2017

Visualizing engineering design data using a modified two-level self-organizing map clustering approach

Adam Robert Kohl
Iowa State University

Visualizing engineering design data using a modified two-level self-organizing map clustering approach

by

Adam R. Kohl

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-Majors: Mechanical Engineering; Human Computer Interaction

Program of Study Committee:
Eliot Winer, Major Professor
James Oliver
Christina Bloebaum

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2017

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to thank my family, Amy, Brian, and Rebecca. Their unwavering love and support made it possible for me to get through the tumultuous process that was this thesis. Additionally, I would like to express my gratitude for my advisor Dr. Eliot Winer, who guided me through this thesis and provided me the opportunity to be a member of the Virtual Reality Applications Center.

ABSTRACT

Engineers tasked with designing large and complex systems are continually in need of decision-making aids able to sift through enormous amounts of data produced through simulation and experimentation. Understanding these systems often requires visualizing multidimensional design data. Visual cues such as size, color, and symbols are often used to denote specific variables (dimensions) as well as characteristics of the data. However, these cues are unable to effectively convey information attributed to a system containing more than three dimensions. Two general techniques can be employed to reduce the complexity of information presented to an engineer: dimension reduction, and individual variable comparison. Each approach can provide a comprehensible visualization of the resulting design space, which is vital for an engineer to decide upon an appropriate optimization algorithm.

Visualization techniques, such as self-organizing maps (SOMs), offer powerful methods able to surmount the difficulties of reducing the complexity of n-dimensional data by producing simple to understand visual representations that quickly highlight trends to support decision-making. The SOM can be extended by providing relevant output information in the form of contextual labels. Furthermore, these contextual labels can be leveraged to visualize a set of output maps containing statistical evaluations of each node residing within a trained SOM. These maps give a designer a visual context to the data set's natural topology by highlighting the nodal performance amongst the maps. A drawback to using SOMs is the clustering of promising points with predominately less desirable data. Similar data groupings can be revealed from the trained output maps using visualization techniques such as the SOM, but these are not inherently cluster analysis methods.

Cluster analysis is an approach able to assimilate similar data objects into "natural

groups" from an otherwise unknown prior knowledge of a data set. Engineering data composed of design alternatives with associated variable parameters often contain data objects with unknown classification labels. Consequently, identifying the correct classifications can be difficult and costly. This thesis applies a cluster analysis technique to SOMs to segment a high-dimensional dataset into "meta-clusters". Furthermore, the thesis will describe the algorithm created to establish these meta-clusters through the development of several computational metrics involving intra and inter cluster densities. The results from this work show the presented algorithm's ability to narrow a large-complex system's plethora of design alternatives into a few overarching set of design groups containing similar principal characteristics, which saves the time a designer would otherwise spend analyzing numerous design alternatives.

CHAPTER 1: INTRODUCTION

Motivation

The need for understanding the variable relationships occurring within large datasets has exponentially increased as the computational performance of personal computers has continued to grow. Investigators now have access to vast amounts of data that have the potential to unearth important insights to vital design information. Unfortunately, the majority of this "big data" is riddled with complex variable relationships. Instead of looking at one variable versus another, "big data" analytics has investigators attempting to understand n-dimensional relationships existing within large intricate datasets [1]. Analyzing this data can provide useful information about problems found in applications such as national intelligence, health care, marketing, and engineering [2]. However, interpreting "big data" has become more difficult as continued growth in computational performance allows more information to be feasibly accumulated. Data visualization and cluster analysis techniques show promise in overcoming this issue by providing insight to the natural characteristics composing complex datasets.

Design Optimization

The abundance of large and complex engineered systems has exponentially increased due to demands for high levels of performance while operating on reduced budget costs. Furthermore, a design of these systems may contain thousands of components, all of which are uniquely designed. Factors such as material properties, geometric dimensions, and manufacturing parameters are evaluated and chosen by designers for component parts. All the

while, an understanding of how each component effects the overarching design must be maintained. For example, if a designer decides to place an additional number of signal transponders to a geo-stationary communication satellite, the designer must understand how the transponders will affect the payload demands responsible for propelling the rocket into the desired space orbit. Designing a single system is a difficult enough task. The problem becomes increasingly complex when multiple systems must interact for the desired output. In these cases, attributes of one "subsystem" (e.g., satellite structure) interact with or influence variables of another subsystem (e.g., launch vehicle). Therefore, each time a subsystem is modified, all remaining subsystems may need to be changed. Determining the optimal combination of design variables satisfying design requirements is referred to as design optimization. The topic will not be explained in detail as a sole research field is dedicated to it.

Multiple subsystems may be well understood when isolated from one another, but the coupled interaction between them can be difficult to comprehend or even unknown. Engineers responsible for designing such overall systems often examine data in what are referred to as design and performance spaces. A design space is composed of all independent variables that determine the performance or dependent variables of the design. Many methods can be used to represent the design space. An example visualization is shown in Figure 1**,** where the x and y axes represent the design variables and the z axis is the resulting the function dependent on x and y. This design space visual is often used to provide engineers a quick representation of infeasible designs.

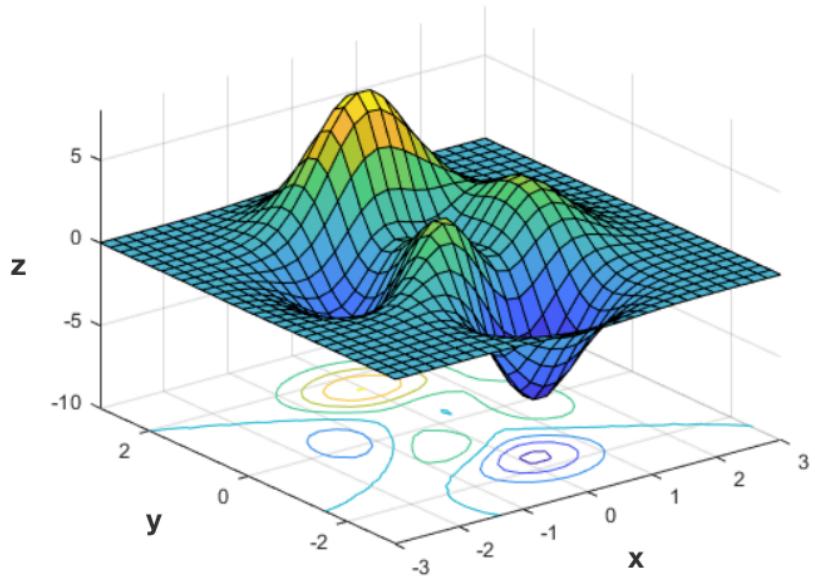

Figure 1. Three-dimensional visualization of a design space [3]

However, representing the design space in the same manner becomes infeasible as the number of design variables increase. If the design were to consist of six variables rather than the two shown in Figure 1, the engineer could not be certain variables x and y are producing infeasible designs. One method to combat this problem would be to view each of the 15 combinations of the six variables. Consequently, the larger number of visualizations make it extremely challenging to comprehend.

To avoid plotting a combination of a multi-variable plots, the design data can be represented in a performance space (i.e., objective formulations plotted against one another). A visualization, referred to as the Pareto frontier, can be used to show the area in which all the objectives are as close to their minimum as possible. For example, the Pareto frontier in Figure 2 plots the overall height and cost performance metrics. The dots colored red signify the Pareto frontier as they are the minimum performance metric evaluations yielding the most desirable designs. As seen in Figure 2, there is more than one optimal design, which produces many optimal designs constructing the Pareto frontier.

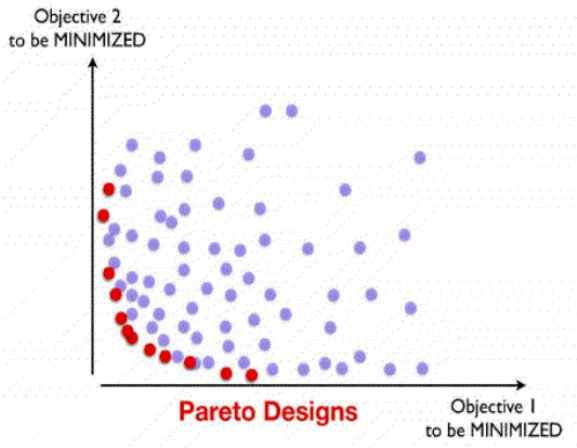

Figure 2. Pareto frontier visualization [4]

A typical problem has more independent design variables than objective functions, therefore the number of dimensions can be reduced while focusing on high-level performance information desired by the investigator. Although the Pareto frontier can appear to reduce the complexity of a problem, a lack of information regarding design variable interaction can lead to a lesser understanding of the overall design space. In addition, depending on the engineering application, more than two objectives may be used to evaluate the performance of a design. Consequently, a series of Pareto frontier visuals would need to be constructed to present all considerable designs, which can be difficult for an engineer to comprehend. Therefore, tradeoffs must be considered to select the most optimal design. Visualizing these tradeoffs have been a heavily research topic and will be addressed in chapter 2.

Summary

Data visualization is continuously improving to provide visual representations that convey relationships amongst design variables and performance characteristics in a directly interpretable manner. As the complexity of engineered designs continues to increase, the ability to understand the coupled interactions between subsystems diminishes. Optimizing

performance characteristics for a single system is a difficult enough task, but becomes increasingly complex when coupled with additional systems (i.e. subsystems). Designers of large and complex engineered systems are constantly in need of decision-making aids to sift through the enormous amounts of data produced through simulation and experimentation. Visualization methods show great potential by offering powerful methods to visualize high dimensional data with the goal to produce simple to understand representations that quickly highlight trends to support decision-making.

Thesis Organization

The work in this thesis will be presented as follows. Chapter 2 will introduce various approaches used to visualize multidimensional data and design spaces. Furthermore, the self-organizing map (SOM) will be introduced as well as the contextual self-organizing map (CSOM). The chapter concludes with a comprehensive review of a variety of cluster analysis methods used to classify a SOM. Chapter 3 contains a journal submitted to the IEEE Transactions on Knowledge and Data Engineering and details the methodology used in developing this work. Lastly, a concluding summary and discussion of future work will finalize the thesis in Chapter 4.

CHAPTER 2: REVIEW OF LITERATURE

As described in the previous chapter, visualizing multi-dimensional data and exploring design spaces presents a difficult challenge. The work of this thesis is built upon contributions and methods established in three core areas: visual trade space exploration, n-dimensional data visualization, and cluster analysis. The following sections of this chapter provide a review of literature highlighting novel work existing in each core area and presents the necessity of additional work.

Visual Trade Space Exploration

As the design of complex engineered products becomes increasingly difficult, the number of independent and dependent variables inherently grows. It is common to have systems of equations representing these products with tens to thousands of design variables. Navigating and examining this design space is truly a formidable task. Even before a formal optimization method can be run, a designer needs to understand the underlying variable relationships entangled within n-dimensional data. Multidimensional data visualization can offer tools and methods to aid in this.

Cloud Visualization (CVis) [5] is a visual tool developed with the purpose of allowing engineers to view large quantities of data to make effective decisions during the design and optimization process. CVis, shown in Figure 3, displays a side-by-side comparison of the data in the design space (left) and performance space (right). Both spaces are displayed three dimensionally. The performance space illustrates a design's proximity to the optimal solution generated from the problem's optimization formulation by presenting the variables with the greatest influence on a given design. Additionally, the performance space can display solutions relative to multiple design objectives.

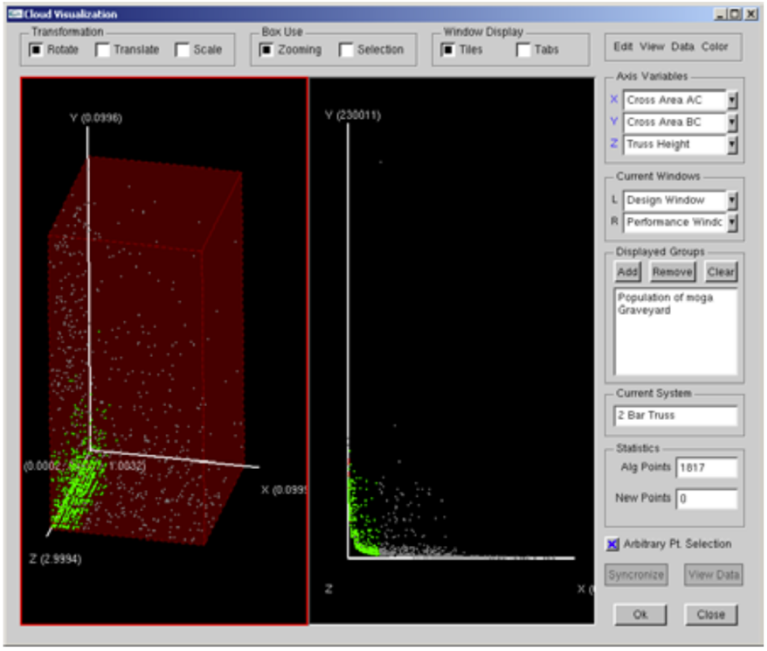

Figure 3. CVis Environment [5]

The CVis software can display both spaces in one, two, or three dimensions. This dimensional visualization provides the engineer the option of viewing multiple objective functions and relationships existing between multiple design variables. CVis provides an intuitive display to view individual design variables from large data sets and objective functions produced during the design and optimization process. However, the software can only visualize a maximum of three design variables and three objective functions. Consequently, multiple plots are needed to view all variables encompassing a large design space, which can confuse and overwhelm a designer.

One such approach was to improve a designer's interaction with an objective function throughout the optimization process by a paradigm termed visual design steering [6]. Visual steering first "ranks and reduces" design variables and constraints based on their respective influence on the objective function. Design variables and constraints with little or no impact are removed from the problem for visualization purposes. Winer and Bloebaum utilized Graph

Morphing, a technique to visualize n-dimensional data in three dimensions, to envision the ranked design tradeoffs by assigning a graphical slider bar to a variable's axis to view a real-time change in objective function as each slider manipulates the value of a variable. An example of the visual steering approach is shown in Figure 4. Three design variables are portrayed along each axis and overlaid with a color gradient representing the objective function value bound by the problem constraints illustrated in green.

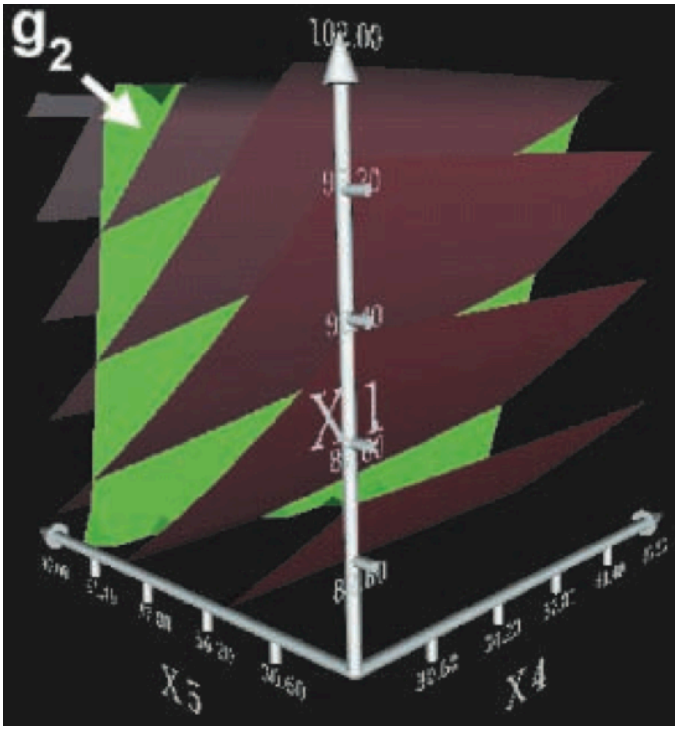

Figure 4. Graph Morphing example displaying three design variables, two constraints, and a decreasing objective function [6]

The visual steering approach suggested an optimization problem's complexity and solution time can be reduced by visualizing the design information. Being able to view an objective function morphing in real-time was a novel approach, but only allowed for a maximum of three variables to be pictured at a time. Therefore, multiple Graph Morphing plots must be generated for a side-by-side comparison when viewing greater than three design variables.

Hyper-Radial Visualization (HRV) was developed by Chiu et al. to view the interactions existing between an optimization problem's multiple objectives [7]. The work visualized the interactions in a two-dimensional space by grouping multiple objectives into two "manufactured" objectives. The manufactured objectives are then minimized to define a utopia point, which is represented as the origin in the example HRV shown in Figure 5. A radial projection is generated and encircles the utopia point. Each radius is defined by an overall objective value, therefore a point located on the radius contains the same objective value.

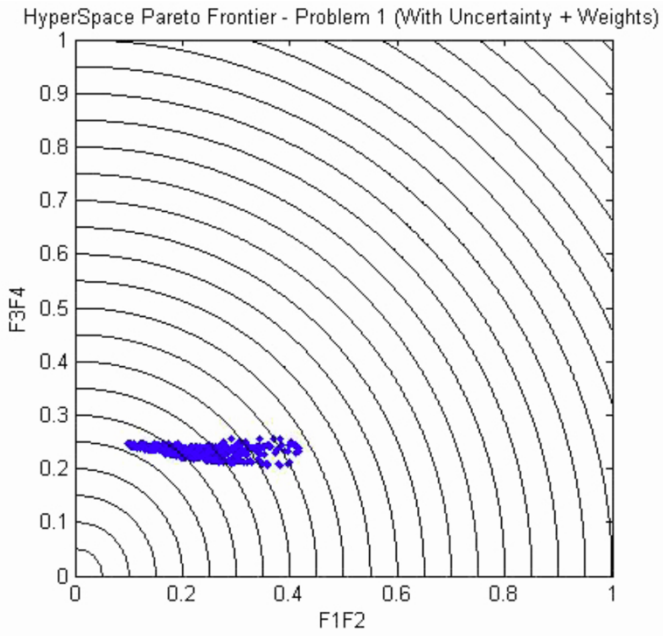

Figure 5. Hyper-Radial Visualization of a multiobjective optimization problem using uncertainty and weighting [7]

Additionally, HVR provides the designer the capability of adding a preference to individual objectives yielding a change in weighting and display of the optimum. To modify the weights, an engineer is presented a set of Likert scales ranging from highly desirable to highly undesirable. A visual representing the tradeoffs between two objective groupings is

then produced from the weights and preferences established by the designer. However, because the HVR interface is focused on the objective function values the approach limits the ability to interact with individual design variables.

Another exploration tool, known as XGobi, was proposed by Swayne et al. to display large data sets in one, two, or three dimensions [8]. The interactive dynamic data visualization system delivers an interface with a set of graphical tools capable of projecting high dimensional data onto a two-dimensional display, axis scaling, brushing, line editing, and point manipulation. As shown in Figure 6, plots produced by XGobi display design variables using parallel coordinates while providing each design variable its own axis.

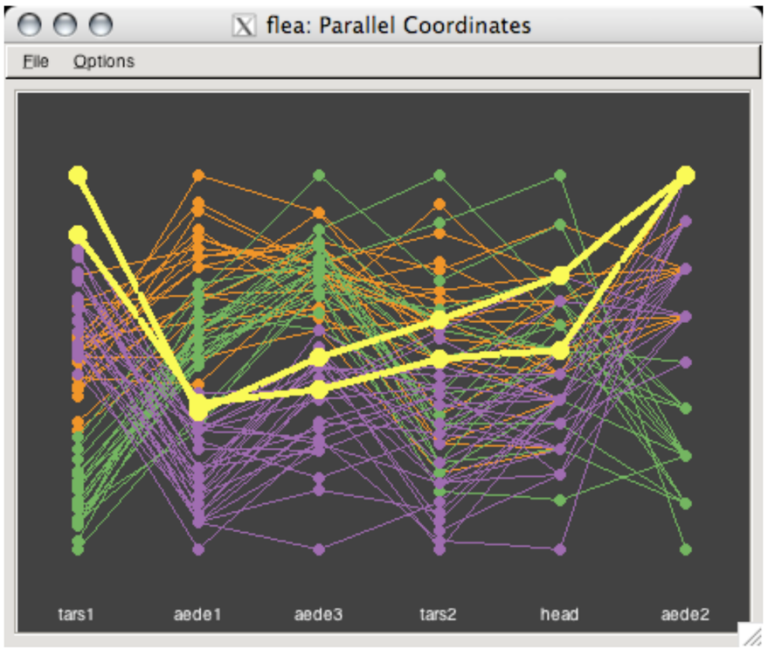

Figure 6. XGobi dimensional reduction using parallel coordinates [8]

XGobi supplies engineers a visual representation to aide in understanding variable relationships entangled within large data sets. However, as the dataset increases in size so does the amount of information presented to the engineer, which can hinder the engineer's ability to quickly interpret the variable relationships constructing the design space.

Stump et al. further expanded upon the dynamic data visualizing system XGobi [8] by using the shopping paradigm [9] to evaluate multidimensional visualization. Instead of running countless simulation iterations, the designer extracts the design variables and manipulates their ranges while an optimization routine is running. The current design variables are then visualized in three or less dimensions. Furthermore, the optimization routine will continue solving until a convergence criteria is met or the design variables are manipulated by the designer. The visualization system's main contribution is the display of extra design space information as the interaction of multiple objectives (i.e., performance space) in the Pareto frontier illustrated on a glyph plot. Additionally, the system adopts the same visual identification techniques as XGobi. The approach contributes the display of extra design space information by showing the interaction between multiple objective functions located in the design space focused on minimizing the objectives (i.e., Pareto frontier). A plot utilizing glyphs to represent points and trends in three-dimensional space is shown in Figure 7.

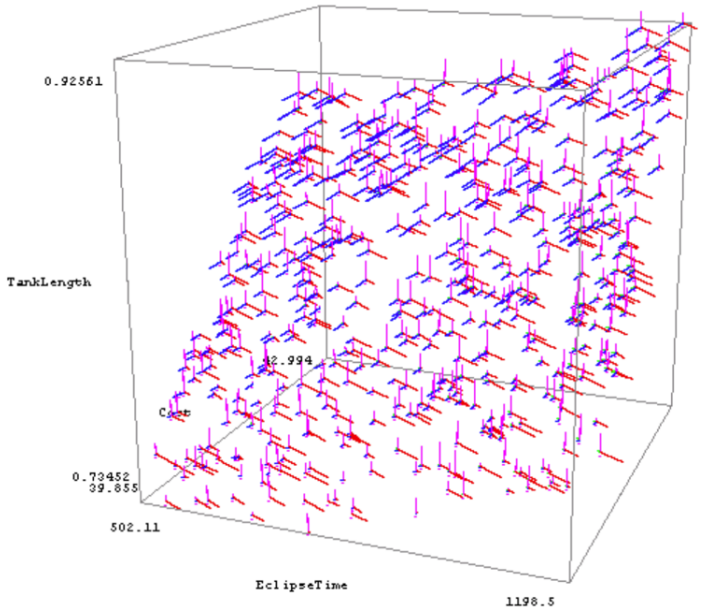

Figure 7. Glyph plot of design space [9]

The glyph plot visualization can provide a better understanding of the design and performance spaces while decreasing optimization times. However, the extra design space information does not represent the specific relationships between design variables. In addition, interpretation of 3D glyphs on a 2D computer monitor can be difficult to do.

Stump et al. also developed the ARL Trade Space Visualizer (ATSV) which focused on displaying tradeoffs within a design space [10]. ATSV takes advantage of common techniques such as glyph plots, scatter plots, and parallel coordinate plots to visualize uncertainty in design variables. Utilizing these techniques, ATSV provides a designer with a variety of visuals to explore engineering data. To generate new design concepts, Yukish et al. [11] incorporated an exploration engine based on preferences set by a designer that modify the visual techniques employed by ATSV. Furthermore, the exploration engine has the potential to link model geometry. Figure 8 shows the extended ATSV's ability to display model geometry associated with the process of design exploration.

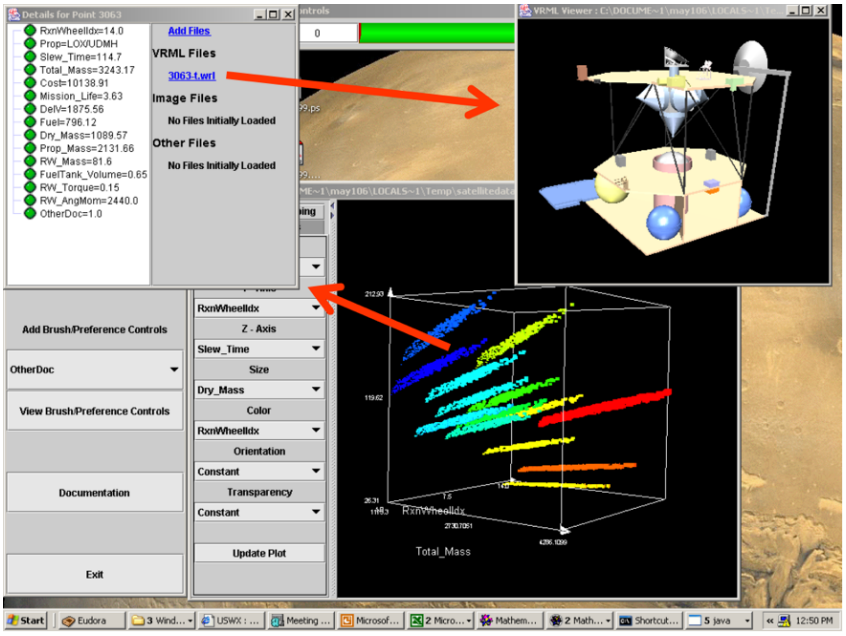

Figure 8. Extension of ATSV [11]

Additionally, the interface gives an engineer or designer the option to adjust the value of design variables and their associated impact on the objective function. However, the visualization techniques employed by Yukish et al. fail to visualize more than three variables at any given instance.

Kanukolanu, Lewis, and Winer also visualized trade-offs in design variables using their developed tool BrickViz [12]. Similar design points were termed a "brick" so the designer could reduce the time spent investigating by quickly visualizing undesirable designs. The brick properties were evaluated using a Monte Carlo simulation encompassing design variables shared by different subsystems. Each axis in a three-dimensional visual represented a single design variable composed of the brick, which in turn decreased the number of trade-off decisions but again limited the visualization to three dimensions.

To improve concept selection, Gurnani et al. used the Hypothetical Equivalents and Inequivalents Method (HEIM) [13]. HIEM visualizes a three-dimensional feasible design space where each axis represents a design attribute. When there is more than one preferred design alternative, the visualization techniques are coupled with indifference point analysis to determine the robustness of a resulting solution as well as identifying design constraints needed to obtain an optimal robust design alternative. However, the feasible space visualization was limited to a maximum of three dimensions.

Multiobjective optimization problems commonly contain designs with conflicting objective functions. Nagrath et al. used multiobjective optimization and visualization to determine trade-offs between conflicting design objectives [14]. The approach aided in determining the appropriate operating conditions from a feasible region of optimal design solutions. The method developed by Nagrath et al. was improved upon using an interactive

visualization for smart Pareto frontiers (s-Paretos) [15]. The s-Paretos present the designer a more effective visual as the frontier quantifies and displays only "good" design solutions.

A commercial market of software tools used to visualize complex data transpired from the potential cost savings for design companies. A series of software packages created by Tecplot, Inc., provides engineers visual solutions for analyzing complex datasets and computational fluid dynamics data [16]–[18]. Phoenix Integration, Inc. developed ModelCenter to provide engineers the ability to design and create an engineering process. In addition, the software allows designers to visual explore a design space while being able to analyze performance trade-offs [19]. To aid the Six Sigma design process, Isight was created by Dassault Systemes to offer lean manufacturing focused techniques such as design of experiments alongside a three-dimensional real-time interaction visual depicting trade-offs between design variables [20]. Tableau Software created tools able to deploy on desktop, server, and mobile. Additionally, the software allowed users to drag and drop data into the application, while being able to quickly create and compare multiple visual representations using their "VizQL" technology [21]. Lastly, Spotfire was designed by TIBCO Software Inc. to provide visualizations for understanding trends in large amounts of data [22].

Understanding high dimensional design data is a very challenging problem. Prior presented research describes a variety of visualization techniques geared towards understanding complex design spaces. A well-visualized design space offers valuable insight to the inherent variable relationships composing a system, but can prove difficult to navigate and analyze. Methods like principal component analysis (PCA) have been used to reduce dimensional complexity of data. However, the accuracy of PCA is bound to linear problems and is sensitive to the relative scaling of the original variables. Techniques are needed to extract

n-dimensional relationships amongst complex data while presenting an easy to interpret visual representation. Non-linear manifold learning techniques offer a promising alternative to handle the reduction of dimensional complexity and have been modified to present a visual interpretation of underlying data.

Self-Organizing Maps

Tuevo Kohonen capitalized on the cerebral cortex's ability to efficiently process and assimilate vast amounts of different input data types to develop a subset of an artificial neural network known as the self-organizing map (SOM). The unsupervised network is commonly a two-dimensional map able to visualize high-dimensional data by projecting it onto a two-dimensional space. Kohonen's SOM [23] utilizes an unsupervised competitive winner-takes-all learning strategy to project multidimensional data onto a two-dimensional hexagonal lattice structure while preserving the natural topology of input data. The result of the SOM is a trained network with similar designs grouped on a two-dimensional nodal map without any prior knowledge of the input data and associated relationships. Figure 9 demonstrates the pattern recognition similarities between the cerebral cortex and the SOM. The example SOM was trained on a variety of data types like art, information, pictures, and sounds. For example, the input data constructing sound is widely different from that needed to formulate pictures. A closer look at the trained map shows the unsupervised networks ability to group similar data types. Pictures, digital, and art can all be seen closely grouped together as they all are media types. The unsupervised learning process is not always perfect. Input data can be incorrectly grouped as seen with the image data, which should be grouped with visual inputs.

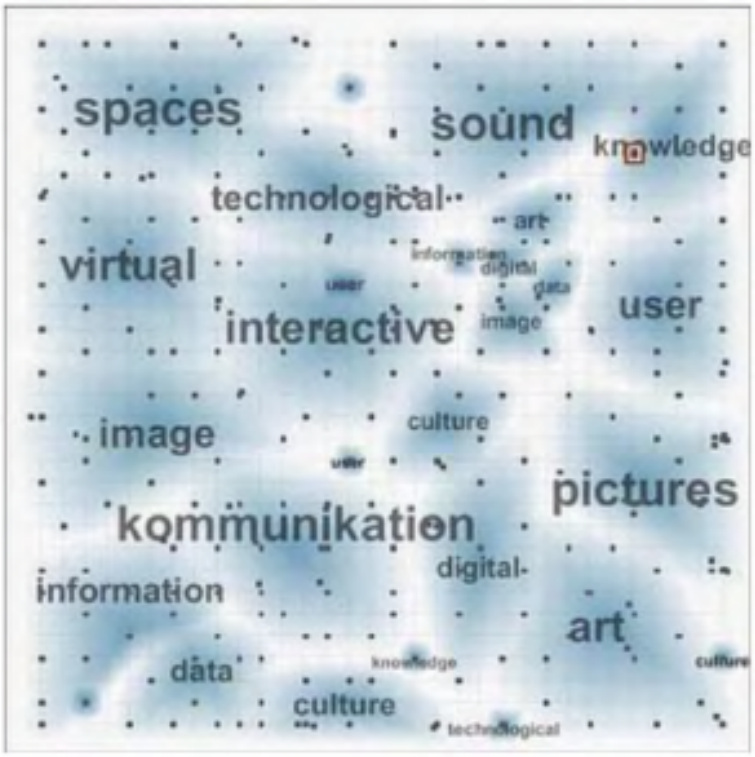

Figure 9. Trained SOM displaying various input types [24]

Three layers comprise the neural network: input, computation, and output. An input layer is used as a concrete base to send data objects to the computational layer where nodes are assigned a k-dimensional weight vector and corresponding set input. Throughout the iterative process, nodal weight vectors are used to statistically fit the data set projected onto the output map. The objects sent from the input layer are then used as input to an activation function in the computational layer, which results in a vector projected on to the output layer. First, the map training randomly selects an input data point to compute the Euclidean distance between the point and nodes. The nodes are not placed at random in an arbitrary space, but instead equidistant from one another in a rectangular or hexagonal manner. There are other nodal arrangements used in practice, but these are the most common. Once all distances have been computed between the input vector and nodes, a "winning node" is determined. The

winning node is established as that with the smallest distance to the input point, therefore it can be inferred as most like the input data point. An illustration of the "winning node" determination process can be seen in Figure 10.

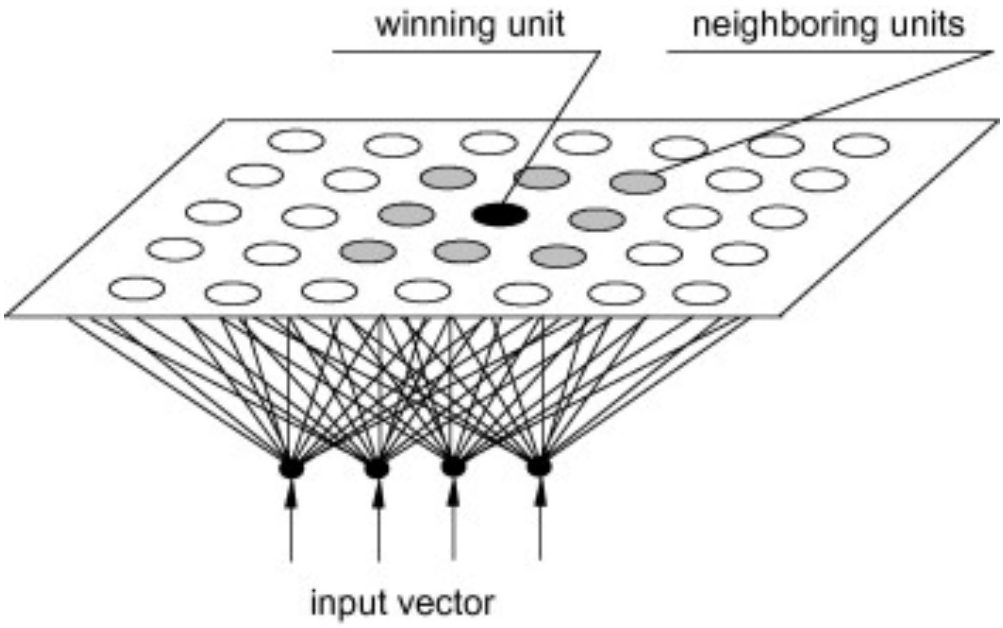

Figure 10. Diagram of the Self-Organizing Map [25]

The map maintains data topology by refining nodal weight vectors using a neighborhood function. A neighborhood surrounds the "winning node" that encompasses a group of neighboring nodes. Nodes nearest to the "winning node" are then modified to be more like the input data point established at the "winning node". The closer the neighboring node, the more its weight vector will be modified.

The SOM visualization technique has been employed in many fields using varying data types. The computational process needed to train a SOM will be explained in the following chapter. The next few sections describe additional techniques applied to the SOM to extend visualization capabilities and structure properties.

U-Matrix

The SOM maintains the "natural" relationships amongst the input data vectors within the dataset, therefore a great number of visualizations can be obtained. A standard method used to visual represent the output of a SOM is a U-Matrix. The size of a U-Matrix is larger than its corresponding SOM. In detail, the U-Matrix contains a total of (2*m*-1) * (2*n*-1) nodes than that of an (*m* x *n*) dimensional SOM lattice structure. An expanded U-Matrix map illustrates the connection strength of a node and all associated neighboring nodes using Euclidean distance.

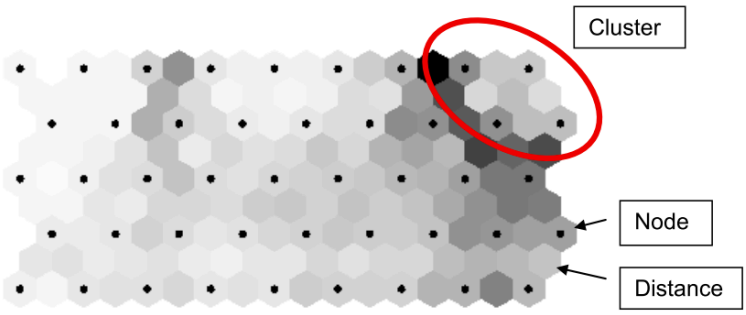

Figure 11. U-Matrix representation of a trained SOM [26]

A representative U-Matrix, as seen in Figure 11, shows the distance differences associated with a node's neighbors, where the individual nodes themselves are signified with a black dot. Hexagons without a black dot represent the Euclidean distance between neighboring nodes in the lattice structure. Additionally, a grayscale coloring scheme can be used to help identify the distance values with the darker colors representing a greater distance. It is possible to identify a cluster as nodes with very similar shades between one another. However, the investigator is solely responsible for identifying such clusters. Consequently, identifying clusters from Euclidean distance measures can be a cumbersome and difficult task while failing to accurately portray arbitrarily shaped data.

Hit Histogram

Trained SOMs tend to have a varying number of data samples located in the map nodes. To understand the distribution of the dataset, investigators commonly use hit histograms to identify the number of times a node is the best matching unit (BMU) for an input sample. Many visual variations of the hit histogram exist with the majority focusing on changing the identifying node marker. For example, Figure 12 uses a marker that changes size in proportion to the number of BMUs within a node. The map can then be generally interpreted as nodes with large black hexagonal markers containing a greater amount of BMUs versus smaller black hexagons. Nodes lacking a black hexagon are considered empty which correspond to zero BMUs.

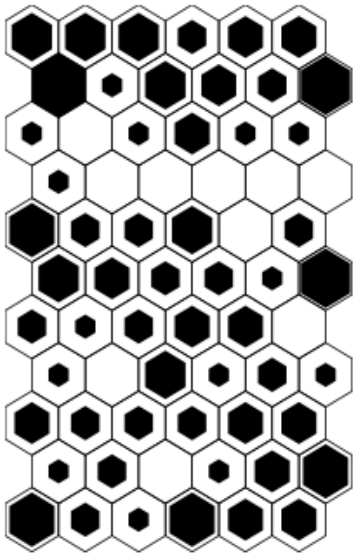

Figure 12. SOM hit histogram for Iris dataset [27]

Although color and shape markers can provide an insight to the data distribution, the exact number of BMUs cannot be interpreted. However, text labels can be overlaid on the node to denote the exact number of BMUs.

Contextual Self-Organizing Maps

An extension of the SOM is the contextual self-organizing map [28]. The training procedure is the same as the SOM, but an additional step is performed to place contextual labels onto the trained map. Once the SOM has been trained, a contextual phase projects the data points onto the map a final time. More specifically, the winning node is found once again, but unlike before the neighborhood radius is not updated. Instead, the winning node is assigned a label. Contextual labels provide a valuable visual by depicting the nodes in accordance to descriptions such as images, text, or a data point's objective function value instead of numerical weight vectors. A basic CSOM tasked with identifying types of animals is shown in Figure 13. Variables such as size, physical traits, and abilities were used to identify the animals. As shown, the CSOM inherently categorizes the animal, which helps depict the general groups of animals: birds, predators, and peaceful species. More importantly, the CSOM creates the map visualization solely based on identified animal traits without knowing prior knowledge of the type of animal. The defined groups are generalized representations of the animals belonging to them, with each group possessing bounding traits. For example, eagles, hawks, and owls are located nearest to the predator group as they have overlapping traits, but are most like the traits contained in the representative bird group

dog	dog	fox	fox	fox	cat	cat	cat	eagle	eagle
dog	dog	fox	fox	fox	cat	cat	cat	eagle	eagle
wolf	wolf	wolf	fox	cat	tiger	tiger	tiger	owl	owl
wolf	wolf	lion	lion	lion	tiger	tiger	tiger	hawk	hawk
wolf	wolf	lion	lion	lion	tiger	tiger	tiger	hawk	hawk
wolf	wolf	lion	lion	lion	owl	dove	hawk	dove	dove
horse	horse	lion	lion	lion	dove	hen	hen	dove	dove
horse	horse	zebra	cow	cow	cow	hen	hen	dove	dove
zebra	zebra	zebra	cow	cow	cow	hen	hen	duck	goose
zebra	zebra	zebra	cow	cow	cow	duck	duck	duck	goose

Figure 13. Contextual SOM representing separation of hunters, peaceful species, and birds [28]

Generalizing designs of systems tend to be less visually apparent as the animal dataset depicted in Figure 13. In industry, a fair number of designs are generated based off returning a profitable return on investment, therefore a designer may use the net present value (NPV) of a design as a contextual label. To help distinguish value functions with large ranges, a new four-map visual was introduced by Richardson et. al [29] which assigns each statistical measure to its own node and creates a fourth map displaying text labeling the number of times a node was the BMU of a design alternative. An example of the four-map visual is shown in Figure 14. Three color gradients, interpolated from each statistical evaluation's numerical limits, were assigned to their corresponding map. The CSOM used color gradients: blue, green, and red to represent each individual node's respective mean, minimum value, and standard deviation.

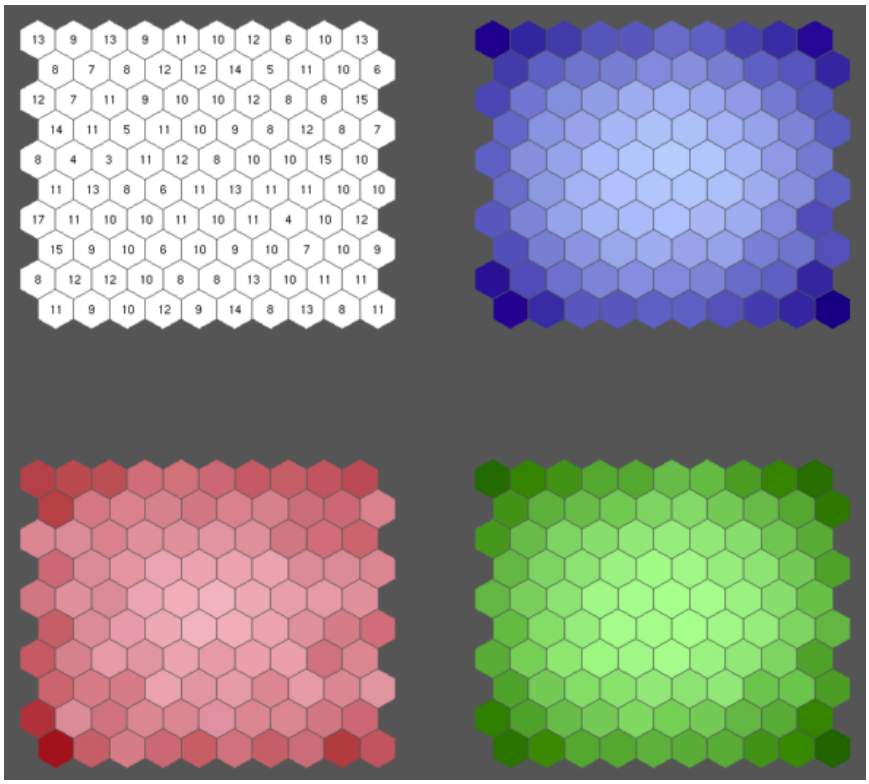

Figure 14. Four-map visual of Contextual Self-Organizing Map [29]

The color gradients employed by Richardson et al. across the four-map visual supplied the means to quickly determine defining characteristics such as modality and curvature of the design. Furthermore, the color gradient can be used as a statistical measure of a performance characteristic decided by the investigator. For example, an engineer tasked with designing a new rocket engine may use minimum mass or maximum thrust performance metrics to serve as a label for the output map. Unfortunately, an increased amount of computational time is produced by applying contextual labels and interpolating colors based off their statistical measures. Additionally, large range objective function values can over-shadow nodes with less dramatic values due to the normalization of the color range values. Therefore, the map visual can lead investigators to overlook vital designs while analyzing the countless alternatives in the space.

Self-Organizing Maps used in Optimization

A significant amount of research has been dedicated to utilizing self-organizing maps for optimization purposes [30]–[34]. Self-Organizing Maps for Optimization (SOMO) was developed by Su et al. to solve and visualize optimization problems [35]. Instead of using Euclidean distance to determine the winning node, SOMO modified the training process to use the optimization objective function. Hence, the distance between nodes are then evaluated as the objective function value of the design points. The trained map is visualized in three dimensions by plotting the two-dimensional lattice on the x-y plane and each node's objective function value on the z-axis. The SOMO approach provides a visual of the objective function's structure and groups similar regions of design points. However, the method is limited to continuous objective functions and was not tested on problems with dimensions greater than

30. A similar method of displaying the results of the self-organizing map was developed by Milano et al., which populated a trained map using weight vectors to represent design variables influencing the objective function. To display the objective values each node must calculate the objective function using the nodal weight vectors. Consequently, this approach requires the dataset to contain an objective function value for each design point.

Obayashi and Sasaki employed SOMs to optimize the design of supersonic wings and fuselages [36]. Training data was obtained by analyzing each design using computational fluid dynamics simulations. The simulations provided results needed to determine an objective function based off the drag, bending moment, and pitch moments. Contextual Self-Organizing Maps were used to group the designs and presented images of the design for contextual labels. These contextual labels, shown in Figure 15, vividly illustrates the distinct designs differences with the bottom right and top left corners of the map containing the two most extremes. A grayscale coloring scheme was used to denote minimum objective function values as a lighter color and maximum values as a darker color.

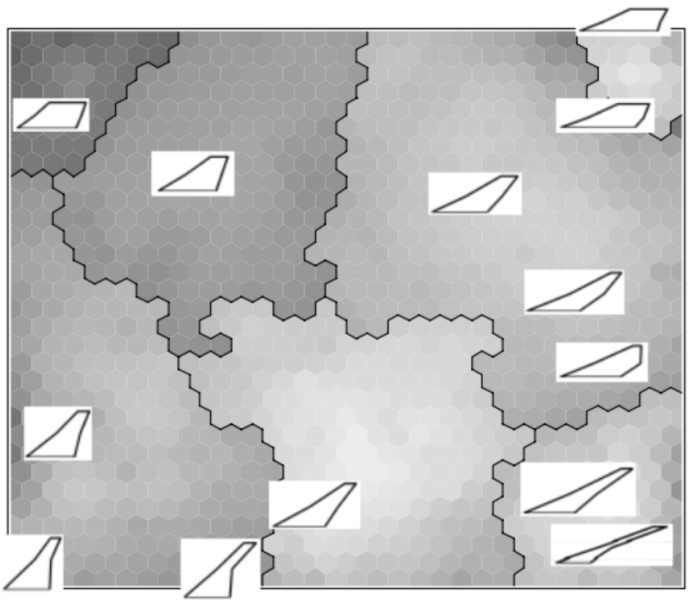

Figure 15. Contextual SOM of supersonic wing designs [36]

In addition, the researches modified the visual representation of the output map by isolating the individual objective values (i.e., drag, bending moment, pitch moment). Separating the objective values helped provide the engineer an understanding of how each objective influences the overall design and the relationships between design variables.

Complex designs often contain large databases of variable and performance information, which can be leveraged to identify similar groupings of design alternatives. The following section describes a variety of cluster analysis methods able to be applied to the SOM, which classifies the alternatives contained in a design space.

Clustering the Self-Organizing Map

Similar data groupings can be revealed from the trained output map using visualization techniques such as the U-Matrix and hit histogram, but these methods are not inherently cluster analysis methods. Large databases often contain data objects with unknown classification labels. Consequently, assigning objects a classification label can be difficult and costly. Clustering algorithms provide a means to organize vectors of data into "natural groups", in which a set of points belonging to a cluster are more like one another than those belonging to a different cluster [37]. Cluster analysis is commonly used in many fields ranging from biology to machine learning [38]. Furthermore, five major categories of clustering exist: partitioning, hierarchical, density-based, grid-based, and model-based [38]. Past research has combined the visualization properties of SOMs with numerous clustering methods to form a two-level SOM-based approach [39]–[43]. The concept of the two-level approach, illustrated in Figure 16, designates the first level to training of the data with the SOM and the second level to clustering the output map using cluster analysis methods.

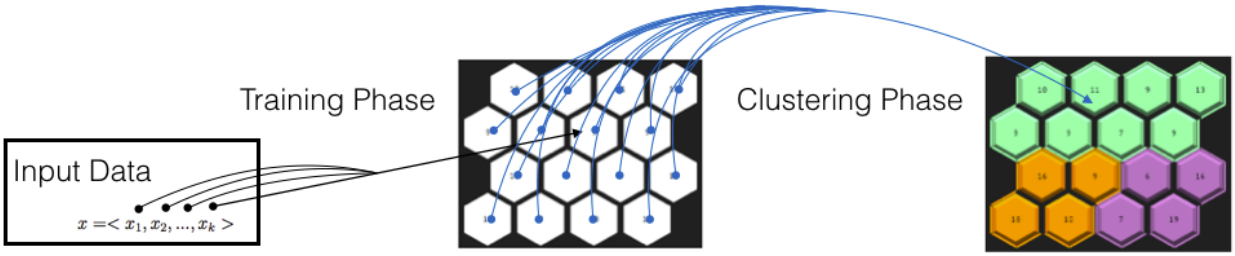

Figure 16. Clustering the SOM using a two-level approach

The input data projected onto the SOM during the training phase assimilates the data while maintaining the data topology, but fails to create precisely defined clusters. However, the nodes containing the data points within their neighborhood provide a great starting point and reduce the amount of computation for cluster analysis methods to accurately classify the data.

Partition Clustering

The most fundamental clustering method is partitioning, which constructs *k* partitions of data consisting of *n* number of objects. Most partitioning methods are distance-based and typically implement *exclusive cluster separation* in which case each object must belong to a cluster, but other methods (i.e., fuzzy partitioning) allow for an object to belong to multiple clusters. All partitions represent an individual cluster with $k \leq n$, such that each partition must contain a minimum of one object. The initial partitions are further improved upon by moving objects from one cluster to another through an iterative relocation technique. The relocation process then produces clusters with similar attributes by optimizing an objective partitioning criterion. *K-means* is a common and well-studied dynamic partitioning method. As a partitioning method, *k-means* reassigns data points throughout each iteration cycle to minimize a distance criterion between centroids of *k* partitions. Common centroid determination

functions (i.e., Manhattan and squared Euclidean distance) are used to compute the centroid in terms of the median and mean value of its encompassing data points. The partitioning nature of the *k-means* algorithm can be seen by the difference in colored areas in Figure 17.

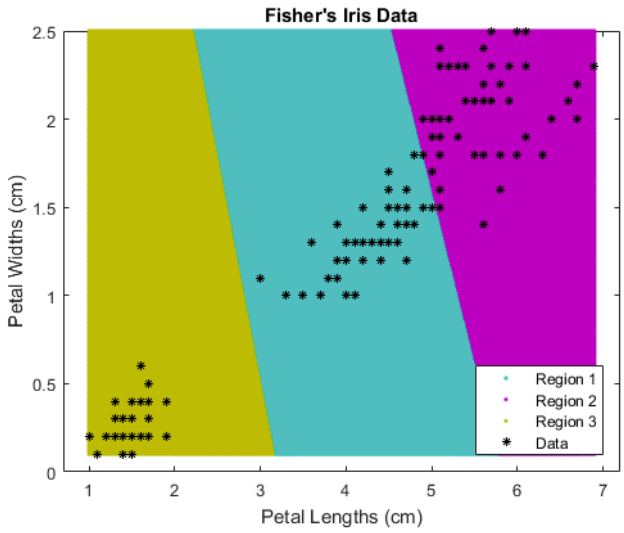

Figure 17. Example K-means clustering using Iris dataset [44]

Nevertheless, there are drawbacks to the K-means algorithm. The number of partitions must be specified in advance, which require the user to have prior knowledge of the data set's topology. Additionally, the method is not suited for identifying clusters with non-spherical shapes, large size deviations, or varying densities as seen in most industrial applications. Lastly, outliers can abnormally deviate the centroid of a cluster, leading to a misrepresentation of similar attributes.

Hierarchical Clustering

Hierarchical clustering is a relatively old technique geared toward the creation of a hierarchical decomposition of clusters. The method can be classified into two types, agglomerative and divisive. Agglomerative clustering begins by assigning each data point as

its own cluster. All clusters will then iteratively merge two clusters, based on a proximity metric, until one cluster remains. Opposite of the agglomerative approach, divisive clustering starts with a single cluster possessing all data points which then splits into smaller clusters until a termination condition is met or each data point is its own cluster. No matter the approach, a tree structure known as a dendrogram is used to visualize how clusters are grouped throughout the clustering process. For example, an agglomerative dendrogram, as seen in Figure 18, displays clusters as points on the graph with each connecting line illustrating the path of two clusters merging. As the clusters begin to grow by merging with one another, the y-axis denoting the similarity distance becomes larger, therefore the two clusters containing {3, 6, 7} and {2, 10, 5, 8, 9, 1, 4} are the least similar.

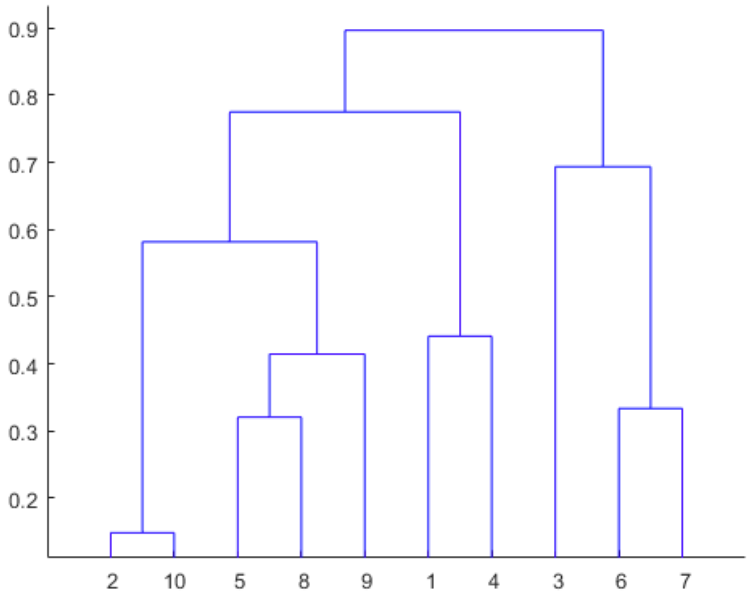

Figure 18. Dendrogram visualization [45]

Both hierarchical methods can integrate distance-based, density-based, or continuity-based measures. A total of four essential linkage algorithms utilize distance-based measures such as minimum and maximum, mean, and average distance. The minimum distance measure

between clusters is often used and regarded as nearest-neighbor clustering. The method uses a single-linkage algorithm to terminate the clustering process when the distance between nearest-neighboring clusters exceeds a defined threshold. Opposite of the minimum distance, farthest-neighbor clustering uses a complete-linkage algorithm to terminate the clustering process when a set threshold is exceeded by the maximum distance between nearest clusters. Furthermore, the algorithm produces substantial results when used with compact and equal sized clusters due to the nature of the algorithm minimizing an increase in cluster diameter. Single and complete-linkage algorithms evaluate distance extremes, therefore are not relied upon for clustering noisy data sets. Instead, mean and average distance measures are employed to mitigate the sensitivity induced from outliers. Each linkage algorithm has advantageous characteristics, but all suffer from the fact that hierarchical clustering cannot correct erroneous merges or splits during the iterative process.

Past research has applied the agglomerative hierarchical technique with the trained SOM to cluster similar points with one another. Murtagh used an agglomerative contiguity-constrained clustering method to classify the trained SOM and merged neighboring neurons based on a minimal distance criterion [41]. Kiang expanded on Murtagh's work by employing a minimum variance criterion to achieve better results [42]. Since both Murtagh and Kiang used a distance criterion, employable data sets are constrained to hyper-spherical and hyper-ellipsoidal shaped clusters. Vesanto had taken a different approach by only using the inter-cluster distance as the evaluation criterion for a cluster merge [43]. However, Both Lampinen and Vesanto require the investigator to know the number of desired output clusters. This can be an issue if no prior knowledge of the dataset exists.

Density-Based Clustering

Partitioning and hierarchical clustering methods have many differences, yet both use distance measures designed to discover spherical-shaped clusters. Consequently, the methods have issues identifying arbitrarily-shaped clusters, therefore density-based clustering methods were created to model clusters as dense regions of space separated by sparser regions [38]. Furthermore, these algorithms can divide a data space into a hierarchy of clusters or multiple exclusive clusters. One of the most popular methods is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [46]. The center-based approach classifies points into three types: core, border, and noise. A core point is classified as one located in the interior of a dense region, and are established if its user-defined neighborhood radius contains at least the user-specified minimum number of points. The edge of a dense region is bound by the neighborhood surrounding the core points. Hence, points located on the edge of the neighborhood radius are considered border points. Points that are neither a core point nor a border point are considered noise. The goal of the algorithm is to continually grow a cluster as long as the number of points within its established neighborhood exceeds a user-defined threshold. For example, smaller dense regions can be formed into a larger cluster if a region's border point is within the neighborhood radius of another region's core point. Compounding core points connected in a density-based cluster can then identify clusters with arbitrary shapes and eliminate outliers since the cluster is based from the local core points' neighborhood.

Wu and Chow proposed a SOM-based clustering algorithm utilizing the agglomerative hierarchical clustering process and a novel cluster validity index [47]. The purpose of using a cluster validity index is to find "compact and well-separated" clusters [48]. The validity index proposed by Wu and Chow is comprised of two types of criterion: cluster compactness and

cluster separation. Cluster compactness evaluates the similarity of data points within a cluster's neighborhood radius. It can be inferred that the higher number of points within a cluster's neighborhood yields a higher density of points containing similar attributes. The cluster separation is constructed of the density of the area between clusters (i.e., inter-density) and the distance of the nearest neighboring pair point over the density of the region separating one another. The product of cluster intra-density and separation yield the overall clustering validity index "Composing Density Between and Within Clusters" (*CDbw*). The clustering algorithm proposed by Wu and Chow used the *CDbw* validity index as the merging criterion in the agglomerative hierarchical process and is summarized as follows:

1. Input data is trained by the SOM.
2. Preprocess the trained input data.
3. Cluster the SOM using agglomerative clustering. Compute the merging criterion (*CDbw)* for all direct neighbors.
4. Compute the global *CDbw* for all clusters before each merge until only two clusters exist or merging is no longer possible.
5. Determine the optimal clusters according to the *CDbw* as a function of number of clusters.

The *CDbw* validity index serves two purposes. First, it finds the pair of neighboring clusters with the strongest tendency to be clustered. Second, it determines the optimal number of clusters. The lowest *CDbw* indicated the clusters were the least compact and separate, therefore the most viable merge. Furthermore, only neighboring clusters can be merged. For example, neuron A in Figure 19a contains eight direct neighbors. Therefore, the neighbor

containing the minimum *CDbw* validity index out of the eight will merge with neuron A. The number of direct neighbors decreases as the number of neurons within a cluster increase, yet if a pair of clusters are not direct neighbors they cannot be merge. This is shown in Figure 19b and c.

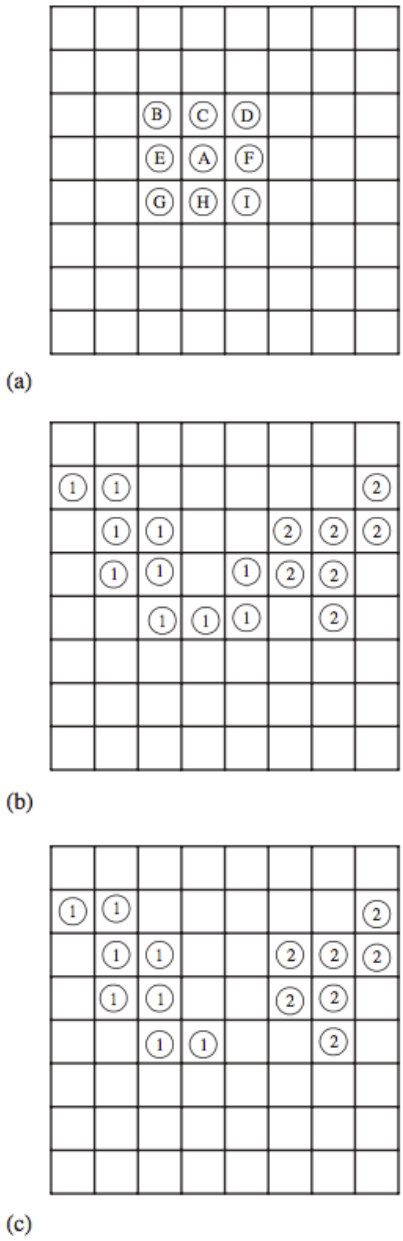

Figure 19. Visualization of cluster neighbors (a) neuron A contains eight direct neighbors: B-I; (b) clusters 1 and 2 can be merged into one cluster since the two clusters are direct neighbors; (c) clusters 1 and 2 are not direct neighbors therefore cannot merge [47]

Once the clustering process has finished, the optimal number of clusters can be determined. Previously, the minimum *CDbw value* indicated a cluster pairing with the strongest tendency to merge. Opposite of merging criterion, the maximum *CDbw* for a given number of clusters determines the optimal clusters. Wu and Chow tested the proposed clustering algorithm against a series of standard machine learning datasets, one of which was the Iris dataset [49]. The Iris dataset contains three classes, each with 50 sample points, distinguishing a type of Iris plant. Each sample point contains four variables: sepal length (cm), sepal width (cm), petal length (cm), and petal width (cm). The first classification is linearly separable from the other two classes, but classes two and three overlap and aren't linearly separable. The researchers global *CDbw* index result for the Iris data set is shown in Figure 20.

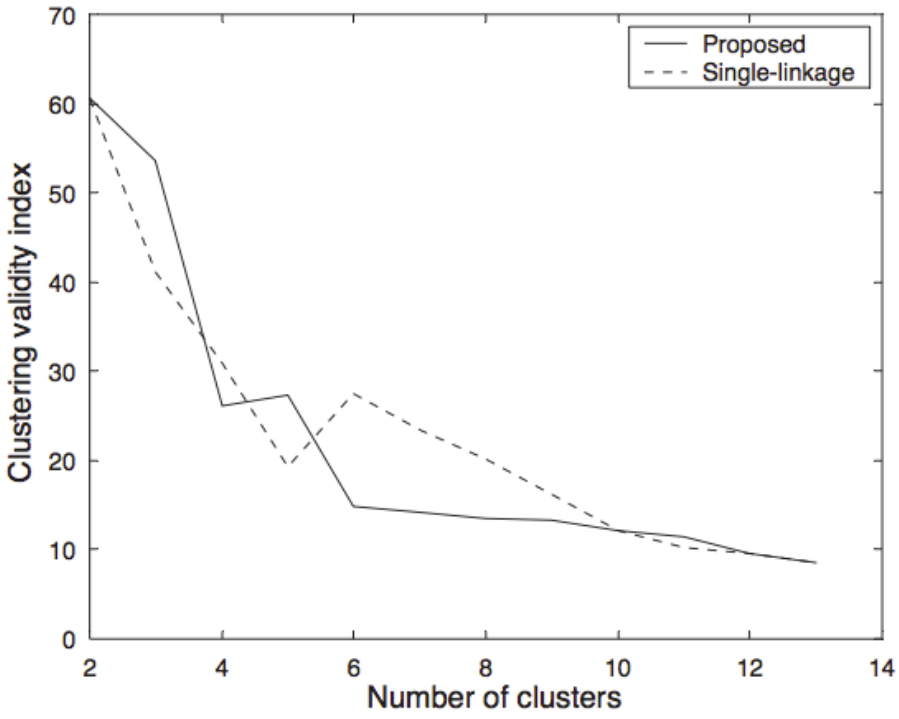

Figure 20. The CDbw as a function of the number of clusters for the iris data by the proposed algorithm (solid line), and the single-linkage clustering algorithm (dashed line) on the SOM [47]

Figure 20 shows the CDbw slope is constantly changing between each iteration and does not resemble a smooth curve. The peaks and valleys are produced from the clusters' overall compactness battling the amount of separation between one another, but it can be clearly seen there are two optimal clusters with a *CDbw* value of approximately 60.

Research Questions

The literature presents three general approaches used to aid the understanding of multi-dimensional data. The first approach is to use multiple visualizations, which obligates the designer or engineer to formulate a mental depiction of inherent relationships when switching between multiple sets of variable visualizations. The second technique focuses on creating a structure able to provide a visual representation that can be directly interpreted by the investigator. The final approach uses clustering methods to organize vectors of raw data into "natural groups". As was shown in this chapter, many issues reside with each of these techniques. This has led to the following research questions addressed in this thesis:

1. **How can cluster analysis methods be applied to self-organizing maps to classify design data into optimal groups containing similar variable relationships?**

   Large databases often contain data objects with unknown classification labels. Clustering algorithms provide a means to organize vectors of data into "natural groups" based on similar characteristics. This research looks at how to use density-based clustering algorithms, suited for

arbitrarily-shaped data found in engineering applications, to classify design data into optimal groups of clusters that segment the design space into manageable sections.

2. **Can the visualization of self-organizing maps be improved to prevent engineers from overlooking influential design points?**

   Visualization methods, such as self-organizing maps (SOM), offer powerful methods to visualize n-dimensional data and produce simple to understand representations that quickly highlight trends to support decision-making. However, a drawback to using SOMs is the clustering of promising points with predominately less desirable data. This work looks at providing a directly interpretable visual representation that classifies trained SOM output data into prominent clusters.

This next chapter contains a journal article submitted to the IEEE Transactions on Knowledge and Data Engineering. A modified cluster validity index is proposed to provide a better means of handling data associated with large-complex systems and an optimal number of meta-clusters segmenting the design space. The paper will describe the algorithm created to establish these meta-clusters through the development of several computational metrics involving inter and intra cluster densities. In addition, details regarding the training and modified density-based clustering of the SOM can be found in the paper.

CHAPTER 3: VISUALIZING ENGINEERING DESIGN DATA USING A MODIFIED TWO-LEVEL SELF-ORGANIZING MAP CLUSTERING APPROACH

Adam Kohl [#1], Eliot Winer [*2]

[#*] *Department of Mechanical Engineering, Human Computer Interaction, Virtual Reality Applications Center, Iowa State University*
1620 Howe Hall, Ames IA 50011 United States

[1] adamkohl@iastate.edu

[2] ewiner@iastate.edu

Designers of large and complex engineered systems are constantly in need of decision-making aids to sift through the enormous amounts of data produced through simulation and experimentation. Visualization methods, such as self-organizing maps (SOM), offer powerful methods to visualize these data with the goal to produce simple to understand representations that quickly highlight trends to support decision-making. A drawback to using SOMs is the clustering of promising points with predominately less desirable data. This paper applies a cluster analysis technique to SOMs to segment a high-dimensional dataset into "meta-clusters". The paper will describe the algorithm created to establish these meta-clusters through the development of several computational metrics involving inter and intra cluster densities. A case study of a satellite design problem is presented using this algorithm to show how optimal designs can be easily located within the visualization for aiding in decision-making.

Introduction and Background

Complicated large-scale designs have become more and more prominent bringing unique challenges with them. Designers now have access to vast amounts of data that have the potential to unearth trivial insights to vital design information. However, the majority of this "big data" is riddled with complex variable relationships. For example, system designs can

contain thousands of components, all of which are uniquely designed. Factors such as material properties, geometry, and manufacturability of parts and assemblies must be evaluated and eventually decided upon by designers and engineers. All the while an understanding of how each component effects the overarching design must be maintained. Designing a system at a single level of a system is a difficult enough task, but when multiple subsystems interact with one another the problem becomes even more complex. Understanding these defining design characteristics is critical to making suitable trade-off decisions. Creating effective visual representations of a system's design space, or other data, offers designers valuable insight to the characteristics composing an overall system. Past research has examined a variety of techniques to visualize n-dimensional design data [1]–[4]. Many of these methods claim to visualize n-dimensions. In reality, three or fewer design variables are visualized as the product of dimensional reduction (i.e., setting all other variables to constant values). Limiting the number of variables obligates the investigator to formulate a mental depiction of inherent relationships when switching between sets of variable visualizations. Therefore, methods are needed to extract variable relationships while maintaining the topological structure of the data. Moreover, these methods must produce a visual representation that can be directly interpreted by the investigator.

Self-Organizing Maps (SOMs) were developed to produce such representations of large datasets. Tuevo Kohonen capitalized on the cerebral cortex's ability to efficiently process and assimilate vast amounts of different input data types to develop a subset of an artificial neural network known as the SOM. Kohonen's SOM [5] utilizes an unsupervised competitive winner-takes-all learning strategy to project multidimensional data onto a two-dimensional hexagonal lattice structure while preserving the natural topology of input data. Further advances have

been made to the traditional SOM in terms of visualization, such as the U-Matrix, hit histogram, and component plane [6], [7]. Richardson et al. [8] proposed a visualization technique that breaks up the original map into three additional maps to present statistical differences regarding contextual labels assigned to each output node. The visualization technique is based off an extension of the SOM called the contextual self-organizing map (CSOM). Training the CSOM is the same as the SOM, but an additional step is performed to place contextual labels onto the trained maps. These maps give a designer a visual context to the data set's natural topology by highlighting the nodal performance amongst the maps. Similar data groupings can be revealed from the trained output maps using visualization techniques such as the CSOM, but these are not inherently cluster analysis methods.

Large databases often contain data objects with unknown classification labels. Consequently, identifying these labels can be difficult and costly. Clustering algorithms provide a means to organize vectors of data into "natural groups", in which a set of points belonging to a cluster are more like one another than those belonging to a different cluster [9]. Five major categories of clustering exist: partitioning, hierarchical, density-based, grid-based, and model-based [10]. Past research has combined the visualization properties of SOM with numerous clustering methods to form a two-level SOM-based approach [11]–[15]. The concept of the two-level approach designates the first level to training of the data with the SOM and the second level to creating the output map using various cluster analyses. One such approach was proposed by Wu and Chow [16]. Wu and Chow utilized the agglomerative hierarchical clustering process to develop a novel validity index based on cluster density to identify "compact and well-separated" clusters. This paper introduces a modification of Wu and Chow's clustering algorithm to place more importance on neighboring cluster influence in

order to discover large cluster regions, meta-clusters, with similar variable relationships in the design space of large-complex systems. Furthermore, automatically segmenting the design space separates design alternatives into manageable sections to prevent investigators from overlooking influential designs amongst countless alternatives in the design space. Classifying design alternatives by associated variable characteristics will offer valuable information for designers for a variety of activities including making important design decisions or locating a starting point from which to being a formal optimization routine.

Design Space Visualization

As the design of complex engineered products becomes increasingly difficult, the number of independent and dependent variables inherently grows. It is common to have systems of equations representing these products with tens to thousands of design variables. Navigating and examining this design space is truly a formidable task. Even before a formal optimization method can be run, a designer needs to understand the underlying variable relationships entangled within n-dimensional data. Multidimensional data visualization can offer tools and methods to aid in this.

One such approach was to improve a designer's interaction with an objective function throughout the optimization process by a paradigm termed visual design steering [2]. Visual steering first "ranks and reduces" design variables and constraints based on their respective influence on the objective function. Design variables and constraints with little or no impact are removed from the problem for visualization purposes. The researchers utilized graph morphing, a technique to visualize n-dimensional data in two or three dimensions, to envision the ranked design tradeoffs by assigning graphical slider bars to a variable's axis to view a

real-time change in objective function as each slider manipulates the value of a particular variable. Being able to view an objective function morphing in real-time was a novel approach, but only allowed for a maximum of three variables to be pictured at a time.

Stump et. al. further expanded upon the dynamic data visualizing system XGobi [17] by using the shopping paradigm [18] to evaluate multidimensional visualization. Instead of running countless simulation iterations, the designer extracts the design variables and manipulates their ranges while an optimization routine is running. The current design variables are then visualized in three or less dimensions. Furthermore, the optimization routine will continue solving until a convergence criteria is met or the design variables are manipulated by the designer. The visualization system's main contribution is the display of extra design space information as the interaction of multiple objectives (i.e., performance space) in the Pareto frontier illustrated on a glyph plot. This research provided a better understanding of the design and performance spaces and decreased optimization times. However, the extra design space information does not represent the specific relationships between design variables.

Stump et al. continued to build off the XGobi research by developing the ARL Trade Space Visualizer (ATSV) which focused on displaying tradeoffs within a design space [3]. ATSV takes advantage of common techniques such as glyph plots, scatter plots, and parallel coordinate plots to visualize uncertainty in design variables. Kanukolanu, Lewis, and Winer also visualized the trade-off in design variables using their developed tool BrickViz [19]. Similar design points were termed a "brick" so the designer could reduce the time spent investigating by quickly visualizing undesirable designs. The brick properties were evaluating using a Monte Carlo simulation encompassing design variables shared by different subsystems. Each axis in a three-dimensional visual represented a single design variable composed of the

brick, which in turn decreased the number of trade-off decisions but again limited the visualization to three dimensions. A well-visualized design space offers valuable insight to the inherit variable relationships composing a system, but can prove difficult to navigate and analyze. Consequently, investigators can overlook vital designs while analyzing the countless alternatives in the space. These designs often contain large databases of variable and performance information, which can be leveraged to identify similar classifications or groupings of designs.

Clustering

Similar data groupings can be revealed from an otherwise unknown prior knowledge of a database. Clustering methods provide a means to organize vectors of data objects into "natural groups", in which a set of points belonging to a cluster are alike to one another [15]. The most fundamental clustering method is partitioning, which constructs k partitions of data consisting of n number of objects. Most partitioning methods are distance-based and typically define each object to a cluster (e.g., exclusive cluster separation). However, other methods, such as fuzzy partitioning, allow for an object to belong to multiple clusters. All partitions represent an individual cluster with $k \leq n$, such that each partition must contain a minimum of one object. The initial partitions are further improved upon by moving objects from one cluster to another through an iterative relocation technique. The relocation process then produces clusters with similar attributes by optimizing an objective partitioning criterion. *K-means* is a common and well-studied partitioning method. As a partitioning method, *k-means* reassigns data points throughout every iteration to minimize the distance between centroids of k partitions. Common centroid determination functions (i.e., Manhattan and squared Euclidean distance) are used to

compute the centroid in terms of the median and mean value of its encompassing data points. The algorithm is normally run a few additional times, with different initial partition centers, to achieve a good result. Nevertheless, there are drawbacks to the K-means algorithm. The number of partitions must be specified in advance, which require the user to have prior knowledge of the data set's topology. This method is not suited for identifying clusters with non-spherical shapes, large size deviations, or varying densities as seen in industrial applications. Furthermore, outliers can abnormally deviate the centroid of a cluster, leading to a misrepresentation of similar attributes.

Hierarchical clustering is a relatively older technique geared toward the creation of a tree of clusters. The method can be classified into two types, agglomerative and divisive. Agglomerative clustering begins by assigning each data point as its own cluster. All clusters will then iteratively merge two clusters, based on a proximity metric, until one cluster remains. Divisive clustering, which can be thought of as the opposite of the agglomerative approach, starts with a single cluster possessing all data points and then splits it into smaller clusters until a termination condition is met or each data point is its own cluster. No matter the approach, a tree structure known as a dendrogram is used to visualize how clusters are grouped throughout the clustering process. For example, an agglomerative dendrogram displays clusters as points on the graph with each connecting line illustrating the path of two clusters merging. Both hierarchical methods have the ability to integrate distance-based, density-based, or continuity-based measures. A total of four essential linkage algorithms utilize distance-based measures such as minimum, maximum, mean, and average [10]. The minimum distance measure between clusters is often used and regarded as nearest-neighbor clustering, where a single-linkage algorithm is used to terminate the clustering process when the distance between

nearest-neighboring clusters exceeds a defined threshold. On the other hand, farthest-neighbor clustering uses a complete-linkage algorithm to terminate the clustering process when a set threshold is exceeded by the maximum distance between nearest clusters. Furthermore, the algorithm produces substantial results when used with compact and equal sized clusters due to the nature of the algorithm minimizing an increase in cluster diameter [10]. Single and complete-linkage algorithms evaluate distance extremes, therefore are not relied upon for clustering noisy data sets. Instead, mean and average distance measures are employed to mitigate the sensitivity induced from outliers [10]. Each linkage algorithm has advantageous characteristics, but all suffer from the fact that hierarchical clustering cannot correct erroneous merges or splits during the iterative process. Incorrect merges can lead to clusters of points with dissimilar properties. These wrongly classified clusters can mislead an investigator by masking ideal design points with the properties of poor design alternatives.

Partitioning and hierarchical clustering methods have many differences, yet both use distance measures designed to discover spherical-shaped clusters. Consequently, the methods have issues identifying arbitrarily-shaped clusters, therefore density-based clustering methods were created to model clusters as dense regions of space separated by sparser regions [10]. Furthermore, these algorithms can divide a data space into a hierarchy of clusters or multiple exclusive clusters. One of the most popular methods is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [20]. The core goal of the algorithm is to continually grow a cluster if the number of points within its established neighborhood exceeds a user-defined threshold. For example, smaller dense regions can be formed into a larger cluster if a region's point is within the neighborhood radius of another region's core point. The core points are established if its user-defined neighborhood radius contains at least the user-specified

minimum number of points. Compounding core points connected in a density-based cluster can then identify clusters with arbitrary shapes and eliminate outliers since the cluster is based from the local core points' neighborhood. Eliminating these outlying points can help remove poor design points from groups, which can narrow the focus of a designer to a more compact group of good design alternatives.

Past research has combined the visualization properties of the SOM with the advantageous attributes of clustering methods to produce a two-level SOM-based clustering approach [11]–[15]. These methods represent each cluster as groups of output neurons. To start, Lampinen employed the popular partitioning clustering technique, *k-means* algorithm, as the second layer of the two-level approach [21]. Murtagh used an agglomerative contiguity-constrained clustering method as the second layer and merged neighboring neurons based on a minimal distance criterion [13]. Kiang expanded on Murtagh's work by employing a minimum variance criterion to achieve better results [14]. Since both Murtagh and Kiang used a distance criterion, employable data sets are constrained to hyper-spherical and hyper-ellipsoidal shaped clusters. Vesanto had taken a different approach by only using the inter-cluster distance as the evaluation criterion for a cluster merge [15]. However, Both Lampinen and Vesanto require the investigator to know the number of desired output clusters. This can be an issue if no prior knowledge of the dataset exists. Halkidi and Vazirgiannis used agglomerative hierarchical clustering for the second layer to propose a clustering validity index based on inter-cluster and intra-cluster densities in order to find an optimal partition of clusters [22]. Like the DBSCAN clustering algorithm, the approach is able to handle an arbitrarily shaped cluster. Expanding on Halkidi and Vazirgiannis, Wu and Chow modified the cluster validity index [16]. Instead of only using the validity index in a global sense, Wu and Chow implemented the index on a

local level to determine the pair of clusters destined to merge throughout the agglomerative clustering process. However, the density of the regions formed between and within a cluster failed to account for the direct influence of a neighboring cluster's similar data points. Therefore, the formulations Wu and Chow [16] presented to evaluate a cluster's properties were expanded on to better account for the influence of a neighboring cluster's similar design points. Compounding these design point groupings, revealed from the trained map of a SOM, with cluster analysis methods can classify design alternatives into manageable sections to help stop designers from overlooking influential designs amongst countless options in the design space.

Methodology

The two-level SOM clustering approach was modified to better aid the visual representations of n-dimensional design data. Visualization properties of the Contextual Self-Organizing Map (CSOM) were combined with a modified density-based clustering algorithm to create larger "meta-clusters". First in this section, the process of developing a CSOM to better understand the complex design variable relationships is explained. Second, the modification of a density-based clustering algorithm is presented to further segregate the CSOM into large groups of meta-clusters. Third, a means of validation of the clustering algorithm is performed using a standard machine learning dataset. Lastly, a communication satellite design case study is used to implement the modified two-level CSOM clustering process in section one and two to better understand the design space for large-complex systems

Contextual Self-Organizing Maps

The CSOM is an extension of Kohonen's Self-Organizing Map that takes advantage of a network topology popular in neural networks. The SOM algorithm utilizes an unsupervised competitive winner-takes-all learning strategy to project multidimensional data onto a two-dimensional neural network topology. Three fundamental layers structure the SOM. The input layer handles the data coming into the network. All incoming data is sent to the underlying neural network's computation layer to serve as an input to the calculation of an activation function. Resulting vectors from the computation layer are then projected to the output layer. Each node within the computation layer contains a k-dimensional weight vector (w) and corresponding set of input vectors (x) as seen in Eqns. (1) and (2). Throughout the iterative process, every individual weight vector is used to statistically fit the data set projected onto the output map.

$$x =< x_1, x_2, \dots, x_k > \quad (1)$$

$$w_j =< w_{j1}, w_{j2}, \dots, w_{jk} > \quad (2)$$

To initialize the map, each node is assigned a random weight vector consisting of the same dimensionality as the input vector. Furthermore, the random data point selection and placement technique was used for three reasons: 1) it makes no prior assumptions to the map ordering, 2) it uses the same order of magnitude as the data, and 3) it requires little additional computational expense. After all nodes have been initialized, the training process begins. The training phase consists of ordering and convergence sub-phases. The ordering phase trains the map's overall topological structure and prepares for fine tuning. The convergence phase employs finer map

updates to statistically fit nodes to the input vector data. A training iteration starts by randomly selecting a data point and computing the Euclidean distance between itself and all nodes. Once all distances have been established, the "winning node" is determined as the node with the smallest distance to the input data vector. A Gaussian-based update is then performed on all the winning node's neighboring nodes. Nodes within the neighborhood are influenced to be more like the input data point by applying a weight vector update as seen in Eqn. (3).

$$w_{j(n+1)} = w_{j(n)} + \eta_{(n)} * h_{j,i(x)}(n) * (x - w_{j(n)}) \quad (3)$$

A neighborhood influence $h_{j,i(x)}$ defined in Eqn. (6), refines the map by influencing each individual node surrounding the winning node to be more like that of the current input vector. For example, if a data point containing bird-like characteristics is assigned to a node, the surrounding nodes' attributes such as "ability to fly" will shift closer to the winning node, whereas "ability to swim" will shift farther away. Initially, the neighborhood width is set to the entire map during the ordering phase. As each iteration completes, the neighborhood width becomes exponentially smaller per Eqn. (4). Another vital factor is the learning rate. The time-varying learning rate $\eta_{(n)}$, shown in Eqn. (5), defines the magnitude of a neighborhood's influence on surrounding nodes during each iteration update. Like the neighborhood width, the learning rate decreases exponentially throughout the ordering phase, but then remains constant throughout the remaining convergence phase.

$$\sigma_{(n)} = \sigma_{(0)} * exp^{-\frac{-n}{\tau_1}} \quad (4)$$

$$\eta_{(n)} = \eta_{(0)} * exp^{-\frac{n}{\tau_2}} \quad (5)$$

$$h_{j,i(x)}(n) = exp^{-\frac{Distance^2}{2*\sigma^2(n)}} \quad (6)$$

Over a specified number of iterations, the decreasing neighborhood influence causes the map to become more refined, resulting in less n-dimensional warping and a 2D visualization of clusters on the nodal map. The CSOM adds an additional step to place contextual labels onto the trained map. Once the SOM has been trained, each data point is projected onto the map a final time. The winning node is found once again, but unlike before, the neighborhood radius is not updated. Instead, the winning node is assigned a label in the form of the design point's objective function value. A set of design variables' order of magnitude can be vastly different from their corresponding objective function value(s) (i.e., there can be more than one objective). Therefore, to develop a better trained map the objective functions are not employed in the training process but instead used as labels. The contextual labels provide a valuable visual by depicting the clusters in accordance to their objective function results instead of numerical weight vectors. Assuming there are more design alternatives than nodes, any given node may contain multiple design alternatives and their respective attributes. To evaluate the significance of a node, its overall performance was calculated as a statistical evaluation of the mean, standard deviation, and minimum value. A new four-map visual was introduced by Richardson et. al [8] that assigns each statistical measure to its own node and creates a fourth map displaying the number of points in a particular node. Three color gradients, interpolated from each statistical evaluation's numerical limits, were assigned to their corresponding map. The CSOM utilized by the researchers used color gradients: blue, green, and red to represent each individual node's respective mean, minimum value, and standard deviation. The color

gradients employed by Richardson et al. across the four-map visual supplied the means to quickly determine defining characteristics such as modality and curvature of the design.

The resulting CSOM reveals the natural groups formed from the input design points but leaves a vast amount of information to comprehend. For example, 1000 design alternatives projected onto a 10x10 map produce 100 nodes for the investigator to analyze. Instead of combing through all 100 nodes, a post-SOM cluster analysis can transform the nodes into larger clusters (i.e., meta-clusters) containing similar attribute relationships. This two-level SOM-based clustering approach is outlined in Figure 1. Once the CSOM has sorted the design alternatives into their respective nodes, the process of forming larger meta-clusters, seen as abstraction level 2 in Figure 1 is set to begin.

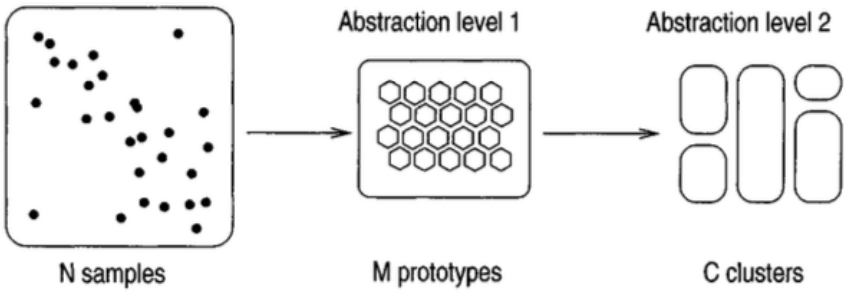

Figure 1. Two-level SOM approach [23]

Post-SOM Clustering

To group the clusters from the output map into larger similar clusters (i.e., "meta-clusters"), each node must first be analyzed with respect to its neighbors. The most similar meta-clusters will then merge with one another throughout an iterative process until two meta-clusters remain or no neighboring clusters remain. Furthermore, each merge is evaluated using a modified cluster validity index to decide the best number of meta-clusters. Since each iteration is

justified by a merge, the optimal number of meta-clusters is determined as the iteration cycle with the most compact and separate meta-clusters. A coloring scheme is then applied to the CSOM to illustrate each respective meta-cluster so the investigator can easily navigate the design space. Design data used in industrial applications tends to present itself in all forms of shapes and size. The accumulated design data rarely represents an easy to cluster hyper-elliptic geometric shape. It would be computationally expensive and difficult to calculate the geometric shape for any given cluster of n-dimensional points, therefore density-based clustering properties were utilized to evaluate a cluster's compactness and separation from others. The formulations Wu and Chow [16] presented to evaluate a cluster's properties were expanded on to better account for the influence a neighboring cluster has upon an individual cluster while taking advantage of the CSOM's resulting topology to reduce computational expense. Each cluster generated by the CSOM contains a set of representative points $V_i = \{v_{i1}, v_{i2}, \dots, v_{ir_i}\}$, where r_i is the number of points within the i^{th}cluster.

The validity index proposed by Wu and Chow [16] is comprised of two types of criterion: cluster compactness and cluster separation. Cluster compactness is referred as the intra-cluster density. The intra-cluster density represents how dense an individual cluster is by computing the percentage of points belonging to its neighborhood of representative points. There have been many modifications to the intra-cluster density formula [10], [16], [22], but the fundamental formulation remains intact. First, the Euclidean distance is computed between all the cluster's representative points. Next, a density value of 1 is assigned to a representative point pairing if the distance is less than or equal to a set neighborhood radius. Once all representative point distances have been evaluated in terms of density, all values are summed to produce the total intra-cluster density. Wu and Chow [16] used the fundamental intra-cluster

density formula and defined a cluster's neighborhood radius as the average standard deviation of all clusters present in the data. However, using the average standard deviation of all clusters fails to account for the sole neighboring j^{th}cluster as the driving impact of the i^{th} cluster's compactness. To account for the neighboring influence missing from the intra-cluster density established by Wu and Chow, the density formulation was changed. The modified density formulation now defines the i^{th} cluster's neighborhood radius as the average standard deviation of it and the neighboring j^{th}cluster. Furthermore, Eqn. (8) sums the density value result from Eqn. (9) for an individual point with respect to the rest of the points within the i^{th} cluster. Then the relative intra-cluster density shown in Eqn. (7) sums the total density of the i^{th}cluster with respect to its j^{th}neighbor and divides by the average standard deviation of the two clusters.

$$Intradensity_{ij} = \frac{1}{stdev_{ij}} \sum_{k=1}^{n_i} Density(v_{ik}) \quad (7)$$

$$Density(v_{ik}) = \sum_{l=1}^{n_i} f(x_l, v_{ik}) \quad (8)$$

$$f(x_l, v_{ik}) = \begin{cases} 1 & \|x_l - v_{ij}\| \leq stdev_{ij} \\ 0 & otherwise \end{cases} \quad (9)$$

In most cases, a cluster containing design points with very similar attributes will exemplify a high intra-cluster density that usually corresponds with a higher separation from neighboring clusters. To determine a cluster's separation, an inter-cluster density must first be established. The inter-cluster density defines the spatial region between neighboring clusters. Wu and Chow [16] used the average standard deviation of neighboring clusters as the bounds of the inter-density region. Consequently, the points of both clusters are combined into one set of

points in which the distance between all points is calculated. This leads to a high computational expense and uses all point distances to influence density instead of only the point pairings between two clusters.

Therefore, a new boundary formula is introduced in this work. The boundary β established by Eqn. (10), must be determined to identify sparse representative points influencing the density of the region between clusters i and j. The i^{th} cluster boundary is subject to the size of its standard deviation versus the difference of length between cluster centers and the j^{th} cluster standard deviation.

$$\beta = \begin{cases} stdev_i & stdev_i < (center\_dist_{ij} - stdev_j) \\ (center\_dist_{ij} - stdev_j) & 0 < (center\_dist_{ij} - stdev_j) < stdev_i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

A representative point is considered an outlying point (z) if the distance between it and the i^{th} cluster mean is greater than the determined boundary. As seen in Figure 2 this operation is carried out for each cluster's representative points yielding a vector of outlying points, which are shown as blue triangles highlighted in yellow. Moreover, as outlying points span further away from the cluster center the least likely they will be the core points within the cluster. Therefore, only the outlying points are considered in the inter-density evaluation of the region between clusters i and j.

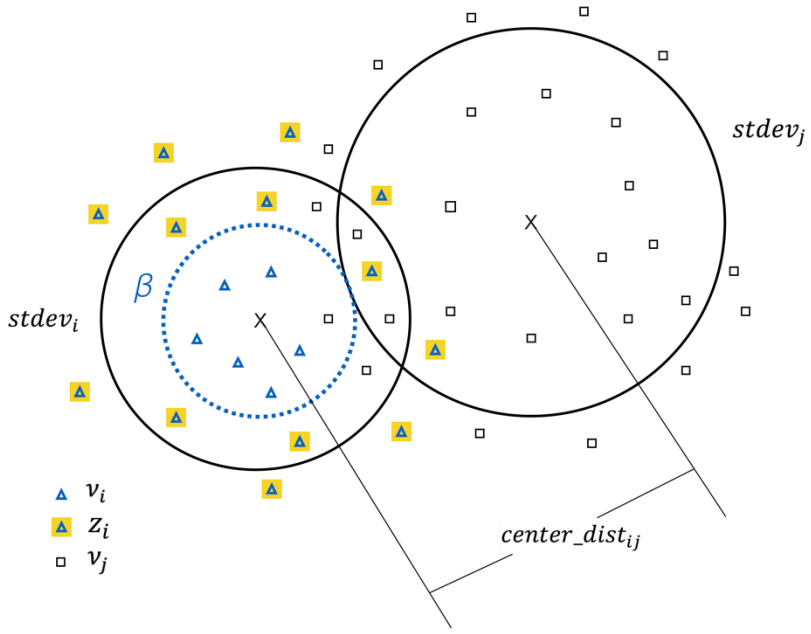

Figure 2. Boundary formulation diagram

Once the outlying points have been established, the density in the region can be computed through Eqn. (11). Only the distance values between neighboring outlying points are used to establish a density value. As Eqn. (12) shows, if the distance between outlying points z_i, z_j is less than half the length between the clusters' centers a density value of 1 is assigned, otherwise a value of 0 is given. All density values based off the outlying points $(\vec{z}_i, \vec{z}_j)$ are summed to produce the region's density.

$$Density(\vec{z}_i, \vec{z}_j) = \sum_{k=1}^{n_i} \sum_{l=1}^{n_j} f\left(z_{ik}, z_{jl}\right) \quad (11)$$

$$f\left(z_i, z_j\right) = \begin{cases} 1 & \|z_i - z_j\| \leq \frac{center\_dist_{ij}}{2} \\ 0 & otherwise \end{cases} \quad (12)$$

The center distance comparison metric is used to filter out outlying points whose attributes are out of the bounds defining the region between the clusters. A cluster may naturally be sparse, yet only a small number of points may be similar to those of a neighboring cluster. To obtain the relative size of the region, the closest pair of i^{th} and j^{th} representations is divided by the length between cluster centers multiplied by the number of outlying pair points. Past derivations only divided the closest pair of i^{th} and j^{th} representations by the standard deviation of the i^{th} and j^{th} cluster. Consequently, clusters' inter-density region may be diluted down as the standard deviation of each cluster grows throughout the clustering process. To better evaluate the region separating neighboring clusters, the inter-density formulation was modified by replacing the standard deviation with the product of half the clusters' center length and the number of possible outlying pairs. This modification prevents sparse clusters containing only a few points in common with their neighbor to outweigh smaller dense clusters overlapping with one another. The inter-cluster density is then obtained in Eqn. (13) by multiplying the region's size by its density found previously in Eqn. (11). The distance between the i^{th} and j^{th} outlying points is the driving factor in determining the separation between neighboring clusters. However, to grasp the full inter-cluster relationship the recently found inter-cluster density is used to accentuate the amount of similarity between the attributes of neighboring clusters. A cluster's separation is then evaluated as the distance of the nearest neighboring pair point over the density of the region separating one another as seen in Eqn. (14).

$$Interdensity_{ij} = \frac{\|close\_rep_i - close\_rep_j\| * Density(\overrightarrow{z_i}, \overrightarrow{z_j})}{center\_dist_{ij} * outlying\_pairs} \quad (13)$$

$$Separation_{ij} = \frac{\|close\_rep_i - close\_rep_j\|}{Interdensity_{ij}} \quad (14)$$

To determine the objective of a compact and separate cluster, the validity index "Composing Density Between and With clusters" (CDbw) [22] is computed. The CDbw validity index serves two purposes. First, it finds the pair of neighboring clusters with the strongest tendency to be clustered. Second, the index can be used as a global function to determine the optimal number of clusters. Equation (15) defines the CDbw as the density of points within a cluster multiplied by the degree of separation from its neighboring cluster.

$$CDbw_{ij} = Intradensity_{ij} * Separation_{ij} \quad (15)$$

The minimum CDbw indicates the clusters were the least compact and separate, therefore the most viable to merge. Furthermore, only neighboring clusters can be merged. For example, neuron A in Figure 3a contains eight direct neighbors. Therefore, the neighbor containing the minimum CDbw validity index out of the eight will merge with neuron A. The number of direct neighbors decreases as the number of neurons within a cluster increase, yet if a pair of clusters are not direct neighbors they cannot be merge. This is shown in Figure 3b and 3c.

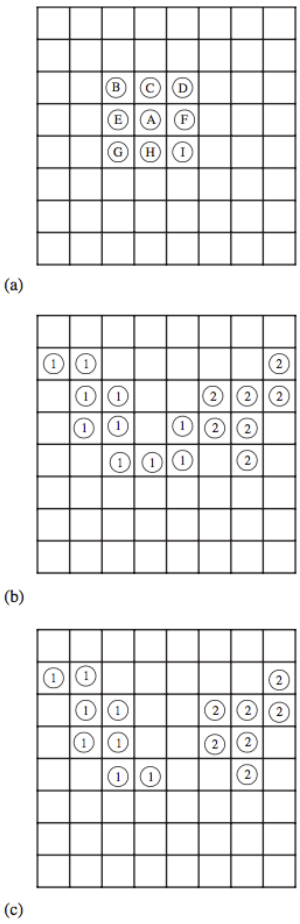

Figure 3. Visualization of cluster neighbors (a) neuron A contains eight direct neighbors: B-I; (b) clusters 1 and 2 can be merged into one cluster since the two clusters are direct neighbors; (c) clusters 1 and 2 are not direct neighbors therefore cannot merge *[16]*

Computing the CDbw validity index is an iterative process. Numerous two-level SOM methods compute the CDbw value between all clusters defined at the start by each node on the SOM output map [16], [22], [23]. Determining a validity index between all clusters is computationally expensive and unnecessary. Since the CSOM has maintained the natural topology of input design variables, only neighboring clusters need to be calculated. To begin, the CSOM separated the original design variables into a set of clusters defined as each node in the original map. For example, if there is a 10x10 nodal map there will initially be 100 clusters.

Out of the 100 clusters, the cluster pair with the minimum CDbw validity index is forced to merge since it is the least compact and separate. This process will then repeat until only two clusters remain. However, the optimal number of clusters is usually more than two. To determine the optimal number of clusters, *c,* a global CDbw validity index is constructed using Eqn. (16). The average intra-cluster density and separation for a cluster's neighbors is summed to compute a global validity index.

$$CDbw_{global}(c) = \frac{1}{c^2}\sum_{i=1}^{c} Intradensity_{i_{avg}} * Separation_{i_{avg}} \quad (16)$$

After the global validity index is found for each corresponding number of clusters the optimal number of meta-clusters can be found. Previously, the minimum CDbw was the defining metric to note a viable merge. The opposite logic is applied to obtain optimal meta-clusters. The number of clusters with the largest global CDbw value are considered the most compact and separate, leading to assumption they contain the highest abundance of similar attributes. Using this modified clustering algorithm, design alternatives can be classified into manageable sections to prevent designers from overlooking influential designs amongst countless alternatives accumulated in the dataset.

Clustering Validation

Two problems are used to test the viability and accuracy of the proposed meta-clustering approach. One is a well-known machine learning problem and the other is an established engineering design benchmark problem. The wine data set [24] has been commonly used to test the performance of various machine learning classification algorithms. The data set is a chemical analysis of wines from the same region of Italy, yet produced by three different

vineyards. There is a grand total of 178 sample wines with 59 in “Class 1”, 71 in “Class 2” and 48 in “Class 3” (class numbers denote the vineyard). Each sample wine contains 13 attributes specifying its defining characteristics. Furthermore, a wine sample’s flavonoid, proline, and color intensity are the most influential variables when it comes to separating the three wine classes [25]. Each sample wine was accompanied by an objective label denoting the class of wine in which it belongs. A 4x4 output map was chosen to ensure the nodes’ weight vector properly assimilated the input wine samples. The test data set was used to validate the performance of the modified clustering algorithm.

The engineering design problem is a case study of a geo-stationary communication satellite tasked with broadcasting television signals. Essentially, the satellite is a relay receiving transmitted signals from a ground station, amplifying and processing the signal, and sending the signal to another ground station. The satellite system is a conceptual design typically consisting of hundreds of subsystems that has been reduced to seven broad subsystems: payload, propulsion, power, ADCS, thermal, structures, and launch vehicle. Nine design variables representing transmission frequencies and powers, antenna diameters and energy densities influence the subsystems directly and indirectly. The Design Structure Matrix (DSM) of the satellite shown in Figure 4 represent the subsystem couplings, which are displayed at dots connecting lines between subsystems. A coupling example is the effect the propulsion and structures subsystems have on one another. The mass of the structures subsystem is an input to the propulsion system while the propulsion subsystem mass and volume are an input of the structures system. All nine design variables are shown on the top line encapsulated in a series of brackets. Detailed descriptions of these variables can be found in Bloebaum et. al. [26].

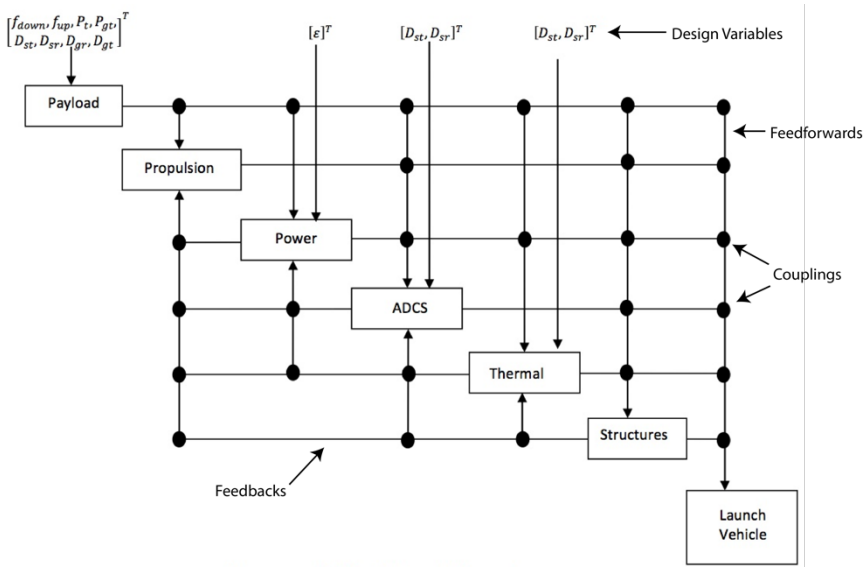

Figure 4. DSM of satellite system

Bloebaum's previous work [26] implemented an objective function to capture system characteristics (mass) that is related to an economic system characteristic (cost). While minimizing the mass of the satellite will reduce its overall cost, only a sparse amount of optimal designs will remain feasible. Minimizing the system's cost with a cost model will generally favor a lesser mass but will narrow the design space towards the cheapest set of designs within the respected design constraints in Figure 5.

$$
\begin{gathered}
g_1: 10dB - SNR_{composite} \leq 0 \\
1\ GHz \leq f_{down} \leq 100\ GHz \\
1\ GHz \leq f_{up} \leq 100\ GHz \\
5\ W \leq P_t \leq 300\ W \\
300\ W \leq P_{gt} \leq 30000\ W \\
0.5m \leq D_{sat,trans} \leq 2.5m \\
0.5m \leq D_{sat,rec} \leq 2.5m \\
2\ m \leq D_{ground.rec} \leq 20m \\
2\ m \leq D_{ground,trans} \leq 20\ m \\
35\frac{W-hr}{kg} \leq \varepsilon \leq 200\frac{W-hr}{kg}
\end{gathered}
$$

Figure 5. Satellite design constraints

However, minimizing the mass and cost do not account for the stakeholder's true preference, which in this example is to maximize the Net Present Profit (NPV). This presents an opportunity to apply the modified clustering algorithm to better understand how the NPV in the design space is influenced by minimizing the systems overall cost versus mass. The researchers sampled a total of 1000 design alternatives, 500 mass and 500 cost design alternatives, using the alternatives' nine design variables as the input and their corresponding NPV as its output label.

Results

The following sections describe the results obtained using the proposed method on the wine data set and the satellite design problem.

Wine Data Set

For this test dataset, the researchers first tested the modified validity index, described in the methodology, using the wine data set. The resulting meta-clusters map has been combined with mean, minimum, and standard deviation statistical maps from the original CSOM implementation. The three statistical maps use the wine classification label as the objective function criteria to compute their respective values. The three distinct shades of blue in the top right map of Figure 6 denote the mean classification value for each node. The darkest shade resembles "Class 3", lightest shade "Class 1", and average shade of the two "Class 2". At first glance, the CSOM looks to have perfectly separated all wine classes from one another. The darker shades of red in the standard deviation map, bottom-right, reveal three nodes with a deviation greater than zero. This shows there are sample points from more than one class in

the three nodes. Since the clustering algorithm is built upon the trained CSOM, a node containing sample points from different classifications cannot be identified by their individual class. Instead, the entire set of points within a node will be identified as one classification. Therefore, it can be understood it is not possible for the post-SOM clustering process to obtain 100 percent accuracy. Once the clustering process was applied to the CSOM a total of four wines were misclassified out of 178 samples, yielding an accuracy of 97.76%. In total, meta-cluster 1 was the most inaccurate with nearly all the misclassifications; two ideally belonging to orange meta-cluster 2 and one to purple meta-cluster 3.

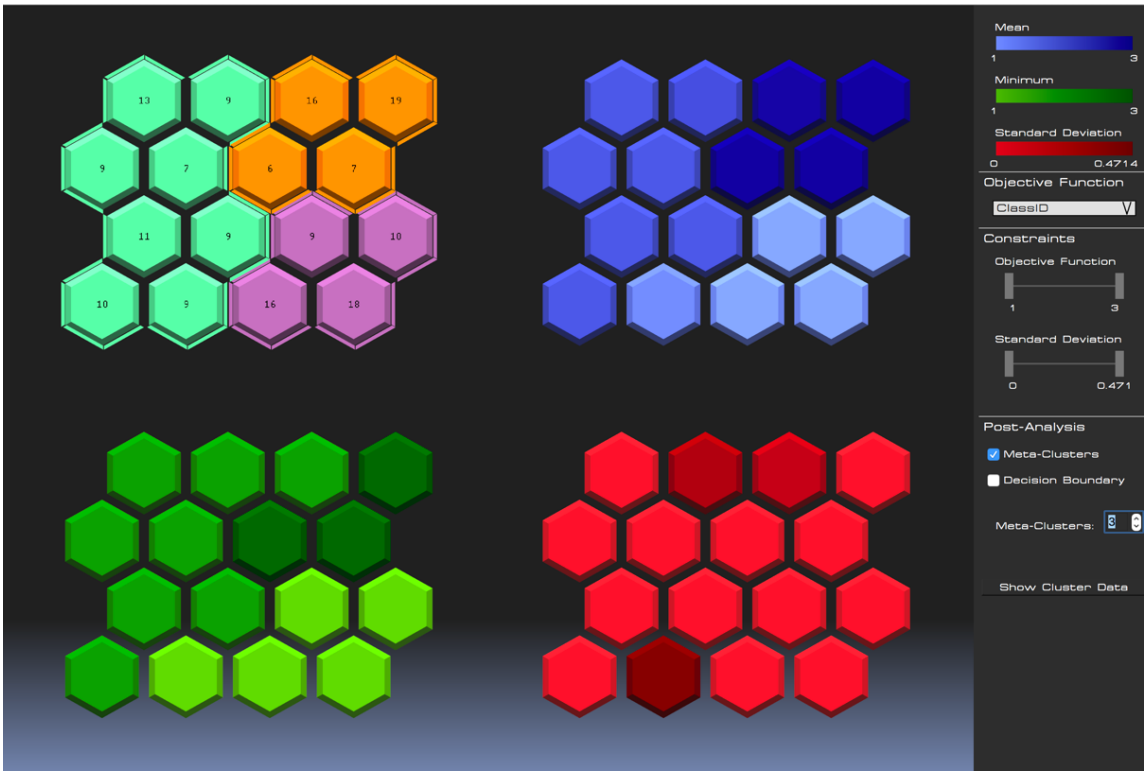

Figure 6. Resulting meta-clusters obtained with the wine data set

The global CDbw value was tracked throughout the iteration process and Figure 7 details the exact global CDbw value per number of meta-clusters. As shown, the CDbw slope is constantly changing between each iteration and does not resemble a smooth curve. The peaks

and valleys are produced from the meta-clusters' overall compactness battling the amount of separation between one another, but the optimal number of clusters in the figure can be clearly seen with the highest CDbw value of approximately 20.

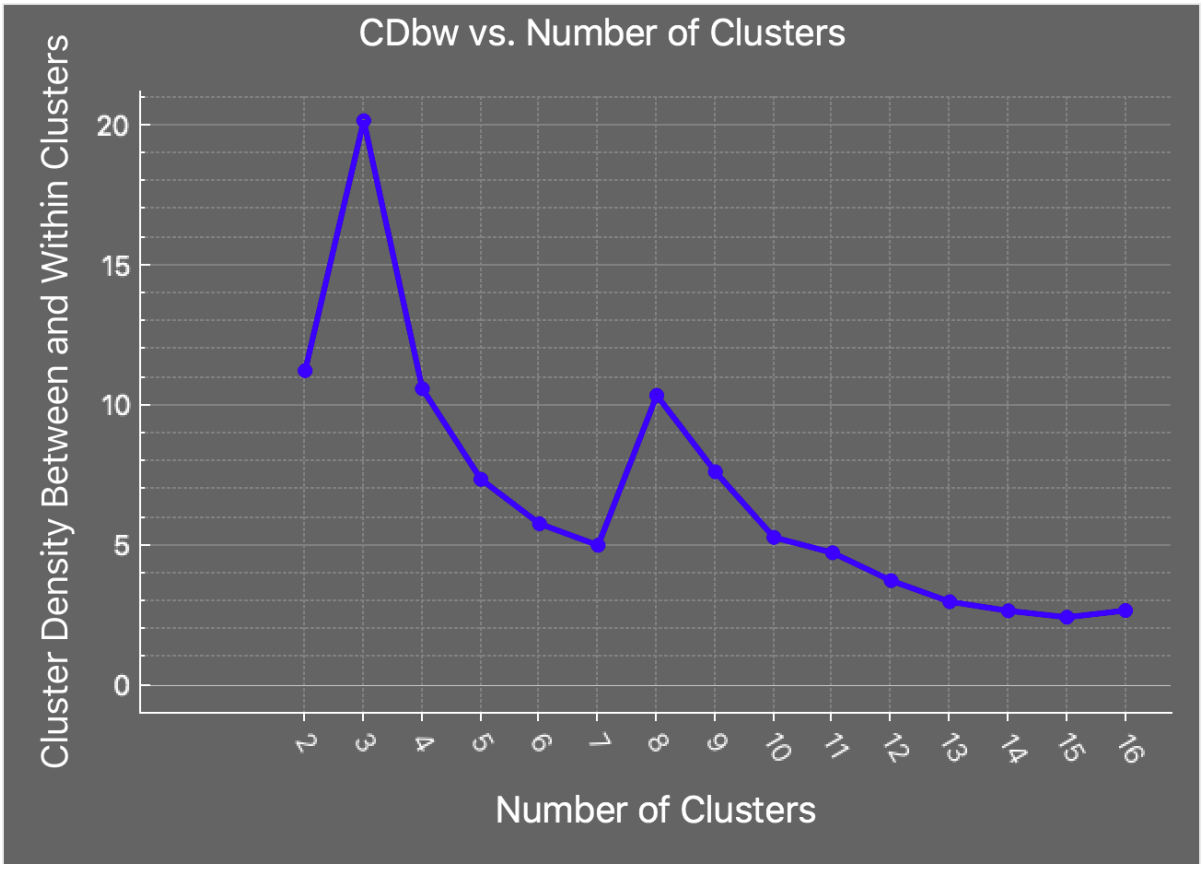

Figure 7. Wine data set global CDbw validity index vs. number of clusters

Although the meta-clusters were not 100 percent accurate, the wine data set validated the performance of researchers modified clustering algorithm.

Communication Satellite

For this test problem, a total of 1000 satellite system design alternatives consisting of nine variables and NPV output were sampled from two underlying objective functions. The first 500 alternatives generated a NPV output with respect to minimizing system mass, while the

second set of 500 alternatives were generated with respect to minimizing cost. These design alternatives were normalized to prevent variables with extremely high magnitudes outweighing those with low magnitudes. For example, the magnitude of the uplink frequency ranged anywhere between 10^8- 10^{11} while the receiving antenna was within 0.5 and 2.5.

After the CSOM was complete, the density-based clustering algorithm iterated through the potential number of meta-clusters to find three optimal compact and separate clusters as shown the upper left map in Figure 8. The meta-clusters show three narrowed regions in the design space allowing the designer to hone in on a select region to lessen time spent analyzing design variable interactions and impact on the system's NPV. At first glance, the nodes contained within the orange meta-cluster, in the figure, appear to have darker shades of blue and green in the accompanying maps for minimum and mean values of clustered data (i.e., upper right and lower left of the figure). Therefore, there are likely more design alternatives containing a higher NPV. The number of different shades of blue indicates the NPV output has a wide range of values. Even though the design points are very similar to one another in each node, their small differences can drastically affect the overall return on investment. Hence, the NPV is sensitive to design differences and not only one design set produces the largest NPV. Instead, there can be many designs that all produce a high return on investment.

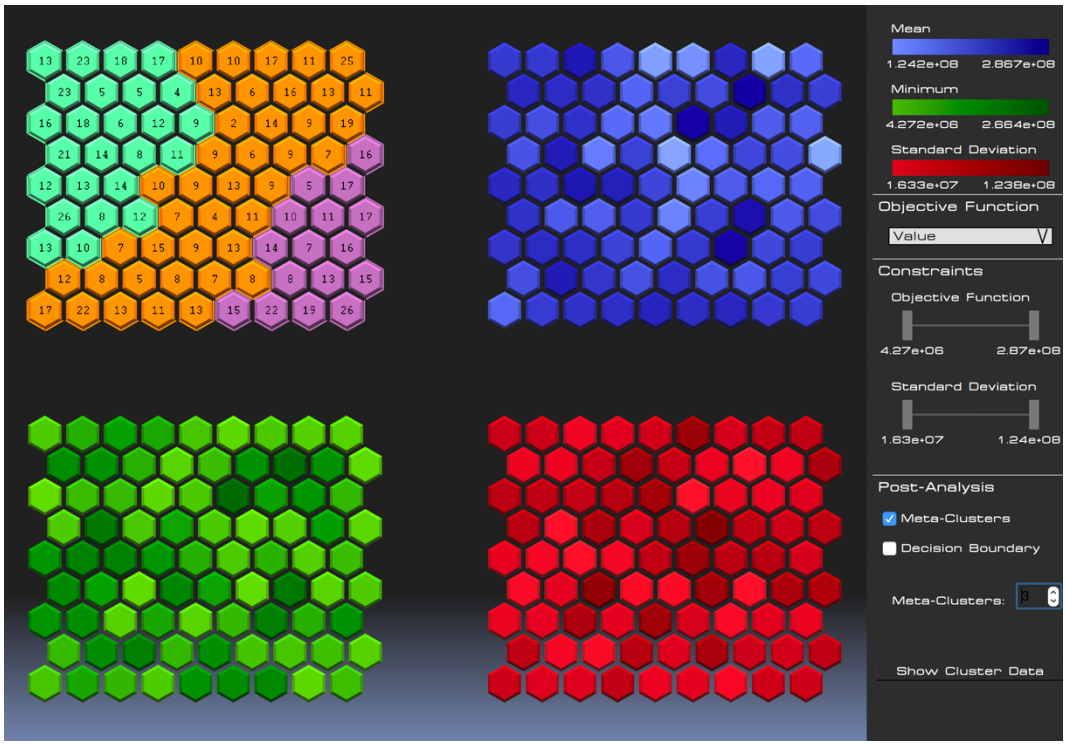

Figure 8. Communication satellite optimal meta-clusters

To better understand the designs within each meta-cluster, the associated design variables need to be examined. Each design variable has been statistically analyzed in Table 1 and shows a few distinct differences between the meta-clusters. For example, the ground receiver and transmitter diameters are similar amongst all meta-clusters, indicating their sizes are not extremely sensitive to the downlink frequency. Therefore, it can be inferred the attributes associated with the satellite system's functionality on earth do not impose a grave impact on the design of the satellite bus that is launched in to orbit. Additionally, the uplink and downlink frequencies have different distributions yet a large difference in their order of magnitude, revealing the uplink frequency may have a large impact on the difference of design cost rather than the downlink frequency.

TABLE 1. META-CLUSTER 1, 2, AND 3 DESIGN VARIABLE STATISTICS

	Mean			Standard Deviation			Minimum			Maximum		
Variable	1	2	3	1	2	3	1	2	3	1	2	3
$D_{ground.rec}$	7.7	8.4	7.4	4.3	4.3	3.5	2.0	2.0	2.0	19.7	19.8	19.2
$D_{ground,trans}$	13.5	11.3	12.9	3.8	4.3	4.2	2.0	2.0	2.0	20.0	20.0	20.0
$D_{sat,rec}$	2.1	1.0	1.0	0.3	0.4	0.4	0.5	0.5	0.5	2.5	2.5	2.1
$D_{sat,trans}$	1.4	1.5	0.9	0.6	0.6	0.3	0.5	0.5	0.5	2.5	2.5	2.3
P_{gt}	5873.6	4354.1	5928.0	2490.1	2761.0	2116.8	300.0	300.0	300.0	10000.0	9768.1	10000.0
P_t	7.8	10.5	15.1	7.7	12.2	18.2	5.0	5.0	5.0	61.0	83.0	95.7
f_{down}	40.0	41.8	43.1	22.3	23.1	22.7	1.0	1.0	1.0	99.0	100.0	99.0
f_{up}	44.8	53.8	35.2	28.6	31.7	25.8	1.0	1.0	1.0	100.0	100.0	100.0
ε	107.0	101.9	124.7	30.6	33.8	36.2	35.7	35.7	35.7	194.5	200.0	199.9

For example, the diameter of satellite transmitter and receiver accompanied by the power consumption drive different sets of designs. Meta-cluster 3, highlighted in purple, contains the smallest transmitter and receiver diameters. Furthermore, the meta-cluster tends to exploit a set of designs focused on minimizing weight to reduce propulsion requirements needed for the launch vehicle to successfully transport the satellite into orbit. Since the satellite is a heavily coupled system, one variable may not create the defining difference, yet an evaluation of multiple design variables may illuminate a pattern. To gain insight into the design variables influencing the divide in design space illustrated by the meta-clusters in Figure 8, a series of histogram plots shown, in Figure 9a-c, were created to visualize the distribution difference amongst the most influential design variables in each design region.

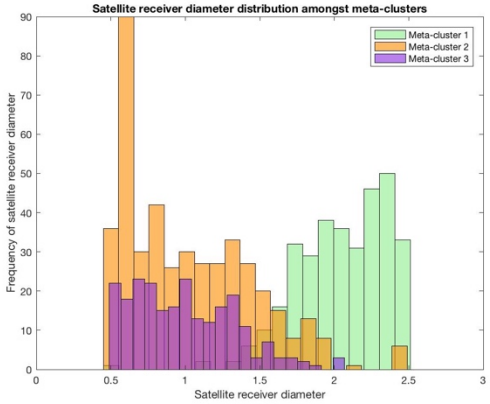

Figure 9a. Distribution of satellite receiver diameter amongst meta-clusters

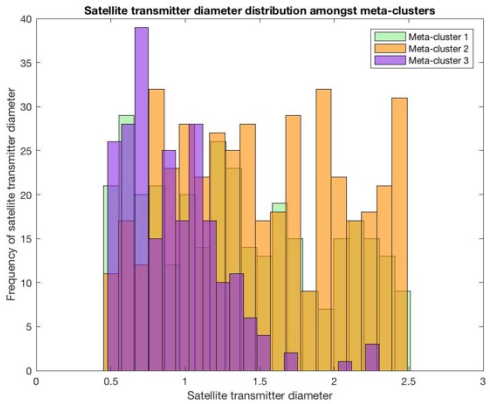

Figure 9b. Distribution of satellite transmitter diameter amongst meta-clusters

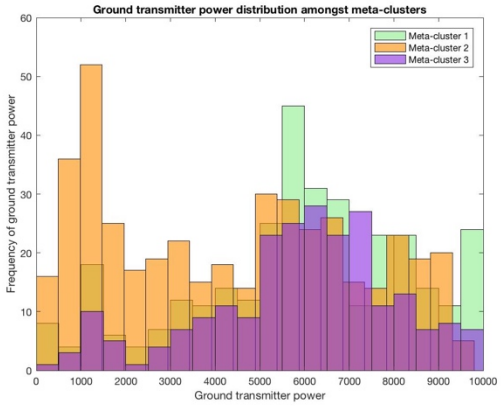

Figure 9c. Distribution of ground transmitter power amongst meta-clusters

A general design description can be developed for each meta-cluster using the statistics in Table 1 and histograms shown in Figure 9a-c. As can be seen in the figure, there are many regions of overlap between the clusters. This is due to the high number of designs that, while not optimal, do contain feasible and even somewhat desirable characteristics. However, even with this overlap, a designer would still want to closely examine the inherent variable relationships for deep understanding prior to making any decisions. The first design region, meta-cluster 1, is geared toward cost minimization by placing an emphasis on designs with a large ground infrastructure to reduce the work load of satellite. Bloebaum et. al. found a high uplink frequency decreases the amount the cost of the ground stations on earth, lessening the

satellite's power consumption needed to amplify the broadcasting signal. Most different from meta-cluster 1 is meta-cluster 3's design region. This region focuses on reducing mass by employing design alternatives with small satellite transmitter and receiver diameters. Even though the ground transmitter power is relatively high, the satellite mass is not effected. The final design region is meta-cluster two, which has a blend of qualities extended from the overlapping regions of meta-cluster 1 and 3. This region is most pertinent because it contains design alternatives with the low mass influence of meta-cluster 3 while utilizing high uplink and downlink frequencies to improve the system's signal to noise ratio. After the defining traits of each region have been revealed, the designer can focus on the design alternatives with the highest NPV. The researchers used the brushing tool built by Richardson et. al. to highlight and isolate the nodes with the largest mean NPV. Two nodes containing a total of eighteen design alternatives and an average NPV of $286.81 million are shown in Figure 10 in the mean contextual map in the upper right.

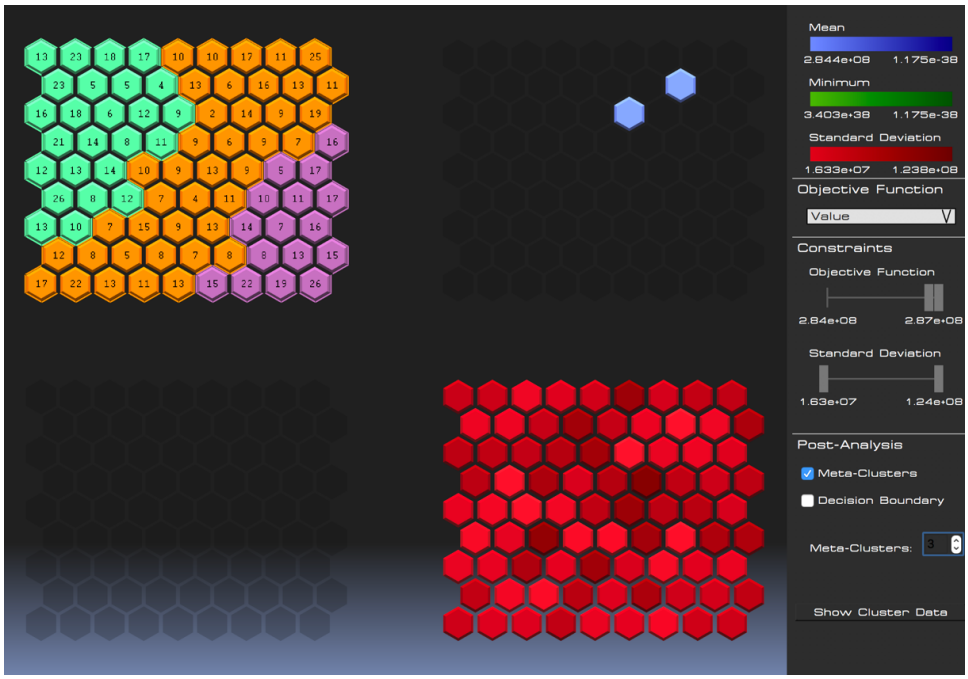

Figure 10. Isolation of design alternatives with the largest NPV

The overarching goal for the designer is to understand how the design variables produced such a large NPV. From a visual standpoint, the two nodes lie on the boarder of meta-cluster 3, therefore the designs within the nodes should contain some variable properties like those in meta-cluster 3.

Table 2 shows the two nodes contain small satellite receiver and transmitter diameters, as do the design alternatives within meta-cluster 3. Unlike the meta-cluster 3 region, the design alternatives contain very large uplink and downlink frequencies which lower cost and generate more revenue by combining an improved signal to noise ratio and reduced power consumption.

TABLE 2. DESIGN VARIABLES STATISTICS FOR NODES WITH LARGEST NPV

Variable	Mean	Standard Deviation	Minimum	Maximum
$D_{ground.rec}\ (m)$	5.8	1.9	3.8	11.0
$D_{ground,trans}\ (m)$	14.1	2.6	9.8	19.5
$D_{sat,rec}\ (m)$	0.9	0.2	0.6	1.2
$D_{sat,trans}\ (m)$	0.9	0.3	0.5	1.5
$P_{gt}\ (W)$	4740.1	1118. 2	2646.8	6294.8
$P_t\ (W)$	6.5	6.0	5.0	28.9
$f_{down}\ (GHz)$	74.7	15.3	42.0	97.0
$f_{up}\ (GHz)$	86.0	10.0	71.0	100.0
$\varepsilon\left(\frac{W-hr}{kg}\right)$	104.5	5.5	91.7	113.3

Ultimately, the large design space was narrowed to three meta-cluster regions. The first with an average NPV of $230.74 million and designs optimized by the overall cost. The second region, containing designs with the highest NPV and $216.83 million average embodied a small mass and reduced cost design. Its relatively low average NPV was due to the fact it contained dramatic outliers from the overlapping designs of meta-cluster 1 and 3. The final region averaged $223.85 million with a focus on reducing the systems mass. The modified clustering algorithm was able to correctly classify the design alternatives according to their inherent characteristics, while locating designs with high NPV in each meta-cluster. Lastly, the

risks associated with each design are dynamically changing as time goes on. For example, if the rocket fuel needed to propel the launch vehicle in to orbit drastically increases the designer would then focus on the third meta-cluster design space to locate a set of design constraints focused on minimizing the mass of the satellite system. The benefit of narrowing the design space into meta-cluster regions is that the design space will stay relatively the same even though the system's NPV may change with fluctuating market evaluations.

Conclusions

In conclusion, a two-level CSOM-based clustering approach was used to transform raw data into large meta-clusters. Using the modified clustering algorithm to segment the design space provided insight to how a large-complex system's plethora of design alternatives can be generalized into a few overarching designs with similar principal characteristics without having to spend a copious amount of time analyzing each design alternative. The wine data set was used to evaluate the validity of the modified clustering algorithm which achieved an accuracy of 97.76 percent. Furthermore, a geo-stationary communication satellite example was employed which narrowed 1000 design alternatives into three meta-clusters of generalized designs. In the future, the validity index needs to be further analyzed to determine whether unique local clusters should be forced to merge or if the other least compact and separate cluster pairings should merge during the clustering process. The researchers are also interested in the development of a convergence criterion to determine optimal clusters during the clustering process to reduce computational expense and time by halting all clustering.

References

[1] J. Eddy and K. E. Lewis, "Multidimensional Design Visualization in Multiobjective Optimization," *9th AIAA/ISSMO Symp. Exihibit Multidiscip. Anal. Optim.*, no. September, pp. 1–11, 2002.

[2] E. H. Winer and C. L. Bloebaum, "Visual design steering for optimization solution improvement," *Struct. Multidiscip. Optim.*, vol. 22, no. 3, pp. 219–229, 2001.

[3] T. Stump, G., Yukish, M., Martin, J. D. J., and Simpson, "The ARL Trade Space Visualizer- An engineering decision-making tool," *10 th AIAA/ISSMO Multidiscip. Anal. Optim. Conf. Am. Inst. Aeronaut. Astronaut.*, p. 17, 2004.

[4] P.-W. Chiu, A. M. Naim, K. E. Lewis, and C. L. Bloebaum, "The hyper-radial visualisation method for multi-attribute decision-making under uncertainty," *Int. J. Prod. Dev.*, vol. 9, no. 123, pp. 4–31, 2009.

[5] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982.

[6] M. Franzmeier, U. Witkowski, R. Ulrich, F. M., W. U., and R. U., "Explorative data analysis based on self-organizing maps and automatic map analysis," *Comput. Intell. Bioinspired Syst. 8th Int. Work Conf. Artif. Neural Networks, {IWANN} 2005. Proc. Lect. Notes Comput. Sci.*, pp. 725–733, 2005.

[7] G. Polzlbauer, "Advanced data exploration methods based on Self-Organizing Maps."

[8] T. Richardson and E. Winer, "VISUALLY EXPLORING A DESIGN SPACE THROUGH THE USE OF MULTIPLE CONTEXTUAL SELF-ORGANIZING MAPS," *ASME 2011 Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, pp. 1–10, 2011.

[9] N. R. Pal, J. C. Bezdek, and E. C. Tsao, "Generalized Clustering Networks and Kohonen ' s Self-organizing Scheme," vol. 4, no. 4, 1993.

[10] M. Han, Jiawei, Pei, Jian, and Kamber, "Data Mining: Concepts and Techniques," *Saint Louis Elsevier Sci.*, 2014.

[11] J. Mao and A. K. Jain, "A Self-Organizing Network for Hyperellipsoidal Clustering (HEC)," *Neural Networks*, vol. I, no. 1, pp. 16–29, 1996.

[12] E. O. J. Lampinen, "Clustering properties of hierarchical self-organizing maps," *J. Math. Imag. Vis. 2*, pp. 261–272, 1992.

[13] F. Murtagh, "Interpreting the Kohonen self-organizing feature map using contiguity-constrained clustering," *Pattern Recognit. Lett. 16*, vol. 399–408, 1995.

[14] M. Y. Kiang, "Extending the Kohonen self-organizing map networks for clustering analysis," *Comput. Stat. Data Anal. 38*, vol. 161–180, 2001.

[15] E. A. J. Vesanto, "Clustering of the self-organizing map," *IEEE Trans. Neural Networks 11*, vol. 586–600, 2000.

[16] S. Wu and T. W. S. Chow, "Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density," *Pattern Recognit.*, vol. 37, no. 2, pp. 175–188, 2004.

[17] D. F. Swayne, D. Cook, and A. Buja, "XGobi: Interactive Dynamic Data Visualization in the X Window System," *J. Comput. Graph. Stat.*, vol. 7, no. 1, pp. 113–130, 1998.

[18] G. Stump, T. Simpson, M. Yukish, and L. Bennett, "Multidimensional Visualization and its Application to a Design by Shopping Paradigm," *9th AIAA/ISSMO Symp. Exihibit Multidiscip. Anal. Optim.*, no. 814, pp. 4–6, 2002.

[19] D. Kanukolanu and E. H. Winer, "A Multidimensional Visualization Interface to Aid in Trade-off Decisions During the Solution of Coupled Subsystems Under Uncertainty," *J. Comput. Inf. Sci. Eng.*, vol. 6, no. November, pp. 288–299, 2006.

[20] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 2, no. 1, pp. 86–97, 2012.

[21] J. Lampinen and E. Oja, "Clustering Properties of Hierarchical Self-Organizing Maps 1 Introduction," vol. 2, pp. 261–272, 1992.

[22] M. Halkidi and M. Vazirgiannis, "Clustering validity assessment using multi representatives," *Proc. SETN Conf. ...*, no. April, pp. 237–248, 2002.

[23] S.-L. Shieh and I.-E. Liao, "A new approach for data clustering and visualization using self-organizing maps," *Expert Syst. Appl.*, vol. 39, no. 15, pp. 11924–11933, 2012.

[24] C. L. Blake and C. J. Merz, "UCI repository of machine learning databasese," *Department of Information and Computer Science, University of California at Irvine, CA,* 1998. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository. html.

[25] Blias. K. Discovery, "Wine Recognition Data Set." [Online]. Available: http://www.bliasoft.com/data-mining-software-wine.html.

[26] T. Richardson, H. Kannan, C. L. Bloebaum, and E. H. Winer, "Incorporating Value-Driven Design into the Visualization of Design Spaces Using Contextual Self-Organizing Maps: A Case Study of Satellite Design," *15th AIAA/ISSMO Multidiscip. Anal. Optim. Conf.*, no. June, pp. 1–10, 2014.

CHAPTER 4: CONCLUSIONS AND FUTURE WORK

Conclusions

The abundance of large and complex engineered systems has exponentially increased due to demands for high levels of performance while operating on reduced budget costs. Understanding these systems often requires visualizing complex multi-dimensional design spaces. For a designer to decide upon an appropriate optimization algorithm, a fundamental understanding of the design space must be established to determine a starting point. Visualization methods, such as self-organizing maps (SOM), offer powerful methods to visualize n-dimensional data with the goal to produce simple to understand representations that quickly highlight trends to support decision-making. A drawback to using SOMs is the clustering of promising points with predominately less desirable data.

Instead of combing through all possible trained nodes, a post-SOM cluster analysis can transform the nodes into larger clusters (i.e., meta-clusters) containing similar attribute relationships. The work of this thesis proposed a modified cluster validity index to provide a better means of handling data associated with large-complex systems while obtaining an optimal number of meta-clusters segmenting the design space. The modified SOM-based clustering algorithm reveals insight to how a large-complex system's plethora of design alternatives can be generalized into a few overarching designs with similar principal characteristics without having to spend a copious amount of time analyzing each design alternative. Furthermore, the proposed clustering algorithm automatically segments the design space into manageable sections to help stop designers from overlooking influential designs amongst the countless alternatives in the design space.

Future Work

Future project work will focus on addressing the large amount of computational expense expended while clustering the self-organizing map. The use of a parallel algorithm will need to be investigated to reduce the computational time spent clustering design points associated with datasets containing a great number sample points and dimensions. Interest also lies in the development of a convergence criterion to determine the optimal meta-clusters during the clustering process to reduce computational time by halting all clustering.

Computational expense is not the only future concern. Data commonly found in industrial applications has a wide range of topology, which can heavily influence the clustering process. A set of preference criterions associated with each variable can provide an engineer with options to optimize clustering by introducing a designer's prior knowledge into the meta-cluster formation. Additionally, feature extraction algorithms could yield the potential of identifying underlying relevant data pertinent to the meta-cluster formation. Finally, the clustering validity index needs to be further analyzed to determine whether unique local clusters should be forced to merge or if other least compact and separate cluster pairings should merge during the clustering process.

REFERENCES

[1] U. G. Pulse, "Big data for development: Challenges & opportunities," *Nac. Unidas, Nueva York, mayo*, no. May, 2012.

[2] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, 2015.

[3] "MathWorks - Contour plot under a 3-D shaded surface plot." [Online]. Available: https://www.mathworks.com/help/matlab/ref/surfc.html.

[4] "Pareto Image." [Online]. Available: http://www.ozeninc.com/images/Pareto4.gif.

[5] J. Eddy and K. E. Lewis, "Multidimensional Design Visualization in Multiobjective Optimization," *9th AIAA/ISSMO Symp. Exihibit Multidiscip. Anal. Optim.*, no. September, pp. 1–11, 2002.

[6] E. H. Winer and C. L. Bloebaum, "Visual design steering for optimization solution improvement," *Struct. Multidiscip. Optim.*, vol. 22, no. 3, pp. 219–229, 2001.

[7] P.-W. Chiu, A. M. Naim, K. E. Lewis, and C. L. Bloebaum, "The hyper-radial visualisation method for multi-attribute decision-making under uncertainty," *Int. J. Prod. Dev.*, vol. 9, no. 123, pp. 4–31, 2009.

[8] D. F. Swayne, D. Cook, and A. Buja, "XGobi: Interactive Dynamic Data Visualization in the X Window System," *J. Comput. Graph. Stat.*, vol. 7, no. 1, pp. 113–130, 1998.

[9] G. Stump, T. Simpson, M. Yukish, and L. Bennett, "Multidimensional Visualization and its Application to a Design by Shopping Paradigm," *9th AIAA/ISSMO Symp. Exihibit Multidiscip. Anal. Optim.*, no. 814, pp. 4–6, 2002.

[10] T. Stump, G., Yukish, M., Martin, J. D. J., and Simpson, "The ARL Trade Space Visualizer- An engineering decision-making tool," *10 th AIAA/ISSMO Multidiscip. Anal. Optim. Conf. Am. Inst. Aeronaut. Astronaut.*, p. 17, 2004.

[11] M. Yukish, G. M. Stump, and S. Lego, "Visual steering and trade space exploration," *IEEE Aerosp. Conf. Proc.*, pp. 1–9, 2007.

[12] D. Kanukolanu and E. H. Winer, "A Multidimensional Visualization Interface to Aid in Trade-off Decisions During the Solution of Coupled Subsystems Under Uncertainty," *J. Comput. Inf. Sci. Eng.*, vol. 6, no. November, pp. 288–299, 2006.

[13] A. P. Gurnani, T.-K. See, and K. Lewis, "An Approach to Robust Multiattribute Concept Selection," *Vol. 2 29th Des. Autom. Conf. Parts A B*, vol. 2003, no. July, pp. 35–44, 2003.

[14] D. Nagrath, W. W. Bequette, S. M. Cramer, and A. Messac, “Multiobjective optimization strategies for linear gradient chromatography,” *AIChE J.*, vol. 51, no. 2, pp. 511–525, 2005.

[15] C. A. Mattson and A. Messac, “Pareto Frontier Based Concept Uncertainty , with Visualization Selection under Corresponding Author,” *Optim. Eng.*, vol. 6, no. 1, pp. 85–115, 2005.

[16] “Techplot 360, Software Package, Ver. R1, Tecplot, Inc. 2012.”

[17] “Techplot Focus 2012, Software Package, Ver. R1, 2012, Techplot, Inc. 2012.”

[18] “Tecplot Rs 2012, Software Package, Ver. R4, Techplot, Inc., 2012.”

[19] “PHX ModelCenter, Software Package, Ver. 10.0, Phoenix Integration, Inc., 2012.”

[20] “Isight, Software Package, Ver. 5.7, Dassalt Systems, 2012.”

[21] “Tableau Desktop, Software Package, Ver. 7.0, Tableau Software, 2013.”

[22] “Spotfire, Software Package, Ver. 5.5, TIBCO, 2013.”

[23] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982.

[24] M. Fleischmann and W. Strauss, “Digital Sparks - Semantic Map,” 2008. .

[25] “SOM Training Image.” [Online]. Available: https://origin-ars.els-cdn.com/content/image/1-s2.0-S0925753511000865-gr2.jpg.

[26] J. Hollmen, “U-Matrix,” 1996. [Online]. Available: https://origin-ars.els-cdn.com/content/image/1-s2.0-S0925753511000865-gr2.jpg.

[27] G. Polzlbauer, “Advanced data exploration methods based on Self-Organizing Maps.”

[28] S. Haykin, “Neural Networks A Comprehensive Foundation,” *Prentice Hall Publ.*, 1999.

[29] T. Richardson and E. Winer, “VISUALLY EXPLORING A DESIGN SPACE THROUGH THE USE OF MULTIPLE CONTEXTUAL SELF-ORGANIZING MAPS,” *ASME 2011 Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, pp. 1–10, 2011.

[30] Vesanto, “Som-based data visualization methods,” *Intell. Data Anal.*, vol. 3, pp. 111–126, 1999.

[31] E. Kasner, C. A. Hunter, D. Ph, K. Kariko, and D. Ph, “Unsupervised Spatiotemporal Analysis of FMRI Data Using Graph-Based Visualizations of Self-Organizing Maps,” vol. 70, no. 4, pp. 646–656, 2013.

[32] X.-Z. Chu, L. Gao, H.-B. Qiu, W.-D. Li, and X.-Y. Shao, "An expert system using rough sets theory and self-organizing maps to design space exploration of complex products," *Expert Syst. Appl.*, vol. 37, no. 11, pp. 7364–7372, 2010.

[33] G. Andrienko *et al.*, "Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns," *Comput. Graph. Forum*, vol. 29, no. 3, pp. 913–922, 2010.

[34] N. Andrienko and G. Andrienko, "A visual analytics framework for spatio-temporal analysis and modelling," *Data Min. Knowl. Discov.*, vol. 27, no. 1, pp. 55–83, 2013.

[35] M. Su, Y. Zhao, and J. Lee, "Som-based optimization," 2004.

[36] S. Obayashi and D. Sasaki, "Visualization and Data Mining of Pareto Solutions Using Self-Organizing Maps," *Evol. Multi-criterion Optim.*, vol. 2632, pp. 796–809, 2003.

[37] N. R. Pal, J. C. Bezdek, and E. C. Tsao, "Generalized Clustering Networks and Kohonen ' s Self-organizing Scheme," vol. 4, no. 4, 1993.

[38] M. Han, Jiawei, Pei, Jian, and Kamber, "Data Mining: Concepts and Techniques," *Saint Louis Elsevier Sci.*, 2014.

[39] J. Mao and A. K. Jain, "A Self-Organizing Network for Hyperellipsoidal Clustering (HEC)," *Neural Networks*, vol. I, no. 1, pp. 16–29, 1996.

[40] E. O. J. Lampinen, "Clustering properties of hierarchical self-organizing maps," *J. Math. Imag. Vis. 2*, pp. 261–272, 1992.

[41] F. Murtagh, "Interpreting the Kohonen self-organizing feature map using contiguity-constrained clustering," *Pattern Recognit. Lett. 16*, vol. 399–408, 1995.

[42] M. Y. Kiang, "Extending the Kohonen self-organizing map networks for clustering analysis," *Comput. Stat. Data Anal. 38*, vol. 161–180, 2001.

[43] E. A. J. Vesanto, "Clustering of the self-organizing map," *IEEE Trans. Neural Networks 11*, vol. 586–600, 2000.

[44] "MathWorks K-means." [Online]. Available: https://www.mathworks.com/help/stats/kmeans.html.

[45] "MathWorks Dendrogram." [Online]. Available: https://www.mathworks.com/help/stats/dendrogram.html?searchHighlight=dendrogram&s_tid=doc_srchtitle.

[46] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: An overview," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 2, no. 1, pp. 86–97, 2012.

[47] S. Wu and T. W. S. Chow, "Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density," *Pattern Recognit.*, vol. 37, no. 2, pp. 175–188, 2004.

[48] S. Theodoridis and K. Koutroubas, *Pattern Recognition*. New York: Academic Press, 1999.

[49] R. A. Fisher, "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS."