA, may aggregate variables into a limited number of components for display. This work extends visualization of the optimization process beyond three dimensions. However, there is undoubtedly progress to be made in visualizing optimization of many variables.

References

[1] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H., 2011, Big data: The next frontier for innovation, competition, and productivity, Global.

[2] Short, J. E., Bohm, R. E., and Baru, C., 2011, How much information? 2010 report on enterprise server information, San Diego, CA.

[3] Magana, A. J., Vieira, C., Polo, F. G., Yan, J., and Sun, X., 2013, "An exploratory survey on the use of computation in undergraduate engineering education," Frontiers in Education Conference, 2013 IEEE, Oklahoma City, OK, pp. 7–9.

[4] Origin, T., Development, E., Author, M. W., Source, W., and Stable, S., 2014, "Thirtieth anniversity of the use of the term operational research," Oper. Res. Q., 1967(2), pp. 111–113.

[5] Dantzig, G. B., 2002, "Linear programming," Oper. Res., 50(1), pp. 42–47.

[6] Henry, J., 1975, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence, MIT Press Cambridge, Massachusetts, USA.

[7] Kirkpatrick, S., Gellatt, C. D., and Vecchi, M. P., 1983, "Optimization by simulated annealing," Science, 220(4598), pp. 671–680.

[8] Kennedy, J., and Eberhart, R., 1995, "Particle swarm optimization," Proceeding of IEEE International Conference on Neural Networks, 1995, IEEE, Perth, WA, pp. 1942–1948.

[9] Jones, D., Schonlau, M., and Welch, W., 1998, "Efficient global optimization of expensive black-box functions," J. Glob. Optim., 13, pp. 455–492.

[10] Messac, A., 1996, "Physical programming : effective optimization for computational design," AIAA J., 34(1), pp. 149–158.

[11] Karwan, M. H., Lofti, V., Telgen, J., and Zoints, S., 1983, Redundancy in mathematical programming: a state-of-the-art survey, Springer-Verlag, New York, Berlin.

[12] Balling, R., 1999, “Design by shopping: A new paradigm?,” Proceedings of the 3rd WSMO, Third world congress of structural and multidisciplinary optimization, Buffalo, NY, pp. 295–297.

[13] Geoffrion, A. M., 1976, “The purpose of mathematical programming is insight, not numbers,” Interfaces (Providence)., 7(1).

[14] Tufte, E. R., 2001, The visual display of quantitative information, Graphics Press, Chesire, Connecticut.

[15] Tukey, J., 1977, Exploratory data analysis, Addison-Wesley, Reading, MA.

[16] Anscombe, F. J., 1973, “Graphs in statistical analysis,” Am. Stat., 27(1), pp. 17–21.

[17] Spence, R., 2007, Information visualization: design for interaction, Pearson Education, Edinburgh Gate, England.

[18] Swayne, D. F., Lang, D. T., Buja, A., and Cook, D., 2003, “GGobi (Software).”

[19] Morandat, F., Hill, B., Osvald, L., and Vitek, J., 2012, “Evaluating the design of the R language: objects and functions for data analysis,” ECOOP’ 12 Proceedings of the 26th European conference on Object-Oriented Programming, pp. 104–131.

[20] Nicolau, M., Levine, A. J., and Carlsson, G., 2011, “Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival,” Proc. Natl. Acad. Sci. U. S. A., 108(17), pp. 7265–7270.

[21] Stump, G. M., Yukish, M. A., Martin, J. D., and Simpson, T. W., 2004, “The ARL trade space visualizer: An engineering decision-making tool,” 10th AIAA/ISSMO Multidisciplinary Analysis and Optmization Conference, Albany, NY.

[22] Winer, E. H., and Bloebaum, C. L., 2001, “Visual design steering for optimization solution improvement,” Struct. Multidiscip. Optim., 22(3), pp. 219–229.

[23] Agrawal, G., Lewis, K. E., Chugh, K., Huang, C. H., Parashar, S., and Bloebaum, C. L., 2004, “Intuitive visualization of pareto frontier for multi-objective optimization in n-dimensional performance space,” 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY.

[24] Mattson, C. A., and Messac, A., 2002, “Concept selection in n-dimension using s-Pareto frontiers and visualization,” 9th AIAA / ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA.

[25] Keim, D. A., Mansmann, F., Thomas, J., and Keim, D., 2010, “Visual analytics : how much visualization and how much analytics ?,” SIGKDD Explor., 11(2), pp. 5–8.

[26] Rhyne, T.-M., Tory, M., Munzner, T., Ward, M., Johnson, C., and Laidlaw, D. H., 2003, "Information and scientific visualization : Separate but equal or happy together at last," IEEE Visualization, IEEE, ed., Seattle, Washington, pp. 611–614.

[27] Freindly, M., and Denis, D., 2005, "The early origins and development of the scatterplot," J. Hist. Behav. Sci., 41(2), pp. 103–130.

[28] Bertin, J., 1981, Graphics and graphic information processing, Walter de Gruyter & Co, Berlin.

[29] Cleveland, W. S., and McGill, R., 1985, "Graphical perception and graphical methods for analyzing scientific data," Science, 229(4716), pp. 828–833.

[30] Mackinlay, J., 1986, "Automating the design of graphical presentations of relational information," ACM Trans. Graph., 5(2), pp. 110–141.

[31] Grinstein, G., Trutschl, M., and Cvek, U., 2001, "High-dimensional visualizations," VII Data Mining Conference KDD Workshop 2001, ACM Press, New York, pp. 7–19.

[32] Steinmayr, B., 2009, "Hypervariate data visualization," Media Informatics Advanced Seminar on Information Visualization 2008/2009, Munich, Germany.

[33] Wong, P. C., and Bergeron, R. D., 1997, "30 years of multidimensional multivariate visualization," Scientific Visualization, Overviews, Methodologies, and Techniques, IEEE Computer Society, Washing, DC, pp. 3–33.

[34] Jones, C. V., 1996, Visualization and optimization, Kluwer Academic Publishers, Boston.

[35] Fisher, R. A., 1936, "The use of multiple measurements in taxonomic problems," Ann. Eugen., 7(2), pp. 179–188.

[36] Elmqvist, N., Dragicevic, P., and Fekete, J.-D., 2008, "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation," IEEE Trans. Vis. Comput. Graph., 14(6), pp. 1141–1148.

[37] Cutbill, A., Hajikolaei, K. H., and Wang, G. G., 2013, "Visual HDMR model refinement through iterative interaction," Proceedings of the ASME 2013 International Design Engineering Technical Conferences, Portland, USA, pp. 1–8.

[38] Inselberg, A., 1985, "The plane with parallel coordinates," Vis. Comput., 1(2), pp. 69–91.

[39] Fua, Y., Ward, M. O., and Rundensteiner, E. A., 1999, "Hierarchical parallel coordinates for exploration of large datasets," Visualization '99 Proceedings, IEEE, San Franciso, CA, pp. 43–58.

[40] Heinrich, J., and Weiskopf, D., 2009, “Continuous parallel coordinates,” IEEE Trans. Vis. Comput. Graph., 15(6), pp. 1531–8.

[41] Walker, R., Legg, P. a., Pop, S., Geng, Z., Laramee, R. S., and Roberts, J. C., 2013, “Force-directed parallel coordinates,” 2013 17th International Conference on Information Visualisation, Ieee, London, UK, pp. 36–44.

[42] Yuan, X., Guo, P., Xiao, H., Zhou, H., and Qu, H., 2009, “Scattering points in parallel coordinates.,” IEEE Trans. Vis. Comput. Graph., 15(6), pp. 1001–1008.

[43] Lu, L. F., Huang, M. L., and Huang, T.-H., 2012, “A new axes re-ordering method in parallel coordinates visualization,” 2012 11th International Conference on Machine Learning and Applications, IEEE Computer Society, Boca Raton, FL, pp. 252–257.

[44] Keim, D. a., and Kriegel, H.-P., 1994, “VisDB: Database exploration using multidimensional visualization,” IEEE Comput. Graph. Appl., 14(5), pp. 40–49.

[45] Jolliffe, I. T., 2002, Principal component analysis, Springer, New York.

[46] Kim, J.-O., and Mueller, C. W., 1978, “Introduction to factor analysis: What it is and how to do it,” Sage Univ. Pap. Ser. Quant. Appl. Soc. Sci., 07(013).

[47] Scholkopf, B., Smola, A., and Müller, K. R., 2005, “Kernel principal component analysis,” Lect. Notes Comput. Sci., 1327(1997), pp. 583–588.

[48] Roweis, S. T., and Saul, L. K., 2000, “Nonlinear dimensionality reduction by locally linear embedding,” Science, 290(5500), pp. 2323–2326.

[49] Kruskal, J. B., 1978, Multidimensional scaling, SAGE Publications, Thousand Oaks, CA.

[50] Tenenbaum, J. B., de Silva, V., and Langford, J. C., 2000, “A global geometric framework for nonlinear dimensionality reduction,” Science, 290(5500), pp. 2319–2322.

[51] Maaten, L. J. P. Van Der, Postma, E. O., and Herik, H. J. Van Den, 2009, Dimensionality reduction : A comparative review, Tilburg, Netherlands.

[52] Shan, S., and Wang, G. G., 2009, “Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions,” Struct. Multidiscip. Optim., 41(2), pp. 219–241.

[53] Takagi, H., 2001, “Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation,” Proc. IEEE, 89(9), pp. 1275–1296.

[54] Sims, K., 1991, “Artificial evolution for computer graphics,” Comput. Graph. (ACM)., 25(4), pp. 319–328.

[55] Anderson, D., Anderson, E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D., and Ryall, K., 2000, “Human-guided simple search,” Proceedings of AAAI (American Association for Artificial Intelligence), Austin, Texas, pp. 209–216.

[56] Klau, G. W., Lesh, N., Marks, J., and Mitzenmacher, M., 2002, Human-guided tabu search, Cambridge, Massachusetts.

[57] Froschauer, M., 2009, “Interactive Optimization , Distance Computation and Data Estimation in Parallel Coordinates (Thesis),” Vienna University of Technology.

[58] Arora, J. S., 2004, Introduction to optimum design, Elsevier, San Diego, CA.

[59] Winer, E. H., and Bloebaum, C. L., 2002, “Development of visual design steering as an aid in large-scale multidisciplinary design optimization. Part I: method development,” Struct. Multidiscip. Optim., 23, pp. 412–424.

[60] Winer, E. H., and Bloebaum, C. L., 2002, “Development of visual design steering as an aid in large-scale multidisciplinary design optimization. Part II : method validation,” Struct. Multidiscip. Optim., 23, pp. 425–435.

[61] Afimiwala, K. A., and Mayne, R. W., 1979, “A contour plotting scheme for design optimization,” J. Mech. Des., 101, pp. 349–354.

[62] Lego, S. E., Stump, G. M., and Yukish, M. A., 2010, “Trade space exploration: New visual steering features,” 2010 IEEE Aerospace Conference, Big Sky, Montana.

[63] Stump, G. M., Lego, S. E., Yukish, M. A., Simpson, T. W., and Donndelinger, J. A., 2009, “Visual steering commands for trade space exploration: User-guided sampling with example,” J. Comput. Inf. Sci. Eng., 9(4).

[64] Yan, X., Qiao, M., Li, J., Simpson, T. W., Stump, G. M., and Zhang, X. (Luke), 2012, “A work-centered visual analytics model to support engineering design with interactive visualization and data-mining,” 2012 45th Hawaii Int. Conf. Syst. Sci., pp. 1845–1854.

[65] Messac, A., and Wilson, B. H., 1998, “Physical programming for computational control,” AIAA J., 36(2), pp. 219–226.

[66] Messac, A., and Chen, X., 2000, “Visualizing the optimization process in real-time using physical programming,” Eng. Optim., 32(6), pp. 721–747.

[67] Eddy, J., and Lewis, K. E., 2002, “Visualization of multidimensional design and optimization using cloud visualization,” Proceedings of DETC’02, Montreal, Canada, Canada, pp. 899–908.

[68] Spence, B., Tweedie, L., and Dawkes, H., 1995, “Visualisation for functional design,” Proc. Vis. 1995 Conf., pp. 4–10,.

[69] Spence, R., 1999, “The facilitation of insight for analog design,” IEEE Trans. circuits Syst. II, 46(5), pp. 540–548.

[70] Tan, P.-N., Steinbach, M., and Kumar, V., 2006, Introduction to data mining, Pearson Education, Boston.

[71] Goodman, T., and Spence, R., 1978, “The effect of system response time on interactive computer aided problem solving,” SIGGRAPH ’78 Proceedings of the 5th annual conference on computer graphics and interactive techniques, ACM, ed., Atlanta, Georgia, pp. 100–104.

[72] Lam, H., 2008, “A framework of interaction costs in information visualization,” IEEE Trans. Vis. Comput. Graph., 14(6), pp. 1149–1156.

[73] Wilde, D., 1978, Globally optimal design, John Wiley and Sons Inc, New York, NY.

[74] Wang, L., Shan, S., and Wang, G. G., 2004, “Mode-pursuing sampling method for global optimization on expensive black-box functions,” Eng. Optim., 36(4), pp. 419–438.

[75] Lewis, K., and Mistree, F., 1996, “Foraging-directed adaptive linear programming: an algorithm for solving nonlinear mixed discrete/continuous design problems,” Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, Irvine, California, p. 1601.

[76] Sobol, I. M., 1993, “Sensitivity Estimates for Nonlinear Mathematical Models,” Math. Model. Comput. Exp., 1(4), pp. 407–411.

[77] Simpson, T. W., Peplinski, J. D., Koch, P. N., and Allen, J. K., 2001, “Metamodels for Computer-based Engineering Design : Survey and recommendations,” Eng. Comput., 17(2), pp. 129–150.

[78] Wang, G. G., and Shan, S., 2007, “Review of metamodeling techniques in support of engineering design optimization,” J. Mech. Des., 129(4), p. 370.

[79] Cheng, G. H., Younis, A., Hajikolaei, K. H., and Wang, G. G., 2014, “Trust region based MPS method for global optimization of high dimensional design problems,” J. Mech. Des., (Accepted).

[80] Ziehn, T., and Tomlin, A. S., 2009, “GUI–HDMR – A Software Tool for Global Sensitivity Analysis of Complex Models,” Environ. Model. Softw., 24(7), pp. 775–785.

[81] Rabitz, H., Alıs, Ö. F., Shorter, J., and Shim, K., 1999, “Efficient Input-Output Model Representations,” Comput. Phys. Commun., 117, pp. 11–20.

[82] Alıs, Ö. F., and Rabitz, H., 2001, “Efficient Implementation of High Dimensional Model Representations,” J. Math. Chem., 29(2), pp. 127–141.

[83] Li, G., Wang, S.-W., and Rabitz, H., 2002, “Practical Approaches To Construct RS-HDMR Component Functions,” J. Phys. Chem. A, 106(37), pp. 8721–8733.

[84] Shan, S., and Wang, G. G., 2011, “Turning Black-Box Functions Into White Functions,” J. Mech. Des., 133(3), p. 031003.

[85] Hajikolaei, K. H., and Wang, G. G., 2013, “High Dimensional Model Representation with Principal Component Analysis : PCA-HDMR,” J. Mech. Des., Submitted.

[86] Rabitz, H., and Alıs, Ö. F., 1999, “General Foundations of High-Dimensional Model Representations,” J. Math. Chem., 25, pp. 197–233.

[87] Cutbill, A., and Wang, G. G., “Mining constraint relationships and redundancies with association analysis for optimization problem formulation,” Eng. Optim., (Accepted with minor revisions).

[88] Agrawal, R., Imielinksi, T., and Swami, A., 1993, “Database mining: a performance perspective,” IEEE Trans. Knowl. Data Eng., 5(6), pp. 914–925.

[89] Agrawal, R., Imielinski, T., and Swami, A., 1993, “Mining association rules between sets of items in large databases,” Proceedings of the 1993 ACM SIGMOD international conference on management of data, ACM Press, New York, NY, pp. 207–216.

[90] Paulraj, S., and Sumathi, P., 2010, “A comparative study of redundant constraints identification methods in linear programming problems,” Math. Probl. Eng., 2010(723402), pp. 1–16.

[91] Caron, R. J., McDonald, J. F., and Ponic, C. M., 1989, “A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary,” J. Optim. Theory Appl., 62(2), pp. 225–237.

[92] Thompson, G. L., Tonge, F. M., and Zionts, S., 1966, “Techniques for removing nonbinding constraints and extraneous variables from linear programming problems,” Manage. Sci., 12(7), pp. 588–608.

[93] Boot, T. C. G., 1962, “On trivial and binding constraints in programming problems,” Manage. Sci., 8(4), pp. 419–441.

[94] Brearley, A. L., Mitra, G., and Williams, H. P., 1975, “Analysis of mathematical programming problems prior to applying the simplex algorithm,” Math. Program., 8(1), pp. 54–83.

[95] Boneh, A., 1984, “Identification of redundancy by a set-covering equivalence,” Operational Research’ 84. Proceedings of the Tenth International Conference on Operational Research, J.P. Brans, ed., Elsevier, Amsterdam, North Holland, pp. 407–422.

[96] Feng, J., 1999, “Redundancy in nonlinear systems: a set covering approach (Thesis),” University of Windsor.

[97] Huang, W., Krneta, M., Lin, L., Wu, J., and Generation5 Math Technologies, 2006, “Association bundle - a new pattern for association analysis,” Sixth IEEE International Conference on Data Mining- Workshops (ICDMW’06), IEEE, Hong Kong, pp. 601–605.

[98] Han, J., Pei, J., Yin, Y., and Mao, R., 2004, “Mining frequent patterns without candidate generation : A frequent-pattern tree,” Data Min. Knowl. Discov., 8(1), pp. 53–87.

[99] Zhou, L., and Yau, S., 2007, “Efficient association rule mining among both frequent and infrequent items,” Comput. Math. with Appl., 54(6), pp. 737–749.

[100] Xiong, H., Tan, P.-N., and Kumar, V., 2006, “Hyperclique pattern discovery,” Data Min. Knowl. Discov., 13(2), pp. 219–242.

[101] Cheng, H., Yu, P., and Han, J., 2006, “AC-Close: Efficiently mining approximate closed itemsets by core pattern recovery,” Proceedings of the Sixth International Conference on Data Mining (ICDM’06), IEEE, Hong Kong, pp. 839–844.

[102] Özden, B., Ramaswamy, S., and Sillberschatz, A., 1998, “Cyclic Association Rules,” 14th International Conference on Data Engineering, 1998 Proceedings, IEEE, Orlando, FL, pp. 412–421.

[103] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L., 1999, “Discovering frequent closed itemsets for association rules,” ICDT ’99 Proceedings of the 7th International Conference on Database Theoy, C. Beeri, and P. Buneman, eds., Springer-Verlag, Jerusalem, Israel, pp. 398–416.

[104] Tan, P.-N., Kumar, V., and Srivastava, J., 2002, “Selecting the right interestingness measure for association patterns,” KDD’ 02 Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press, Edmonton, AB, pp. 32–41.

[105] Atluri, G., Gupta, R., Fang, G., Pandey, G., Steinbach, M., and Kumar, V., 2009, “Association analysis techniques for bioinformatics problems,” Lect. Notes Comput. Sci., 5462(1), pp. 1–13.

[106] Potter, C., Klooster, S., Steinbach, M., Tan, P., Kumar, V., Shekhar, S., Nemani, R., and Myneni, R., 2003, “Global teleconnections of ocean climate to terrestrial carbon flux,” J. Geophys. Res., 108(4556).

[107] Ceglarek, D., and Shi, J., 1996, “Fixture failure diagnosis for autobody assembly using pattern recognition,” J. Eng. Ind., 118(1), p. 55.

Appendix

Additional Mathematical Details

Mathematical procedure of PCA on optimization design data

The mathematics of PCA is well established and presented here. It can be summarized as a four step process.

Step 1a- Preprocessing: Data standardization

The first step in performing PCA, for optimization, is to standardize the design space between [0,1]. This ensures that each variable has the same impact, regardless of range or units. For example, in the PV problem, if the data were not standardized, L would account for most of the variance, as its range is between 25 and 240. On the other hand, if the tested designs are standardized, each variable is treated with equal weight.

Standardization can be achieved through the following equations: (31) or (32). Eq. (32), (a.k.a. *normalization* in statistics), is most common for PCA in general. However, Eq. (32) assumes a normal distribution for each attribute when computing $\boldsymbol{\sigma}(\boldsymbol{X})$ (the variable value standard deviations). This assumption is not valid for optimization, where data is biased towards the optimum, or user selected regions. Therefore Eq. (31) is used for standardization. In Eq. (31) and (32), $\boldsymbol{D_L}$ and $\boldsymbol{D}_U$ are the lower and upper bounds of the design space (respectively).

$$\boldsymbol{x}_S^{\boldsymbol{i}} = \frac{\boldsymbol{x}^{\boldsymbol{i}} - \boldsymbol{D}_L}{\boldsymbol{D}_U - \boldsymbol{D}_L} \tag{31}$$

$$\boldsymbol{x}_{\boldsymbol{s}}^{\boldsymbol{i}} = \frac{\boldsymbol{x}^{\boldsymbol{i}} - \boldsymbol{\mu}(\boldsymbol{X})}{\boldsymbol{\sigma}(\mathbf{X})} \tag{32}$$

Step 1b- Preprocessing: Data centering

The second step is to center the data such that the mean the data for each attribute is zero. This step is called data centering and is accomplished by simply subtracting the mean of each attribute (column of $\boldsymbol{X_s}$) from each design.

$$\boldsymbol{x}^{\boldsymbol{i}\prime} = \boldsymbol{x}^{\boldsymbol{i}} - \boldsymbol{\mu}(\boldsymbol{X_s}) \tag{33}$$

Step 2- Computing the covariance matrix

Once the data is preprocessed, the next step is compute the sample covariance matrix of $\boldsymbol{A}$. The preprocessed version of $\boldsymbol{X}$ will be denoted as $\boldsymbol{A} = \boldsymbol{X}'$ to avoid confusion. Covariance is a measure of how a pair of two variables changes together. For example, if the increase in one variable indicates the increase of another variable, they have positive covariance; if the increase of one variable indicates the decrease of another they have negative covariance. The notation $\boldsymbol{a_{*j}}$ means all of the (preprocessed) sample values for the j_{th} variable, while $\boldsymbol{\mu}_j$ is the mean of that column. Sample covariance (the estimated covariance based on the sample of data) can between two attributes (j and k) can written as in Eq. (34):

$$s_{jk} = cov\left(\boldsymbol{a_{*j}}, \boldsymbol{a_{*k}}\right) = E[(\boldsymbol{a_{*j}} - \boldsymbol{\mu}_j)(\boldsymbol{a_{*k}} - \boldsymbol{\mu}_k)] = \frac{(\boldsymbol{a_{*j}})^{\boldsymbol{T}}(\boldsymbol{a_{*k}})}{m-1} \tag{34}$$

Note that since the data is centered such that the mean is zero, $\boldsymbol{\mu}$ is eliminated in (34). The overall sample covariance $\boldsymbol{S}$ matrix (of size nxn) can be written as in Eq. (35).

$$\boldsymbol{S} = \frac{\boldsymbol{A}^T\boldsymbol{A}}{m-1} \tag{35}$$

Step 3 Eigenvectors and Eigenvalues of S

The covariance matrix summarizes which attributes vary together with sign and magnitude. If $\boldsymbol{S}$ is thought of a scaling matrix for a vector $\boldsymbol{u} \in \boldsymbol{U}$**,** then the multiplication $\boldsymbol{Su}$ is proportional to the data variance in the direction $\boldsymbol{u}$**.** To find the matrix $\boldsymbol{U}$ and the values $\boldsymbol{u} \in \boldsymbol{U}$**,** which maximize $\boldsymbol{Su}$**,** Eigen-decomposition can be performed on $\boldsymbol{S}$. The matrix $\boldsymbol{U}$ is simply the set of eigenvectors of $\boldsymbol{S}$. Furthermore, the eigenvectors corresponding to the largest eigenvalues represent the most variance as shown Eq. (36) ($\boldsymbol{Su}$ increases with λ). Therefore, $\boldsymbol{U}$ can be sorted column wise, by the values of λ, such that the first columns represent the orthogonal directions of maximum variance.

$$\boldsymbol{Su} = \lambda\boldsymbol{u} \tag{36}$$

Step 4 Projection of standardized points onto Eigenvectors

The last step is to simply plot all of the (standardized and centered) designs from $\boldsymbol{A}$ in the two directions that maximum variance of the selected points $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ (the 1st and 2nd principal components) as in Eq. (37).

$$\boldsymbol{A_{projected}} = [\boldsymbol{A}u_1 \; \boldsymbol{A}_{u_2}] \tag{37}$$

This completes the overall process of PCA.