Contextual self-organizing maps for visual design space exploration

Brett Nekolny
Iowa State University

Contextual self-organizing maps for visual design space exploration

by

Brett Matthew Nekolny

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Co-majors: Human Computer Interaction; Mechanical Engineering

Program of Study Committee:
Eliot Winer, Major Professor
Amy Kaleita
Song Zhang

Iowa State University

Ames, Iowa

2010

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Visualization of an optimization problem (i.e the “design space”) becomes complex when the number of independent variables of the problem increases beyond two. Unfortunately, realistic optimization problems and their design spaces are often greater than two dimensions and therefore difficult to visualize. In order to create and display in greater than three dimensions it is necessary to use color, size, or symbols to show added dimensions. With the complexity in a visualization that uses these extra dimensional features, an observer is often overloaded with data and it can be difficult to grasp a firm understanding of the relationships therein. Furthermore, this solution of adding dimensions greater than three can only increment to a few dimensions beyond three and cannot achieve higher dimensions. There are currently two general areas for visualizing a higher dimensional design space: dimensional reduction, and individual variable comparison. With either of these methods, it is possible to display the resulting design space, or portion thereof, in a viewable dimensionality such as two or three dimensions. Self-organizing contextual maps provide a solution to this visualization problem by utilizing the dimensionality reduction capability of self-organizing maps and the display capability of the contextual map.

Self-organizing maps (SOMs) are able to map a design space of varying dimensionality to a two dimensional neuron lattice. The SOM can then be provided contextual information to display the similarities between areas of the design space either in terms of alphanumerical labels or visuals. This method will organize the numerical objective values associated with a design space to apply labels to the contextual SOMs. These contextual self-organizing maps allow the user to observe the entire design space in a two dimensional representation.

The ability to view an entire design space in this manner provides many advantages such as an understanding of the characteristics of the design space and optimization problem. This thesis will explain the work completed to apply contextual self-organizing maps to the visualization of optimization design spaces by:

1. Providing a visualization of the design space in two dimensions.

2. Extract characteristics of the design space using the resulting contextual map.

The resulting visual representations are achieved by generating a typical self-organizing map, and applying the objective function values as labels to each winning node. With a set of labels on each node, it was possible to calculate the mean, standard deviation, and minimum value for each node and display the results visually in the representation. The hue saturation value coloring scheme was used to display these three statistical measures using a single color for each node. The visual display of this coloring system makes the optimal node the closest to a brighter colored and more vibrant green colored node than the rest of the nodes in the map.

The results from this work show that contextual self-organizing maps can display valuable information about the design space that can then be extracted and applied to the solution of the optimization problem. The primary characteristics identified in the results are the modality of the design space and the optimal region within the design space. The results of this research will improve optimization by decreasing the time needed to solve optimization problems by gaining an understanding of the design space prior to a solution run.

1 INTRODUCTION

1.1 Purpose

The goal of this research is to organize data and create a visual representation of a design space in which designers can easily understand the characteristics influencing the data. This will also benefit optimization researchers by providing a starting point for a formal solution algorithm.

1.2 Design Space

A design space is a theoretical space in which the possible configurations of a design reside. In other words, if the design of an object requires three inputs it would be possible to plot those three input values in a three dimensional space. The resulting space would appear similar to Figure 1.

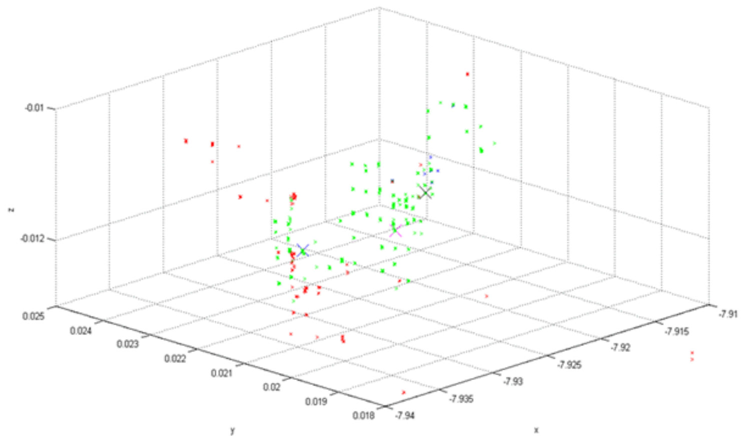


Figure 1 - Three dimensional cisualization of a design space (*Adapted from* **[1]**).

The sample design space in Figure 1 has the design variables plotted along the x, y, and z axes. The feasibility of each design is denoted by the color of the dot, which is either red representing infeasible and green representing feasible. This visualization becomes quite abstract and therefore proves difficult to extract information about individual design variables. The viewing perspective makes discerning the position of a point in space difficult. The use of color as a ‘fourth dimension’ to determine feasibility begins to overload the visual display. Overall, a visualization of a design space with only three design variables is not a simple visualization.

With these considerations, it is important to realize that many design and optimization problems (discussed in the next section) contain a much more complex data set and design space than provided in Figure 1. With these large and complex data sets, it is impossible to plot a high dimensional design space in the same manner as Figure 1; fortunately there are other methods to visually represent this space.

A design space can be visually represented through multiple methods that will be explained in further detail later, but two examples are: dimensionality reduction and individual variable comparison. One example of a dimensional reduction method is through visualizing the pareto frontier, or the area in which all of the objectives are as close to their optimum as possible. The pareto frontier can be displayed in a multidimensional plot, such as the performance space in Figure 2.

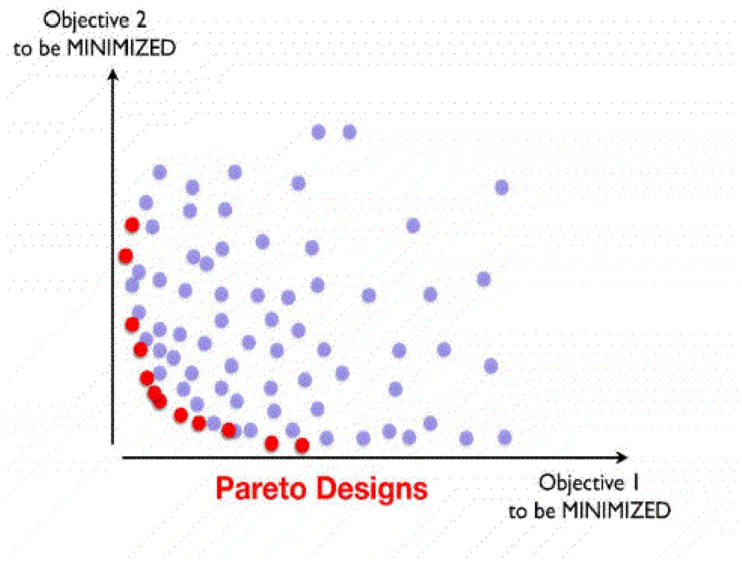


Figure 2 - Visualization of the pareto front (Adapted from **[2]**).

Figure 2 provides a display of the pareto frontier in red, where this area minimizes both objectives. The individual objectives are Objective 1 on the x axis and Objective 2 on the y axis. Typical design space representations place the independent variables on the coordinate axes, whereas here the objectives (e.g., weight, cost) are plotted against each other. Since a typical problem has far less objectives than independent design variables, this is one way to "reduce" the dimensions a designer has to view.

These techniques, dimensionality reduction and individual variable comparison, for viewing the design space provide a meaningful display in a viewable dimensionality for problems that range from one to many dimensions. Comprehension of the design space is invaluable to the designer. Tradeoffs can affect the resulting design more than can be represented in the theoretical space. For example, in the design of an aircraft the designer can save on cost by using steel but will severely decrease the fuel economy of the airplane because of the added weight. If this tradeoff is not captured in the mathematical problem, it will not be apparent in the resulting visual representation.

1.3 Optimization

The goal of optimization is to achieve the most desirable result from a given input, data set, or objective. The most desirable result can be in the form of a maximum or minimum, on a local or global scale. The optimization process can incorporate multiple objectives containing many variables that interact to create complex relationships. These relationships form mathematical problems that are either uni-modal and provide a single solution, or multi-modal and provide multiple solutions. Unfortunately, no single optimization routine is the perfect solution to every problem. Therefore, an alternative path to solution would involve extracting the characteristics of these complex systems or functions before choosing an optimization algorithm and solving for the optimal value.

Accompanying the varying complexity of optimization problems is a vast array of optimization algorithms. Two categories of common solution algorithms are numeric and heuristic. Numerical optimization techniques solve optimization problems by performing calculations that result in minimum objective values until convergence [3]. Alternatively, heuristic methods allow for an increase of the objective value in order to adequately explore the design space to discover the global minimum [4]. The caveat of this seemingly simple comparison between these two methods is that numerical methods specialize in uni-modal problems and cannot effectively solve multi-modal problems while heuristic methods are designed to operate on multi-modal behavior. A multi-modal problem can contain local minima and global minima. Even though heuristic methods are developed for the purpose of attempting multi-modal optimization problems, they can run into complications depending on the distance between local and global minima. These methods can get caught in local minimum values and never find the global minima. Therefore, it would be advantageous to have prior knowledge of the data represented by the design space to anticipate where the global minimum will be located.

To help illustrate this process, a formal optimization problem statement is described:

Min $F(x)$

S.T. $g_j(x) \leq 0 \qquad j = 1, \ldots, m$

$h_k(x) = 0 \qquad k = 1, \ldots, l$

$x_i^l \leq x_i \leq x_i^u \qquad I = 1, \ldots, n$

Specifically, “Min” poses the problem as a minimization problem of the objective function F(x). The constraints in the problem statement are $g_j(x)$, a function that is typically less than or equal to zero (i.e. an allowable upper limit), and $h_k(x)$, a function that must be equal to zero. The design variable x_i is bounded between the lower bound x_i^l and the higher bound x_i^h. Below is an example of a basic 2D optimization problem:

Min $F(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2$

S.T. g_1: $-2x_1 - x_2 + 4 \leq 0$

g_2: $-x_1 - 2x_2 + 4 \leq 0$

$-10 \leq x_1 \leq 10$

This problem could be to minimize the cost of product design (i.e. minimize F(x)) subject to two inequality constraints. These constraints could be a maximum stress and displacement that the product has to withstand during use. Thus, a solution to the problem is the combination of x_1 and x_2 that finds the lowest value for F while being below the bounds set by g_1 and g_2.

The complexity of the optimization problem can also determine which algorithm will efficiently converge on a solution. For example, a function's linearity and degree of curvature determines which method will be most effective in solving the problem. Characteristic knowledge of these optimization problems would provide a more straightforward solution path. Beginning with an understanding of the problem features can only benefit the solution process. Finally, it is important to remember that with optimization there is "No Free Lunch" [5], meaning that there will never be one best solution method and all options must be considered for the most desirable outcome.

1.4 Design and Optimization and Visualization

As discussed above, there are many techniques for solving optimization problems, and there are also numerous approaches to visualizing a design space. As the use of optimization in real-world design becomes more prevalent, more variables are introduced and an effective visualization tool becomes increasingly necessary. It is important to realize the benefits of viewing the design tradeoffs in a design or optimization function. Additionally, as the designers become more aware of the tradeoffs from one design to another, they can more effectively make the next generation designs.

There is a large effort pushing for visualization multi-dimensional data sets and problems. The research that is directly linked to optimization typically focuses on dimensionality reduction. The concept of dimensionality reduction is to display a complex multivariate system in a dimensionally viewable plot (three dimensions or fewer) while capturing the intricacies of each individual design variable. This can be achieved using a variety of statistical and heuristic methods such as principal component analysis [6] or Self-Organizing Maps (SOMs) [7]. Another method proposed for viewing the design space focuses on specific performance characteristics, or the objective of the design. The values of these characteristics or objectives can be observed at the beginning, end, or throughout the design or optimization process.

1.5 Motivation

The motivation for this project was to develop a visualization tool that allows examination of multi-dimensional design spaces in a simple and intuitive manner. This is crucial to the advancement of engineering design, and will allow designers to make quicker and more informed design decisions. The current solutions available provide abstract visualizations that limit the complexity of a problem in order to effectively view the design space.

With a method such as this, decisions such as proper solution method, good initial point, and other characteristics about the problem could be known. This information would lead to significant savings in time, project resources, and overall cost as well as lead to more effective designs.

1.6 Thesis Organization

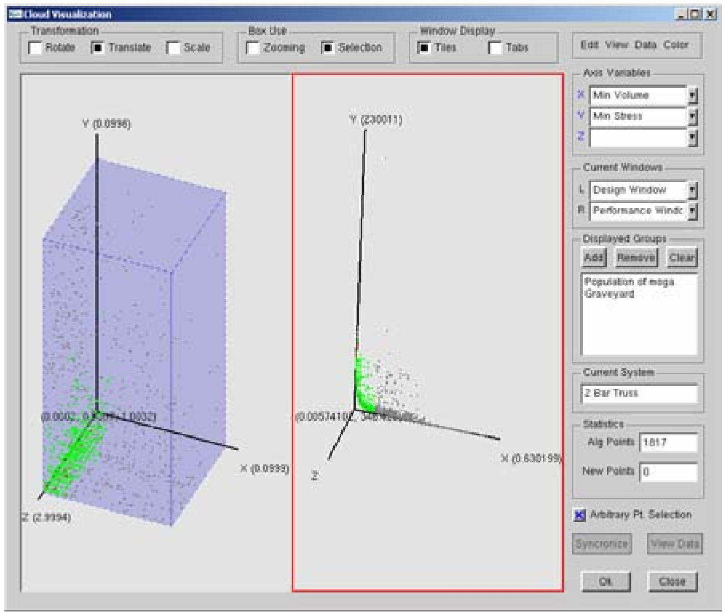
Chapter 2 contains the background research from the related areas of optimization, visualization, and self-organizing maps as well as present foundational research to this thesis. Chapter 3 describes the methodology, and explains the details of the technique, the process of achieving the final procedure, and the resulting application. Chapter 4 discusses the testing suite of optimization problems and results of this work. Lastly, Chapter 5 concludes by summarizing this work and discussing its implications as well as noting target areas for future work.

2 BACKGROUND

2.1 Optimization Visualization and Design Space Exploration

As described in Chapter 1, the visualization of large amounts of data and especially optimization design spaces is a difficult challenge. Many researchers have attempted this task and provided valuable methods and insight into the display of this information. This section will summarize this related research, provide an analysis of where this research area stands, and present a list of research goals for this thesis.

Cloud Visualization (CVis) [8] is an optimization visualization tool that allows a designer to view large amounts of data for the purpose of effective decision making throughout the design and optimization process. This design information is displayed three dimensionally in both the performance space and design space as shown in Figure 3. The performance space contains the most influential design variables that optimize the system based upon the desired performance criteria. This means that depending on the objective of the optimization routine, the performance space will display the design's proximity to the optimal solution. The performance space can also display the solution in terms of multiple objectives. The design space is shown three dimensionally and therefore can only display three design variables.



Axis Variables
X Min Volume
Y Min Stress
Z

Current Windows
L Design Window
R Performance Windc

Displayed Groups
Population of moga
Graveyard

Current System
2 Bar Truss

Statistics
Alg Points 1817
New Points 0

[✓ Arbitrary Pt. Selection]

Figure 3 - CVis Environment (*Adapted from* **[8]**).

Both the performance space and design space in the CVis software can be displayed in one, two and three dimensions as shown in Figure 3. The flexibility in dimensional visualization allows the designer to view single or multiple objective functions, as well as view the relationships between single and multiple design variables together in one space.

This environment provides an intuitive display to view not only the individual design variables, but a visual display of the objective values for each design. This work advances the capability of optimization visualization by providing an outlet for the large data sets generated in the process of design and optimization. Unfortunately, while this method does display the parameters and characteristics of the optimization process, viewing data in three dimensional space can be rather difficult. Additionally, when a designer has a high dimensional problem, they are only able to view three design variables at a time in each design space plot. This limitation would require the use of many plots to show design variables, which adds confusion.

Winer and Bloebaum first introduced the concept of visual design steering [9], a method to aid the design and optimization process by allowing the designer to interact with the optimization algorithm throughout the process. This design paradigm, visual design steering, was further examined with the goals of visualization and solution improvement [10]. The first step in design steering is to ‘rank and reduce’ the problem constraints and design variables. This is achieved by calculating the influence and effect of each constraint on the objective function. The least impactful constraints are then be ignored when the three dimensional plot is displayed. Next, a similar procedure is performed to determine each design variable’s impact on the objective function and its set of constraints.

The visual aspect of Winer and Bloebaum’s work, graph morphing, utilized three dimensional graphs to visualize design tradeoffs. In order to generate the graphical displays, structured data sets are generated so that specific cases of each design variable can be plotted in association with its constraints. Each visual representation then plots a maximum of three design variables, with the objective value shown using a color gradient and the problem constraints in green. Other design variables appear on graphical slider bars within their specified ranges. In Figure 4 each axis represents one design variable, the shade of the surfaces equates to a varying objective function value, and the green surfaces constrain the problem.

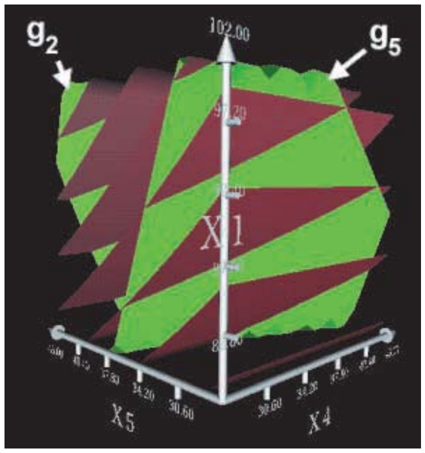


Figure 4 - GmorphVR (Graph Morphing) displaying three design variables, two constraints and a decreasing objective (*Adapted from* **[10]**).

Using GmorphVR, the designer is able to view the interplay between design variables, constraints, and the objective value. The slider bars can be moved to alter the display in real-time. This method clearly shows the trends of the design variables, and the tradeoffs with respect to the objective and constraints. The method for dimensionality reduction, eliminating the least important variables, also works to simplify the visualization and ease comprehension of the design space. This research validated the benefits of displaying the design information to the designer, because it increased design awareness and decreased solution times. Unfortunately, this method as with others is constrained by the ability to view only three design variables simultaneously. If the designer wants to view more correlations, it is necessary to construct multiple GmorphVR plots, and view them side-by-side.

The hyper-radial visualization method [11], developed by Chiu et al., is used to view the interaction of multiple objectives on an optimization problem. This method transforms a multi-objective problem into a two dimensional visualization in which the axes are both groupings of ‘manufactured’ objective functions. These manufactured objective functions are created by incorporating objectives together into one value so that the visualization can be displayed in two dimensions. The utopia point (both objectives are minimized) is represented at the origin, and the design space is projected using a radial method. The radial projection constructs a space in which each radius has an equal overall objective value, so every point on a specific radius has an identical objective value. Therefore, the goal is to achieve the innermost radius.

The Hyper-Radial Visualization method (HRV) allows the designer to add preference to individual objectives, which changes the weighting and display of the optimum. The individual weights are set using a likert scale, highly desirable to highly undesirable. After weights and preferences are set, the designer is provided a visualization by which they can view the tradeoffs of the two objective groupings. The designer can also set an uncertainty of the objective function value that allows similar objective values to be classified together as occasionally the accuracy of specific objectives is not of the utmost importance. An example plot can be seen in Figure 5.

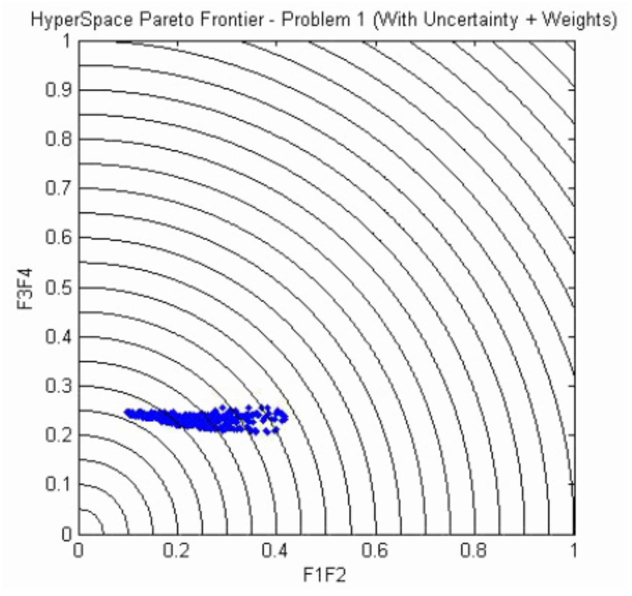


Figure 5 - Hyper-Radial Visualization of a multi-objective optimization problem using uncertainty and weighting (*Adapted from* **[11]**).

In Figure 5, the designer is able to visualize the performance space given the preferences set to each objective. This method provides an invaluable means to identify the preferences that a designer has for each objective in a multi-objective problem and aids in narrowing the possible design choices. The visualization method is not difficult to comprehend, because the resulting plot is two dimensional. This is a great method for viewing tradeoffs in objectives, but provides little means for interfacing with the individual design variables, as the focus of the display is on the objective values.

Swayne et al. developed the XGobi system for interactive dynamic data visualization [12]. The XGobi system specializes in visualization techniques for large amounts of data. The graphical tools available in the XGobi package range from: one dimensional to three dimensional plots, high-dimensional projection onto a two dimensional display, axis scaling, brushing, identification, line editing, and moving points. The one, two, and three dimensional plots are similar to the previously discussed plotting methods, but the projection method groups variables together on a given axis and plots them simultaneously, parallel coordinates shown in Figure 6

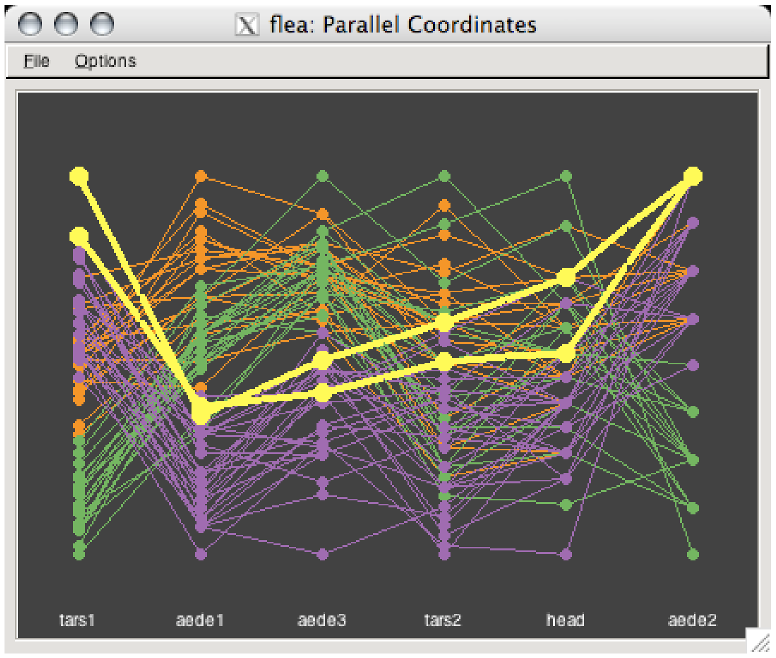


Figure 6 - XGobi dimensional reduction through parallel coordinates (*Adapted from* **[12]**).

In Figure 6, parallel coordinates are used to display a set of high dimensional data in a two dimensional space. Other tools available to sort through the vast amount of data that XGobi can display include brushing which allows the user to specify certain areas of the design space to display, identification which labels points, line editing which creates connections between points, and moving points which can allow the user to create a design space or design.

The ideas utilized in XGobi are essential to the visibility of a high dimensional data set, and they are used in unison to display this large quantity of information. Unfortunately, there is still a lot of information being displayed to the designer simultaneously, and individual relationships can still be difficult to discern.

Building upon XGobi, Stump et al. evaluated the application of multidimensional visualization to the design by shopping paradigm [13]. The design by shopping paradigm allows for the selection and refinement of designs throughout an optimization process. This method prompts the designer to choose acceptable ranges for design variables. These variables are physically displayed by extracting their values from the design space and plotting them individually in dimensionally viewable plots (one to three dimensions). When the user selects new ranges for the design variables, the optimization routine will resume solving along its new path until the user makes further modificaitons. This iterative loop will continue so long as the solution has not converged (optimal value or iteration count) or the designer has placed sufficient constraints.

Similar to XGobi, Stump et al. utilizes the brushing, coloring, and identification techniques to enhance data visualization. This method attempts to display further information of the design space by showing the interaction of multiple objective functions in the pareto frontier, the area where all objectives are being minimized. The plots in this method typically use glyphs which are a method of displaying points and trends in three dimensional space, see Figure 7.

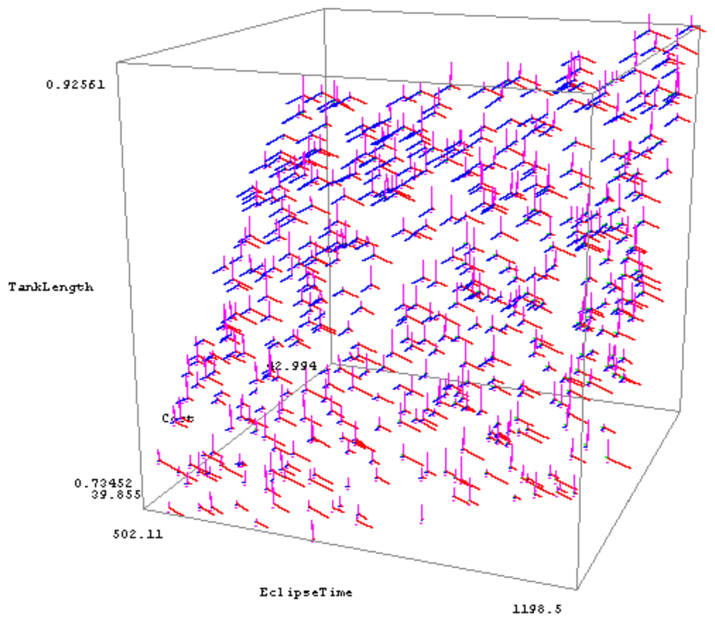


Figure 7 - Glyph plot of design space (*Adapted from* **[13]**).

Similar to previous methods, Figure 7 sufficiently displays the design space, but interpreting the interplay between variables is difficult. The difficulties present are again due to the perspective three dimensional displays. Fortunately, this research does decrease optimization times and provide a better understanding of the design space.

More recently, Stump et al. [14] have developed the ARL Trade Space Visualizer (ATSV) to display trade spaces, or tradeoffs within the design space. Many of the graphical options within this software are inherited from previous work, including but not limited to scatter plots, glyph plots, and parallel coordinates. The newest feature presented in this article was uncertainty visualization. A screenshot from the ATSV tool is displayed in Figure 8.

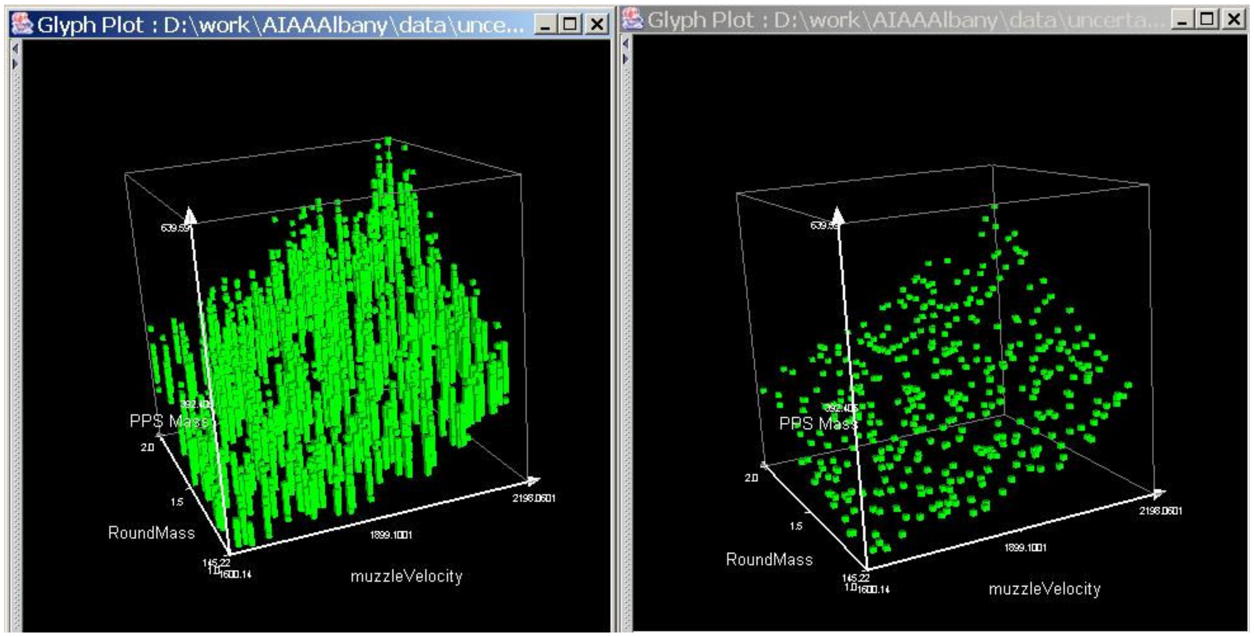

Glyph Plot : D:\work\AIAAAlbany\data\unce...
Glyph Plot : D:\work\AIAAAlbany\data\uncerta...
639.59
PPS Mass
2.0
1.5
RoundMass
145.22
1.0
1600.14
1899.1001
2198.0601
muzzleVelocity

Figure 8 - ASTV uncertainty visualization of rail gun data (*Adapted from* **[14]**).

The graphs in Figure 8 account for uncertainty in two design variable values and their influence on the objective. This uncertainty is displayed on the left by a bounding box for each point represented in the design space. The bounding box encompasses the uncertainty value within the point on the map. The image on the right displays only the mean value for each point represented in the design space rather than the group of points. While the visual display can only plot three dimensions simultaneously, the designer is allowed to adjust values of the other design variable and see their impact on the objective function value.

2.2 Self-Organizing Maps

Tuevo Kohonen developed a type of artificial neural network, the self-organizing map (SOM) [7], which he modeled after the learning process in the cerebral cortex. The theory is that the brain, or in this case the SOM, trains itself with a topological structure so that certain regions are more efficient at processing specific inputs or input types. The result is a trained network with prior experience that given a specific signal or location within the design space, the designs (set of design variable values resulting in an objective value) that fall within that region of the design space will be located in proximity to each other in the network. This network or map is typically a two dimensional map providing the ability to visualize high dimensional data in a low dimensional space. Figure 9 shows an example of a trained SOM to provide a connection between the SOM and the cerebral cortex.

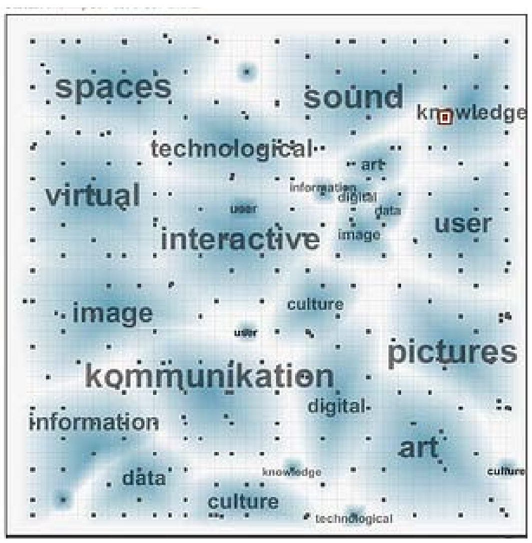


Figure 9 - Self-Organizing Map displaying the organization of various input types (*Adapted from* **[15]**).

The map displayed in Figure 9 was trained on a range of input types such as pictures, sounds, information, etc. The map takes these inputs and organizes them so that certain regions of the map specialize in specific data types. This can be clearly seen by examining the bottom right of the map, where pictures, art, and digital are grouped together; these three media fall into an overarching category of visuals so the map grouped them together. Unfortunately the map does not always train the data in the ideal manner, and therefore small groupings of the data categories can get separated which results in two identical labels in different sections of the map.

The map and training process maintain this topological pattern through the neighborhood function, or the effect of one neuron’s learning on its surroundings. An illustration of this concept can be seen in Figure 10.

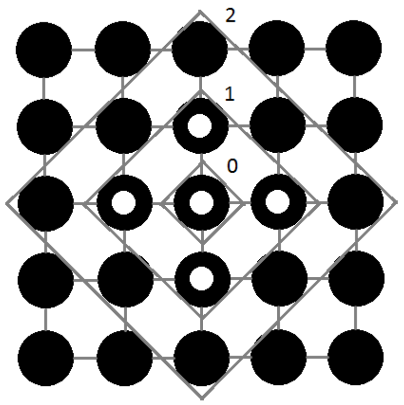


Figure 10 - Structure of a SOM node lattice, showing the neighborhood connections.

Figure 10 is an image of a lattice of nodes, with the black circles representing nodes and the horizontal and vertical grey lines representing the connections between nodes. The nodes with the white circles in them are an example of a node and its immediate neighborhood (center node and surrounding nodes) which respond to a specific input and influence the rest of the map through its neighborhood. As illustrated, the center node is the node best fit to accept the input, and it teaches its immediate neighborhood (denoted by 0 in the figure) about that signal or input, which teach its neighborhood (denoted by 1 in the figure), and so on. The influence of the learning decreases as the distance from the original neuron decreases, providing a higher concentration of similar nodes closer to one another (the center in this case).

It is important to mention that SOMs are always displayed as a lattice structure, but the representational shape of the node can vary.

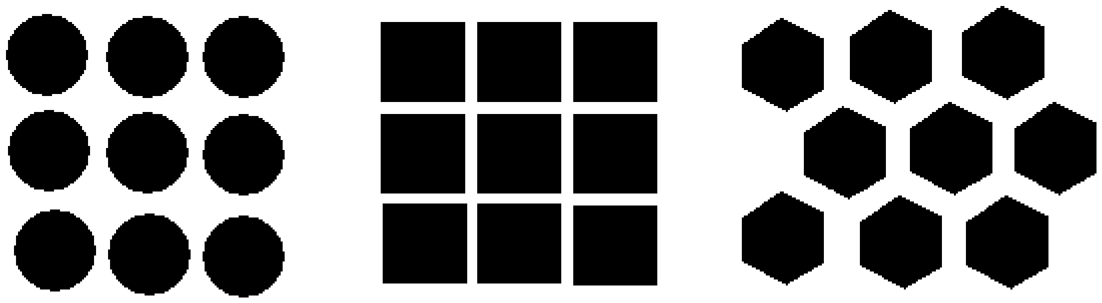


Figure 11 - Various representations of an SOM node lattice.

Figure 11 contains three of the most common display techniques for an SOM, with the nodes represented as circles, rectangles, and hexagons. The reasoning behind each shape is purely aesthetic. This thesis utilizes the hexagon representation of nodes because they provide a smoother color gradient across the map.

This method can be extrapolated to a variety of different data types, problems, and solution methods such as optimization, data mining, and artificial intelligence. A further explanation of SOMs can be found in Chapter 4. The next section will describe additional methods that can be applied to or with the SOM to enhance its visualization and structure abilities.

2.2.1 Beyond the Self-Organizing Map

U-Matrix

The U-Matrix [16] is a standard method of displaying the results of a trained SOM. The U-Matrix is larger than the node lattice of the SOM itself because it requires a display of the distance between each node. In other words, these distances are shown as connection strengths between nodes. This means that in order to display the U-Matrix, the Euclidean distance must be calculated between all adjacent nodes on the map. Once these values are obtained, the U-Matrix can be displayed with colors representing the distance between adjacent nodes, see Figure 12.

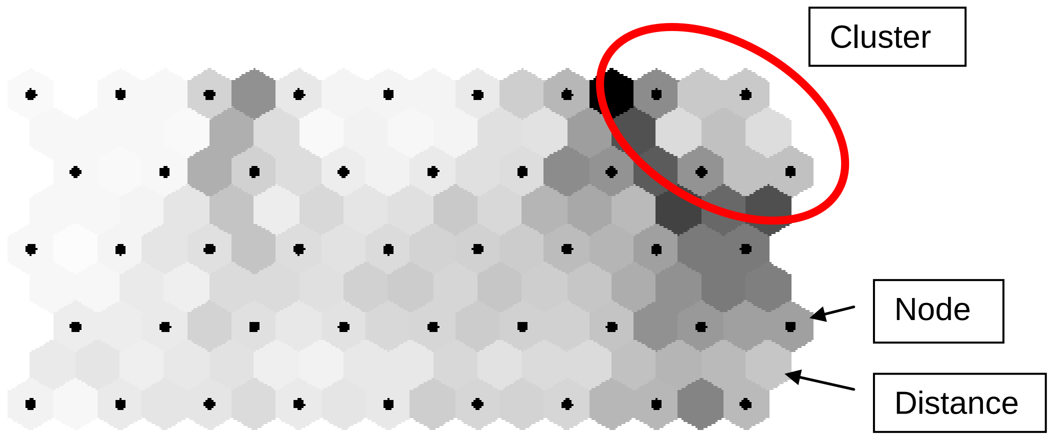
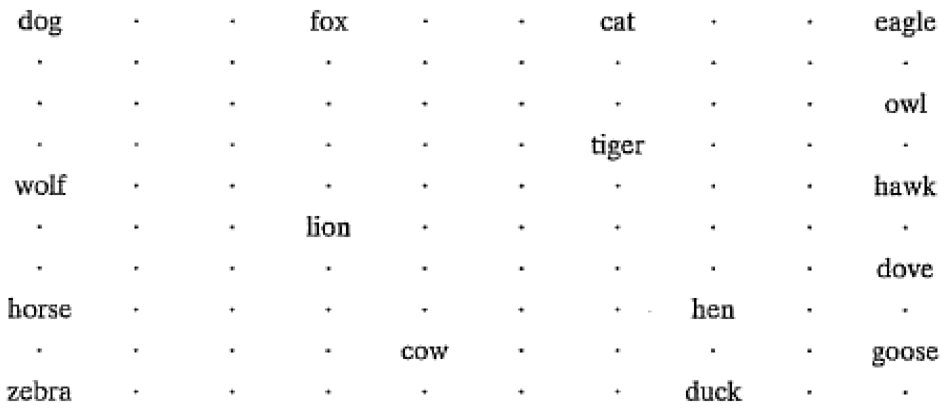


Figure 12 - U-Matrix of a trained self-organizing map highlighting a cluster of nodes with similar properties (*Adapted from* **[17]**).

The U-Matrix displayed in Figure 12 contains black dots on alternating hexagons to represent nodes in the lattice. The intermediate hexagons (between the black dots) are the color representation of the Euclidean distance between each node in the lattice. The U-Matrix provides a means of visualizing the entire trained map. It is possible to extract defined clusters of data in the design space that are represented by clusters in this U-Matrix visualization. For example there is a four node cluster in the top right of the map, see Figure 12. The dark colors represent a gap between clusters of data in the design space.

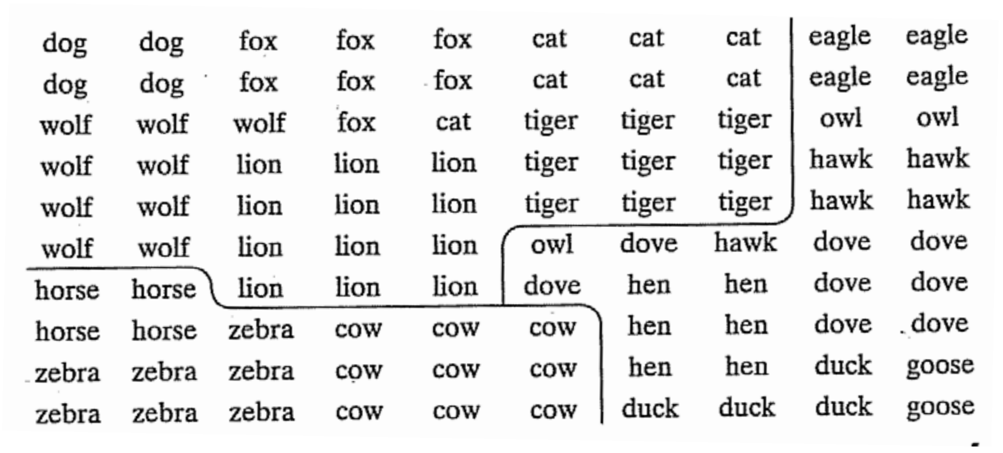
Contextual Maps

A trained SOM provides an effective model and clustered understanding of the input space, but the basic self organizing map provides no means of visualizing the map's value. One standard method of visualizing information on the SOM is through a method called contextual maps [16]. These contextual self-organizing maps show the SOM lattice in two dimensional space, and apply 'labels' to the various nodes in the map. These labels can be used to describe the clusters of the input space or used to comprehend the similarity of various inputs to the map. Haykin provides an easy visualization of a cluster map through an example training set where the inputs used to train the SOM are animal attributes such as: size, number of legs, types of feet, and movement ability (swim, fly, run, etc.). Each input vector of these characteristics also carried with it a contextual label, or animal name. Figure 13 and Figure 14 display the contextual map resulting from this training.



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

Figure 13 - A contextual map displaying the results of training an SOM on animal characteristics (*Adapted from* **[16]**).



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 14 - Filled in contextual map showing the separation of hunters, peaceful species, and birds (*Adapted from* **[16]**).

The contextual map, Figure 13 and Figure 14, show the 'winning neurons' for each input vector by assigning a label to that neuron. The contextual map is generated as a post-processing of the trained self-organizing map. Once a map's training is complete, the data is fed back into the map one last time for contextual labeling. Just as the Euclidean distance was calculated to find the winning node in the training of the SOM, the

Euclidean distance is calculated again for contextual labeling. The winning node for each input vector is assigned the 'label' of that input vector (in this case animal name). After all of the input vectors have been run through the map one final time, it is possible to see a resulting map similar to this form.

Further examination into the structure of Figure 13 can result in identification of three distinct quadrants to the map: birds, peaceful species, and hunters. These regions are shown by the map but must be identified by the researcher. These separate regions are properties of the input data structure, meaning that animals with claws are more likely positioned in the hunter area, and hooves are most likely in the peaceful species. This visualization method provides a means to view similar data types; for example, if one wanted to examine an animal similar to a hen, looking at hen on the map would point toward ducks and geese as similar species.

Generative Topographic Maps

The Generative Topographic Map (GTM) [18] was developed by Bishop and Svensen. This method is based upon the SOM as well as other neural networks. The benefits of the GTM over other methods are the adaptive training components and the topographic preservation.

The adaptive training components mean that it solves for its own training parameters through statistical analysis of the training, two parameters are learning rate and neighborhood size. The self generated parameters make certain that the map reaches a full trained state, but never becomes over trained. This can be an important characteristic, because over fitting of data is a potential in other learning algorithms.

The GTM also preserves the topology of the input space by first running statistical analysis on the input data set, and then mapping itself to the resulting probabilistic distribution. This is achieved by solving for the best coverage of the input space using Gaussian spheres. When the Gaussian distribution is decided, the GTM is overlaid on the Gaussian locations and the map is created, see Figure 15.

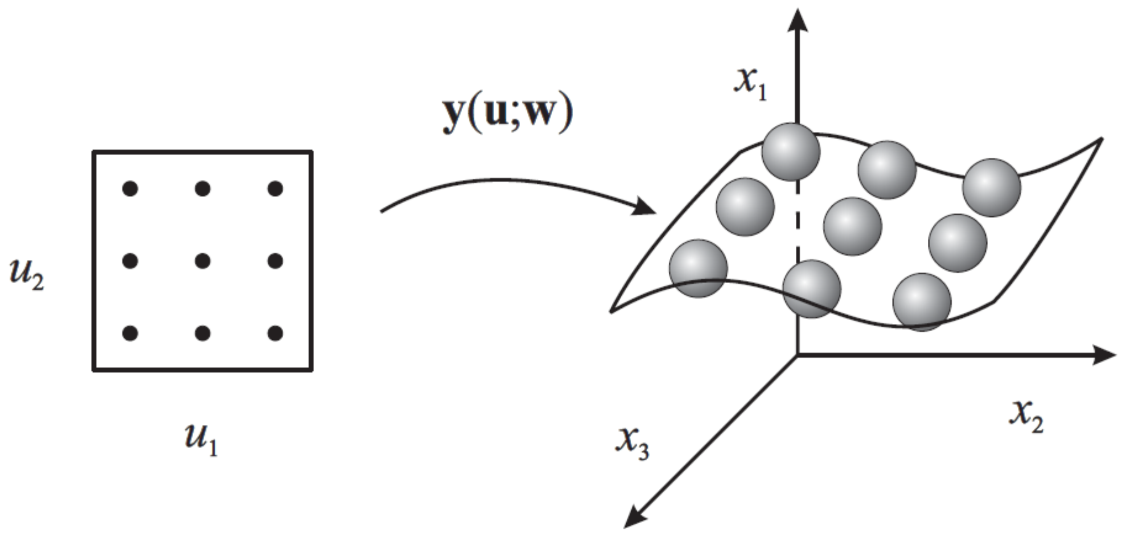


Figure 15 - A GTM node configuration on the left, and the corresponding mapping to the input space on the right fitting to the Gaussian spheres (*Adapted from* **[18]**).

Figure 15 displays the predetermined map for the GTM, and the calculated Gaussian spheres distributed across the data space. Then it shows the overlay of the network onto the input space. This method provides an elegant solution to customizing neural network training parameters and visualizations in a viewable space, but these advantages come at the cost of computations. While the GTM can be computationally expensive, the benefits can outweigh this aspect given the correct problem or data set.

Holden and Keane provide the GTM as a solution to the aerodynamic design complexities [19]. In this work, the authors were able to display the results of an aerodynamic design with the goal of a minimum drag coefficient. The data set for this work was developed using computational fluid dynamic simulations, and then provided to the GTM. The GTM then fit to the design space, and provided an approximation algorithm so that any position in the input space would return a value, even without a training sample at that location.

This interpolation technique that GTM can use provides a continuous mapping between input and output space, and an ability to display this relationship. The basis for the interpolation uses the distance between the input and the nearest Gaussian spheres, and interpolates between the spheres. A typical GTM display is similar to that of a two dimensional display of an SOM, see Figure 16.

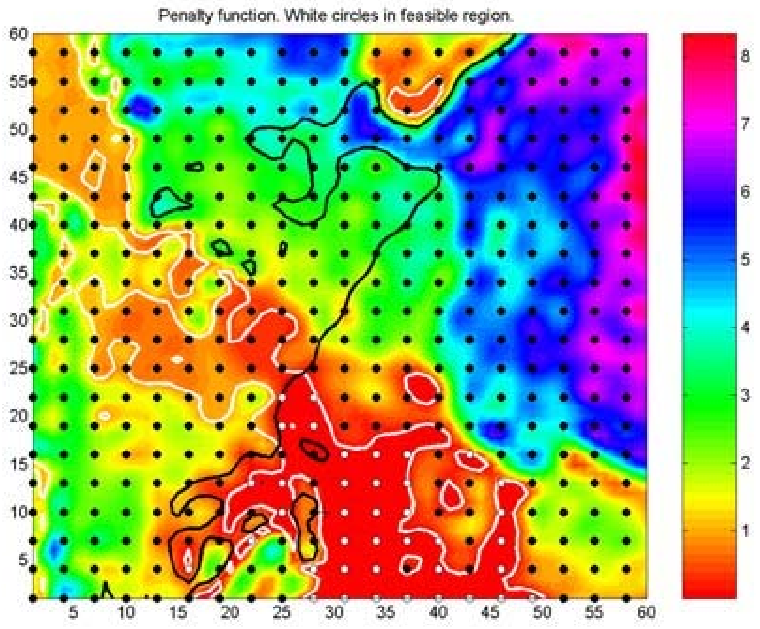


Figure 16 - The resulting trained GTM displaying color based upon the objective value, drag coefficient (*Adapted from* **[19]**).

Figure 16 displays the behavior of the design space in a two dimensional grid, allowing the user to choose the optimal design and value. This study gave examples of 8 dimensional and 14 dimensional design spaces, but made mention that the 8 dimensional space was much more effective than the 14 dimensional space. As a result, the authors ran the 14 dimensional space to fit on a four dimensional topographic grid, Figure 17, rather than the standard two dimensional grid shown in Figure 16.

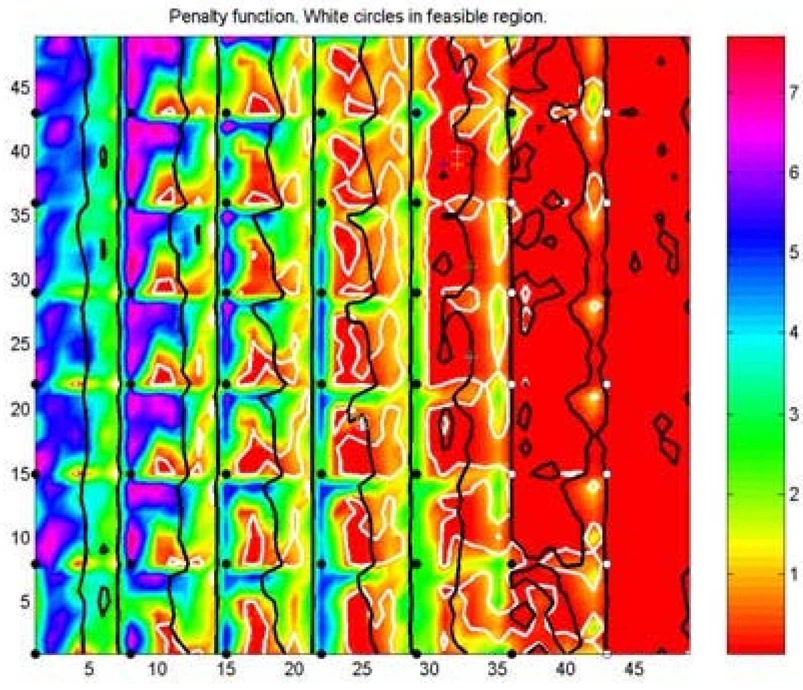


Figure 17 - A four dimensional GTM displaying a 14 dimensional design space (*Adapted from***[19]**).

As seen in Figure 17, attempting to visualize the four dimensional space becomes increasingly complicated. This work proves very beneficial to the advancement of design space visualization, but more research needs to be focused on the applicability of GTMs to high dimensional problems.

Equalized Orthogonal Mapping

Meng and Pao created a new method, Equalized Orthogonol Mapping (EOM) [20] based upon the Kohonen SOM. The EOM functions using a covariance matrix, with a closer tie to principal component analysis (PCA) where as the SOM builds upon the strict input values. The benefits of working with the covariance matrix are the preserved topology throughout the map. In some cases, an SOM could be under-trained and distant nodes could map to similar input values. Topology preservation in the case of EOM prevents such an occurrence from happening, as each node in the lattice is expected to be in a position relative in the design space to its adjacent nodes. Because of this strict topology preservation from two dimensional lattice space to n-dimensional design space, the EOM possesses the ability to perform interpolation between neurons similar to that of GTM.

2.2.2 Self-Organizing Maps Applied to Optimization and Visualization

There has been a substantial amount of work that utilizes self-organizing maps for the purposes of optimization. The research in this area can be encompassed in two different categories: visualization of the entire design space and visualization of the extracted design variables. These categories can be simplified with an understanding that viewing the trained SOM provides a visualization of the entire design space, and single variables can be viewed and compared by extracting the component maps from the trained SOM.

Visualization of the Dimensionally Reduced Map

SOMO, Self-Organizing Maps for Optimization [21], were developed by Su, Shao, and Lee to solve optimization problems. This method utilizes the SOM training to 'learn' the optimal value for a given objective function. The modified training algorithm for SOMO uses the traditional winner-takes-all training; however, SOMO replaces the Euclidean distance calculation (to find the winner) with the optimization objective function. The winning node search would proceed by calculating each node's 'distance' by evaluating the objective function ($F(x)$) using each node's weight vector (w_k) as inputs ($F(w_k)$) where the resulting F is the node's 'distance'. After the winning node is calculated, the neighborhood is updated using the weight update equation (Equation 4) and an input vector, $x = \{x_1 = 1, x_2 = 1, ... x_k = 1\}$.

After training the SOM on a specific objective function, SOMO provides a method to visualize the objective function in a three dimensional space. Just as standard SOMs perform dimensional reduction, SOMO allows the visualization of a high dimensional function in a three dimensional space. This display is accomplished by computing an objective function value (F) for each node in the lattice. The result is then plotting the 2D lattice on the x-y axis with the z axis showing the F value across the map nodes. As all SOMs do, SOMO groups similar regions of the design space, and reduces the complexity of the visualization while still representing the structure of the objective function, see Figure 18.

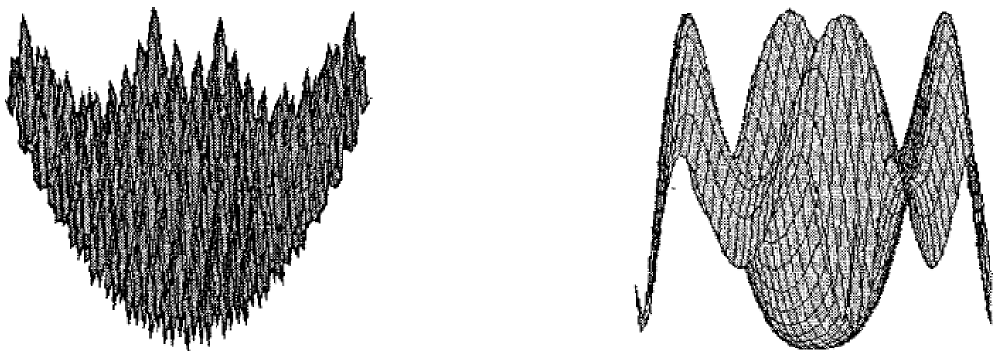


Figure 18 - Display of 2D Rastrigin function compared to the SOMO trained on a 50 dimensional Rastirigin function (*Adapted from* **[21]**).

While this method was presented as an efficient solution to optimization problems there are drawbacks: in order to use this method the user is required to provide a continuous objective function. Additionally, the presented results were only scaled to 30 dimensions, and may not be feasible at a higher dimensionality.

A similar method was developed by Milano, Koumoutsakos, and Schmidhuber to display the results of a trained self-organizing map [22]. This method trained the SOM in a standard manner, using values within the design space of the desired objective function. After the training was completed, the resulting map had weight vectors that represented the design variables of the objective function. In order to display an objective value for the node, these weight vectors were used in the objective function, and the resulting function value was assigned to that specific node in the lattice. With the lattice structure and objective function evaluation values for each node, it is possible to plot the dimensionally reduced objective function in three dimensional space. The resulting display would appear very similar to Figure 18. Again, the display of the map in three dimensions clearly shows the objective function, but does require that the data set also be accompanied by an objective function for this method to be possible.

Obayashi and Sasaki utilized self-organizing maps to visualize and explore the design space of supersonic wings and wing-fuselages [23]. The data for this visualization was generated using computational fluid dynamic simulations of each design so there was no objective function. The researchers utilized the contextual self organizing maps, but used images for the contextual labels. These images were various design profiles, or wing designs, see Figure 19.

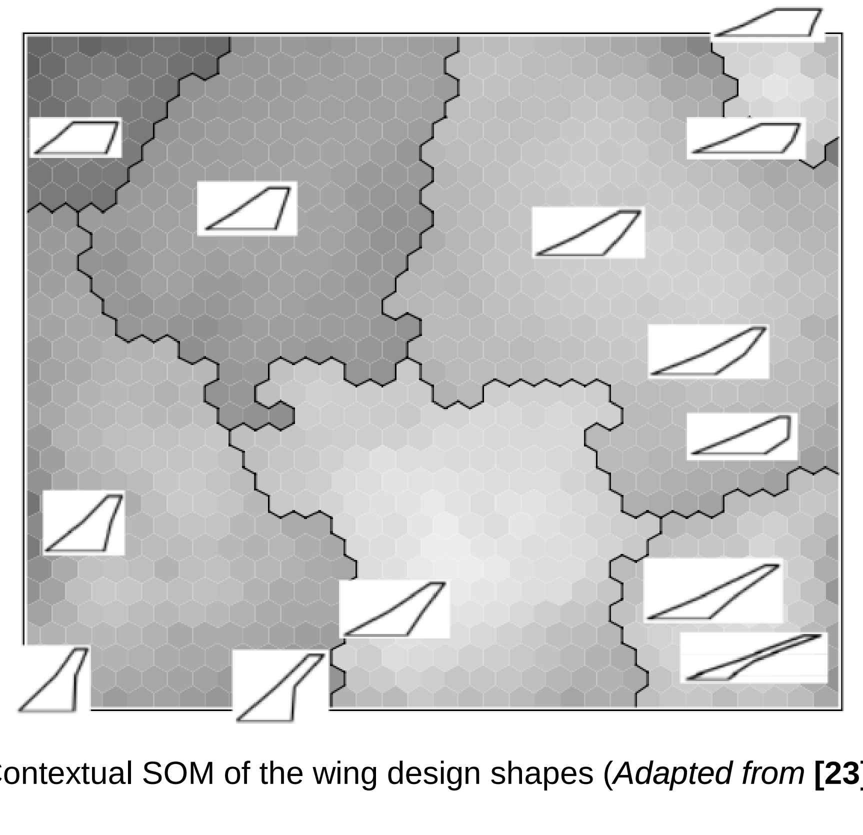


Figure 19 - Contextual SOM of the wing design shapes (*Adapted from* **[23]**).

As seen in Figure 19, the SOM separates out the design space and places similar designs in similar regions. The top left and bottom right corners of the map show two extremes for wing designs, short and long respectively. This visualization clearly lays out the various designs and also provides a gray scale color for each node depending on its objective function value. The coloring scheme used was a lighter color for minimum objective function value, and a darker color for a higher objective function value. The objective function value in the case of the wing design is a combination of the drag, bending moment, and pitching moments. In this case, the middle area of the map provides an optimal design because it is the lightest colored area.

Obayashi and Sasaki were able to investigate further by displaying only the individual objective values (drag, bending moment, and pitching moment) so that the user could gain an understanding of which objectives were best shown in each design. These maps look similar to Figure 19 with the exception that the color patterns are different as only individual objectives are being viewed. Furthermore, they were able to extract individual design variables from the map, such as leading-edge sweep angles, root-side chord lengths, and wing cambers. These resulting maps displaying the individual variable patterns are beneficial in exploring the explicit relationships between a design variable and other design variables or the objective function value. This method of viewing individual maps extracted from the entire trained SOM leads in to the next section which discusses research utilizing the design variable extraction and visualization ability of the SOM.

Extracted Individual Variable Visualization

Matthews used the abilities of self-organizing maps to extract the relationships between variables and show them in individual maps [24]. The author generated test points for sample problems to train the map and display the results, he was able to apply this method to a conceptual design project as well as a gas turbine and an aircraft wing. Data was generated for these cases by running physical simulations of equipment, thereby providing a data set without an explicit objective function. Matthews trained his SOM in accordance of traditional SOMs, and used the information in the resulting map to extract relationships between the design variables [25]. Using this trained map it is possible to isolate an individual design variable at each node, and display a map showing the variability in that single design variable. The map can be colored by interpolating between two colors (such as red and blue in Figure 20) to assist in displaying this variability. This method can be further explored by extracting multiple variables from the trained SOM and viewing the relationships between these variables to visualize implicit relationships within the data set, Figure 20.

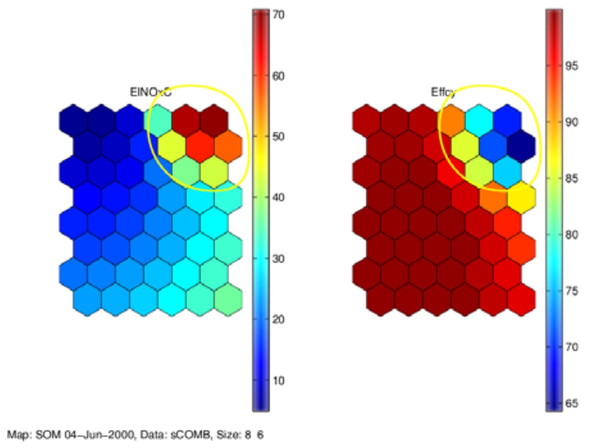


Figure 20 – Independent variable maps extracted from the trained SOM showing an anti-correlation, or negative correlation, between two variables (*Adapted from* **[24]**).

The ability to explore the correlations between design variables is extremely useful as it can provide an understanding of which design variables contribute to more or less of the variability in an outcome. Matthews showed that these maps were able to detect relationships between variables that were otherwise unnoticed after using the linear principle component analysis.

Due to the number of complex relationships available within one data set, the author used a heuristic called the Tanimoto metric [25] to detect these complexities in the extraction maps. The Tanimoto metric was able to successfully identify potential areas of interest, but was unable to comprehend these relationships on its own. This allowed for the user to examine a reasonable number of maps to visualize these relationships. This work provided significant potential advancement of self-organizing maps in the areas of design and optimization because the method for examining individual maps could give further information into the structure of the SOM. Unfortunately, due to the potentially highly dimensional data sets, there can be a large number of extraction maps to analyze.

2.3 Color Visualization

For the purposes of visualization, color provides an intuitive alternative to a numerical display. Many of the projects mentioned in the background section have used a color interpolation to display the variation of a variable’s value. Traditionally, most applications that use color to display the relative value of a variable interpolate between two colors, such as red and green. This interpolation functions that a high value will be on one end of the spectrum, i.e. red, and a low value will reside on the other end, green. When a value is in the middle of the spectrum, it is colored yellow. An example of this interpolation between colors can be seen in Figure 21.

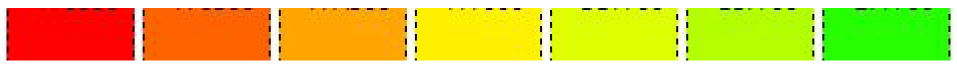


Figure 21 - Color interpolation from red to yellow to green.

This system for color interpolation works well to explain one set of values, but when the data sets require the display of additional information this color interpolation method is not sufficient. This research presents contextual self-organizing maps which need to convey additional information beyond the mean of a node such as the standard deviation and minimum value contained in the context of a node. In order to accomplish this task a different color scheme, Hue Saturation and Value (HSV) [26] is utilized.

The principles of the HSV coloring scheme begin with a color interpolation on the hue, or the base color being displayed. This functions exactly the same as Figure 21, in this case between the colors red and green with yellow in the middle. This color interpolation is determined by the numerical value assigned to the hue. The hue can range in colors across the color spectrum, but is limited to solely red to green in this case for simplicity of the display.

After the hue, or base color, is set the next parameter is the saturation of a color. The saturation of a color determines the amount of pigment allowed in a color. A color with less pigment will appear closer to white, and a color with more pigment will appear closer to the base color or hue. Figure 22 and Figure 23 display the result of interpolating between full saturation, 1, and zero saturation, 0, in the hue colors red and green.

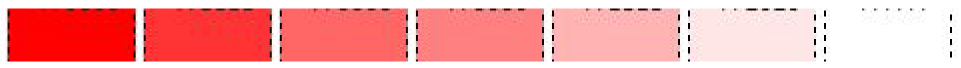


Figure 22 - Saturation interpolation from high to low saturation in the color red.

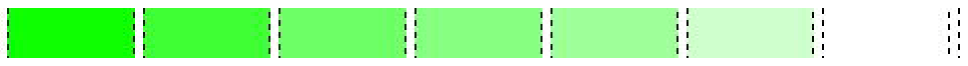


Figure 23 - Saturation interpolation from high to low saturation in the color green.

As displayed in Figure 22 and Figure 23, higher saturation values display more pigment within a color. When a color has more pigment it resembles a color closer to that of its base color.

The last component of the HSV coloring scheme is the value. The value of a color is determined by the amount of brightness allowed in a color. When a color has a low value, it will appear darker because the lack of brightness. Alternatively, when a color has a high value it will appear closer to its hue color. Figure 24 and Figure 25 display the effects of varying the value of the colors red and green.

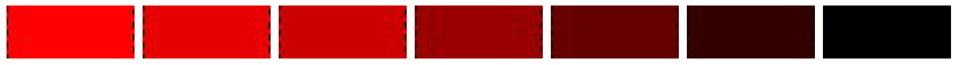


Figure 24 - Value interpolation from high to low value in the color red.

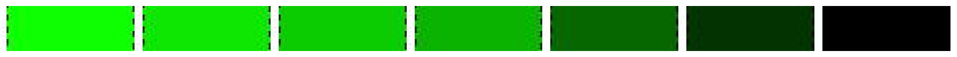


Figure 25 - Value interpolation from high to low value in the color green.

The change in a color’s value changes the brightness of a color, so as colors become closer to a zero value they all begin to blend into a black color.

This coloring scheme can surpass the abilities of basic color interpolation because it has the potential to explain more than one value. The HSV color method, with its added components of saturation and value can take into account other variables and provide results by displaying varying ‘qualities’ of a hue or color. For example, a hue of red can become a closer to grey if it has a low value and a low saturation; this is because it has a lack of pigment and a lack of brightness. So while the color is still red, is it possible to gain excess information about the values used to make that color. The application of this coloring method to this work is described in further detail in Section 3.1.4.

2.4 Research Issues

From the previous work in this subject area, there are many solutions to tack the various problems related to visualizing the optimization design space. The issues left unresolved in this field are the capability to effectively view the entire design space, the ability to draw conclusions about the characteristics of the design space, and a method that can be incorporated into a variety of optimization stages as well as a variety of data sets.

This research will focus on two research issues:

1) Developing a method that can effectively display an entire design space in one intelligible representation
2) Conveying information about the optimization problem including its modality, linearity, curvature, and an initial search region.

In order to complete these goals, this paper will describe the process of implementing methods from previous research while developing new schemes for displaying information to the designer.

3 Methodology

3.1 The Self-Organizing Map

Self-organizing maps [7] are a class of neural networks developed by Teuvo Kohonen for the purposes of data classification. These maps use an unsupervised competitive learning process called ‘winner-takes-all’ to 'learn' the input space or input data. The structure of the SOM is typically a two dimensional lattice of nodes, but can vary from as low as one dimension up to any number of dimensions. The two dimensional lattice can be utilized for dimensionality reduction by which higher dimensional data is *mapped* to this lower dimensional space or node lattice. The benefit of using a two dimensional node lattice is that it is easily viewable, being two dimensional. When viewed, these maps can display individual neurons in many forms: squares, circles, hexagons, and more. The reason for the variety in node representation is simply visual appeal and preference of the designer.

Depending on the dimensionality of the node lattice each neuron will have a variable number of connected neurons, but in the case of a two dimensional lattice there are four immediately connected neighbors.

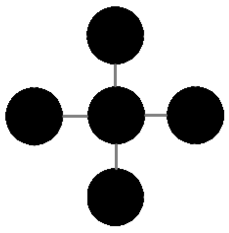


Figure 26 - A node and its neighbors.

The process of learning entails the SOM being mapped to an input space made up of input vectors. An input vector holds the structure of x_n, see Equation (1). The input space in the case of design would be the design space, and an input vector would be one location within the design space. Once an input space and SOM map size are provided, each node can be assigned a weight vector, which is its location within the design space. The weight vector will be of the form w_j, see Equation (2), where k from the input vector and the weight vector are equivalent.

$$\mathbf{x = <x_1, x_2, ..., x_k>} \quad \textbf{(1)}$$

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

Equation (1) shows a sample input vector, x, where k is the dimensionality of the input data or the number of design variables. Equation (2) is an example of a node's weight vector, where k is again the number of weights (dimensionality of the input data). In order to create a map (the node lattice), the dimensionality of the input space must be known, and the size and dimensionality of the SOM must be chosen. As previously discussed, SOMs are typically two dimensional but can be any size.

With an input data set and an SOM containing empty weight vectors, the next step is to randomly initialize the weights of these node weight vectors. After the weights are initialized, the map is ready to be trained. Training has two phases: ordering and convergence. Both phases of training proceed by selecting an input vector at random and feeding it into the map. When the map is being trained on an input vector, the Euclidean distance between the input vector and every node in the map is calculated, see Equation (3). Each training phase consists of many iterations, and one training iteration is defined by the use of the entire input data set. Training begins by using Equation (3)

$$\textbf{Distance} = \text{sqrt}((x_1 - w_{j1})^2 + (x_2 - w_{j2})^2 + \ldots\ldots + (x_k - w_{jk})^2) \qquad (3)$$

The node with the least distance is declared the 'winner'. After finding the winner the next step is a neighborhood update, at which time the weight values of each node in the lattice will be updated. The node update equation is expressed in Equation (4).

$$w_{j(n+1)} = w_{j(n)} + \eta_{(n)} \cdot h_{j,i(x)}(n) \cdot (x - w_{j(n)}) \qquad (4)$$

η in Equation (4) is the time varying learning rate that determines the allowable influence of the input vector, or how much an input vector can modify the node weight vector. The learning rate decreases throughout the training so that the map becomes more refined, and is subject to less drastic change. The $h_{j,i(x)(n)}$ in Equation (7) is the neighborhood influence for the training iteration. The neighborhood influence, like the learning rate, decreases through the course of training to help refine the map. At the beginning of training, the neighborhood encompasses the entire map, so each input vector will affect the whole map. Near the end of training the neighborhood will be limited to only one node, and possibly its immediate neighbors. Equations (5) and (6) explain how η and $h_{j,i(x)}(n)$ vary with time.

$$\eta\ (n) = \eta_0 * \exp^{(-n/\lambda)} \tag{5}$$

$$\sigma(n) = \sigma_0 * \exp^{(-n/\lambda)} \tag{6}$$

$$h_{j,i(x)}(n) = \exp(\ -\ \text{Distance}^2\ /\ (2 * \sigma^2(n))) \tag{7}$$

A general guideline for choosing the time constants is given in Neural Networks: A Comprehensive Foundation [16]. During the ordering phase the learning rate, η (n) is set at or below 0.1, and during the convergence phase η (n) is set at or below 0.01. The initial neighborhood width, σ_0 is set to the entire map for the ordering phase, and then adjusted to be approximately two nodes for the convergence phase of training.

The map is created with a predetermined size, and each node has weight vectors whose length matches the dimensionality of the input space. Ordered training begins and iterates for a set number of iterations, where each iteration runs the entire input data set through them map. Convergence training follows ordering, but runs approximately 20 times the number of iterations and focuses less on drastic changes to the map, but subtle modifications to individual nodes. In other words, after sufficient training the nodes in the SOM will each occupy a region of the input space so that the entire input space can be represented by the trained SOM. This result is a map that is topologically trained, meaning adjacent SOM nodes will reside in adjacent locations within the trained map. Additionally, given a new input vector the SOM will respond with the appropriate, least distance node in the map. This input and output space (neuron space) is shown in Figure 27.

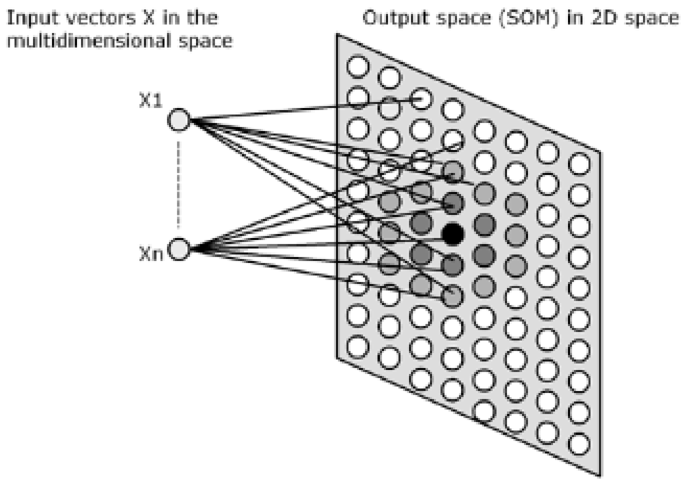

Input vectors X in the multidimensional space
Output space (SOM) in 2D space
X1
Xn

Figure 27 - Diagram of the Self-Organizing Map *(Adapted from* **[16]***).*

Figure 27 displays the affects of training from one input vector. It is possible to see a 'winning node' for the given input vector denoted with a black colored node. The neighborhood surrounding the winning node is colored various shades of gray depending on the neighborhood node's proximity to the winning node. The nodes that are colored white in Figure 27 will be unaffected by the information gained at the winning node, because they are out of the neighborhood of influence and the resulting change to their weight vectors will be zero.

The result of a trained map is that given an input vector, x, one node in the SOM will activate. The activation of a node can be used for a variety of goals such as response action and data comprehension. The response action means that if an input is provided to a trained map, it will use its knowledge to produce the same result as previous inputs with similar characteristics had for responses. The SOM also represents a dimensionally reduced (two dimensional) representation of the input space, which has many applications especially in the area of visualization. The SOM provides additional benefits such as a continuous input space, a spatially discrete output space of neurons, and the ability to capture non-linear data sets, which many other statistical methods, such as PCA, are unable to tackle.

3.2 Contextual SOMs Applied to Design Space Exploration

3.2.1 Map Generation

Generating a self-organizing map requires initializing an empty map with a random network of nodes. The user defined network size sets the number of nodes in each row and column of the lattice. The results presented in Section 4.2 are built with a network size of 15 rows and 15 columns because that size was discovered to be an appropriate fit for the variable data set sizes. After an empty network is created and a data set dimensionality is provided, each node in the network can generate its weight vector using Equation 4. As mentioned, the weight vector of each node is of the same dimensionality as the input vectors. The result of this is a randomly generated map, which needs to be trained as shown in Figure 28.

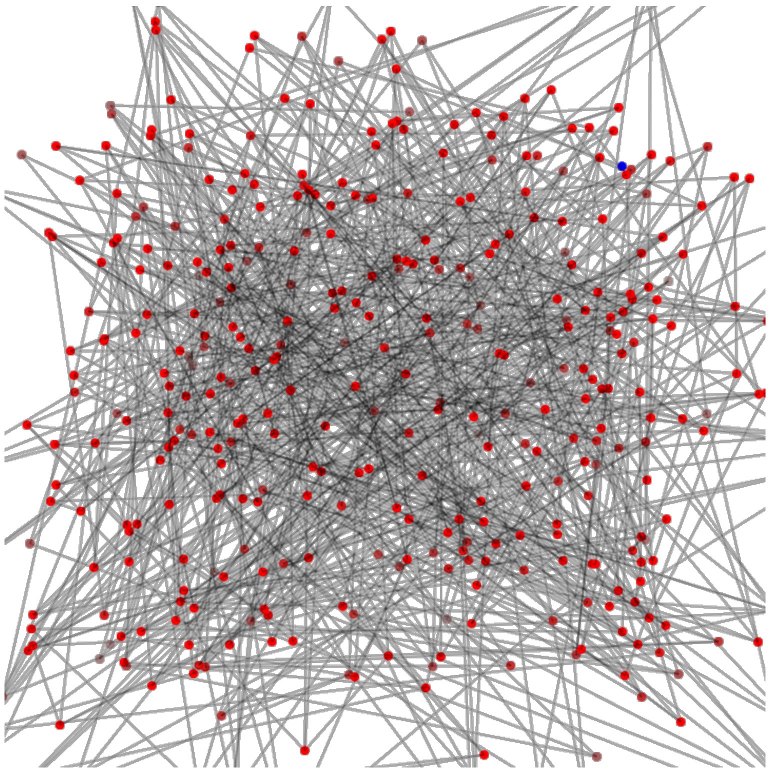


Figure 28 - Randomly initialized two dimensional SOM (*Adapted from* **[27]**).

Figure 28 displays a randomly initialized map with a two dimensional weight vector, shown in a two dimensional space. The red dots correspond to each of the nodes in the lattice, and the lines between the red dots correspond to the connections between the nodes. These connections initially appear scattered because there is no structure to the untrained map, but will later resemble a lattice structure after training is underway. The connections between nodes are important because they influence which nodes will be affected most after a winning node is found. A sample map can be seen in Figure 29.

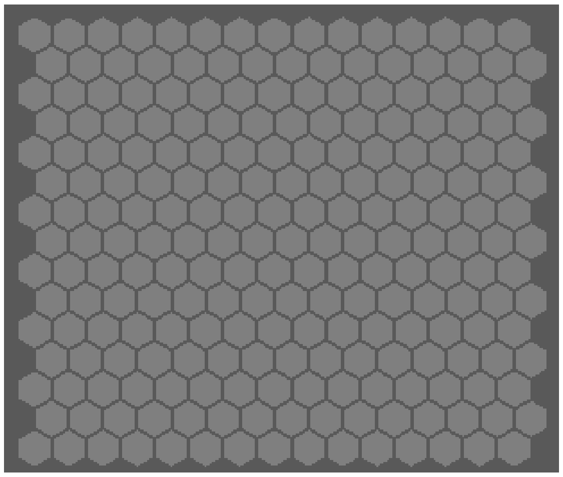


Figure 29 - An empty SOM prior to training.

3.2.2 Training Process

After a map is generated and a dataset is provided, the next step is to train the map. Training begins with the ordering phase. The ordering phase takes the randomly organized map and begins the organization so that its structure resembles that of the input space. All of the training parameters in this section: initial learning rate, initial neighborhood width, network size, and number of training iterations are referenced from “Neural Networks: A Comprehensive Foundation” [16]. Often the number of iterations required to change a randomly generated map to an ordered map is approximately 1000 iterations, where one iteration is the map being trained on each member of the input dataset. The initial neighborhood size is set to a neighborhood of the entire map during the ordering phase of training; this means meaning that at the beginning of training each node in the network is affected by every input vector. This neighborhood width decreases over time according to Equation 6. The initial learning rate is typically set to a value around 0.1 [16], and also decreases over time according to Equation 5. These values are set to ‘higher’ values in the ordering phase so that the map can quickly gain a topology that resembles the input space. An example of a map that is completing the ordering phase of training can be seen in Figure 30.

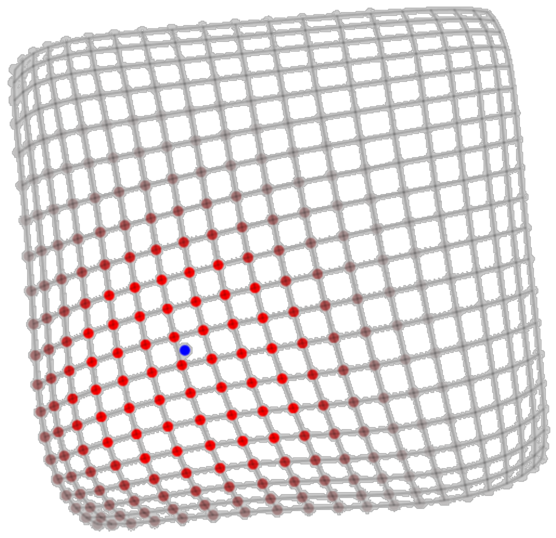


Figure 30 - SOM near the end of the ordering phase of training (*Adapted from* **[27]**).

As seen in Figure 30 the ordering phase trains the general shape of the input space, which in this case is the entire space shown in the figure. Because the input space is the entire display, the map will eventually occupy most of the space in the figure and display a lattice of nodes. The blue dot in Figure 30 corresponds to an input vector, and the red area corresponds to the neighborhood influence around the winning node (center of the red area and closest node to the blue dot). This neighborhood area is decreasing over time, but began with the entire neighborhood being influenced.

Following the ordering training is convergence training. Convergence training proceeds from the initial organization attained in the ordering phase and tweaks the map to best fit the input space. Because this phase is focused on modifying sections of the map rather than organizing the map as a whole, its training parameters are set to vastly different values and ranges. The initial neighborhood is two neurons wide and it decreases over time according to Equation 6. Additionally, the initial learning rate is set to approximately 0.01 [16] and decreases over time according to Equation 5. Finally, the convergence phase trains for approximately 20 times the number of iterations in the ordering phase [16] because the modifications and manipulations resulting from each training iteration are minimal to the map. An example of a map at the completion of the convergence phase can be seen in Figure 31.

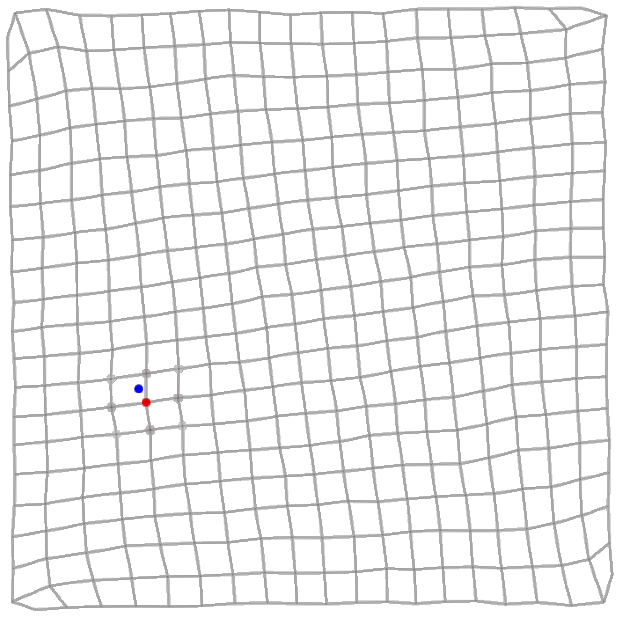


Figure 31 - An SOM at the end of convergence training (*Adapted from* **[27]**).

Figure 31 is the resulting trained map at the completion of ordering and convergence training. The blue dot is again the input vector and the red dot being the winning node with a neighborhood of its immediate neighbors. This portion of training began with a map that had a general fit of the input or design space and finished with a map that fits the design space completely. At this point in the training, the convergence phase is making the final tweaks to the map.

At this point, the SOM is trained in the n-dimensional design space, but provides no means for visualizing this space. The next step is to transition from the high dimensional design space to the two dimensional display, SOM projection in Figure 32.

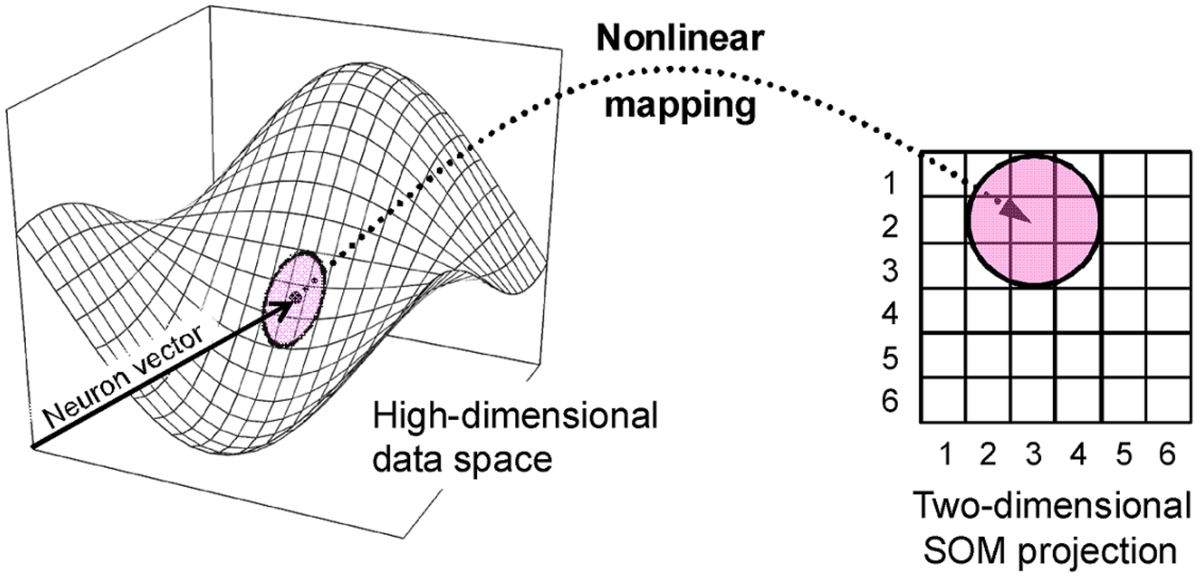


Figure 32 - Dimensionality reduction visual (*Adapted from* **[28]**).

The trained map occupies the design space denoted by the left side of Figure 32, but is represented in the two dimensional space on the right. This dimensional reduction is possible because the non-linear mapping property of self-organizing maps. Each node in the two dimensional lattice represents a region of the n-dimensional design space. Therefore, the two dimensional display of the SOM is a collage of many regions of the design space, providing a visualization of the entire design space. With this mapping, it is possible to display a high dimensional design space in a two dimensional (visible) space.

3.2.3 Contextual Labeling Process

After the map has been trained, the last phase is labeling the map with contextual labels. These contextual labels provide the information and feedback to the user or designer that describes the design space or input space. The labels are applied by executing one final ‘pseudo’ training loop, where each input vector is inputted to the self-organizing map one final time. The resulting winning node for each input vector is then tagged with a contextual label from the input vector for later processing.

After the contextual labels are applied to the nodes, it is possible to calculate the average, standard deviation, and minimum value for the groupings of contextual labels. Generally, each node will have a set of contextual labels if the data set is large enough to accommodate the 35 to one ratio of inputs to nodes respectively. With all of the contextual labels, and the general node information of mean, standard deviation, and minimum the next step is the visualization of these properties, which leads to the coloring applied to the map. The mean value is vital to discovering the representative value of the node, the standard deviation explains the variability within the node and a low standard deviation is ideal, and the minimum value helps reiterate the proximity to the optimum value in the design space.

3.2.4 Coloring Process

Utilizing the Hue Saturation and Value coloring scheme is essential in order to convey all of the details that are contained within a node’s contextual information. This research uses the HSV coloring scheme in the following way: the hue is determined by the mean of the node’s contextual information, the saturation is determined by the minimum value of the node’s contextual information, and the value is determined by the standard deviation of the node’s contextual information. The goal of using this color method is to have the best, or optimal, nodes stand out from the rest of the nodes on the map. This can be accomplished by coloring nodes with have low means, low minimum values, and low standard deviations brightly on the map. In order to make this happen a low mean value is set to green, a low minimum value is set to full saturation (full color), and a low standard deviation is set to full brightness (full color). Therefore, when a node has a high minimum value it will appear white colored, and when a node has a high standard deviation it will have a dark color.

To explain this in further detail, the first parameter is the hue. The hue of the node is set by the mean value of the contextual information contained within the node. In this case, the contextual information can be averaged, and then a color can be assigned to the node based upon how its mean contextual value compares to the rest of the map. This mean value is the most important of the three characteristics, which is why it controls the component that results in largest color change.

The minimum value is also important to a node, because it can allow the designer or user to understand how the node’s contextual labels compare to other nodes with a similar mean. Using the minimum context value of a node to determine the saturation color component of a node quickly differentiates two nodes with similar average values but different minimum values.

The last color component, value, is set using the standard deviation of the contextual labels in a node. The standard deviation of the node is important because it can explain how well the node is trained, or how consistent values are in a portion of the map. A node with a high standard deviation can be quickly identified by visually noticing the darker color of the node. A high standard of deviation does not always designate a useless area, but it can signify an optimal area and therefore is important that it can easily be identified. These properties can all be identified in the SOM displayed in Figure 33.

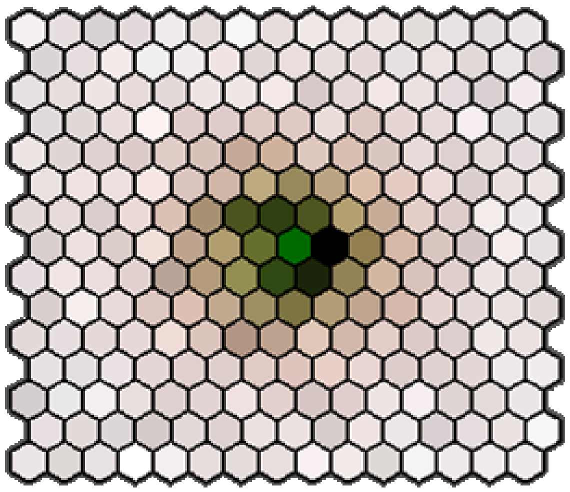


Figure 33 – An SOM displaying the coloring process using the hue saturation value coloring scheme.

In Figure 33, there is a significant color variation in the map, but generally there are two sections: the majority of the map is a white/pink mixture and the center of the map is a dark green color. The large coverage of the white/pink area is representative of a higher mean value because the hue is more red then green. Additionally, because the nodes have lower saturation value, or less pigment in each color, this area has higher minimum values. The center of the map contains green colored nodes, which contain lower mean values. The very center node is a 'brighter' green than its surrounding nodes, meaning it most likely has a better standard deviation than its surroundings. The nodes surrounding the center have greater standard deviations, represented by darker colors.

The HSV color display method effectively displays the characteristics of each node: mean, standard deviation, and minimum value in a colorful fashion that allows the user to quickly discern the difference between various nodes in the map. From this, an intuitive map is created, minimum values are nodes with bright green color, maximum values are nodes that are colored white, and high slope areas have dark colors. For example, if a user wants to find an optimal, minimum, area on the map he or she would look for the brightest green node.

3.3 Progress of SOM Contextual Maps

3.3.1 Generation of Trained Map

The first task was to understand the method of choice, self-organizing maps. After gaining a thorough understanding, training an SOM is well documented so that one may plug data into the SOM and receive a resulting trained map. The question therein lies, what can be done with this trained map? Many past researchers have utilized the U-Matrix, or individual variable maps to display this trained map. This leads to the next section, where the process of selecting a method to display the information gained through training of the map begins.

3.3.2 Research into Displaying Valuable Map Information

The U-Matrix provides distinct advantages if clustering is the goal of utilizing a self-organizing map, because it is able to clearly display node groupings and distance between clusters of nodes. Unfortunately, this research was focused more on the specific and relative values across the input space. With this goal in mind, the U-Matrix did not prove to be a solution to the visualization problem.

The next path this research took delved into the display of individual variables from the data set. Since this problem is an optimization problem, the design variables are crucial to the success of the optimization or design; however, the overarching goal will always be the objective function value or performance characteristic value. In order to extract a single variable objective function value from each node for display purposes, it would require that the SOM be trained using the objective function values of the training data as a design variable. The problem with this idea is that having the objective function value incorporated into the training of the map can skew the organization result. This result provides a trained map that clusters around objective function values, which seems to be appropriate, but also provides a ‘false’ grouping due to the map’s awareness of the objective function value. When the map is trained without the objective function value, any clusters that it makes are developed because of the similarity of the design variables rather than the objective function value. For this reason, single variable extraction visualization was not the chosen method to display the resulting map and its accompanying objective function value.

Su et al. [21] describes another method of displaying the characteristics of the trained map based upon the objective function value. This is achieved by calculating the resulting objective function value from the node weight vectors of the trained map. Unfortunately training and then visualizing in this manner required that there be an objective function present. One goal of this research was to create a method that could be utilized on data sets that did not have an explicit objective function. One example of data that would not contain an explicit objective function is data obtained through computer simulated analysis, such as finite element analysis.

The final method examined was contextual maps as a post processing event to the self-organizing map training. The contextual map allowed for a data set to be trained from solely its design variables, but subsequently display the characteristics of its objective function value. This method uses the objective function values as contextual ‘labels’ to the map. These labels are then analyzed to gain an understanding of each node in the map, from the perspective of the objective function values that it contains. With an understanding of each individual node, the map can be displayed so it is possible to see the relationship that each node has with its adjacent nodes. The contextual map is an efficient method for gaining an objective function value for the nodes in the lattice without the consequences of the previously described methods.

The last step in the process of displaying the node’s contextual information was to devise a method for coloring the node. Because each node has an array of objective function values, it would not be a sufficient solution to simply display the mean of the objective function values. Additionally, displaying the minimum objective value at each node is also not a fair comparison between nodes. These methods are insufficient because of the potential for high variability of objective function value within each node. Therefore, it is important to consider the variance or standard deviation of the objective function values within each node in combination with a display of the objective function value.

At this point, the hue saturation value color scheme became part of the development. This color scheme allowed for a three dimensional color manipulation that provided more than a color interpolation. The benefit of using HSV, described in Section 2.3, is that a color can be displayed through color interpolation but then it is also possible to alter that color to convey further information. The goal of using the HSV coloring method was to allow the user to quickly spot the important nodes and structures within the data set such as minima, optima, high curvature areas, and low curvature areas. These sections of the map are identifiable through the variation from the standard color interpolation used in other research methods.

It is important to note that not every map will be trained on a sufficiently large data set to acquire a group of contextual information for every node. For this reason, the SOM has the capability of displaying a standard color interpolation between red and green when nodes have only one contextual label. This process follows the guidelines provided in the standard HSV coloring scheme, but considers that the standard deviation is zero so the brightness of each node will be fully brightened. Additionally, the minimum value for each node without contextual information will be the same as the mean value, so the minimum value component, or the saturation component, is ignored and set to the maximum saturation.

When a node has no contextual information it is not displayed on the map in order to minimize confusion with other nodes that do contain contextual information. While all nodes may not be displayed on the map, they are each trained to accept certain inputs, so the location of the nodes is still important to the comprehension of the map. An example of a map displaying both circumstances of nodes with single contextual values and nodes without contextual values can be seen in Figure 34.

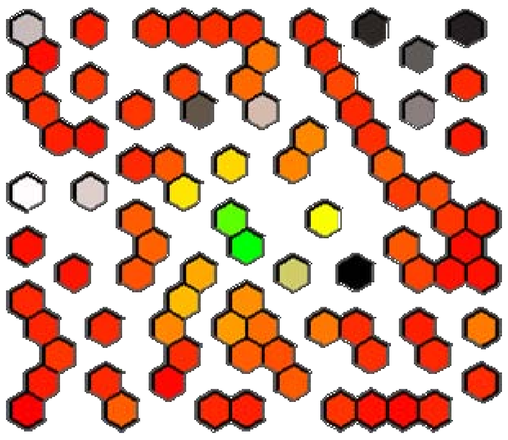


Figure 34 - A trained SOM that contains nodes without contextual information as well as nodes with only one contextual label.

Most of the nodes in Figure 34 are bright colors because they only contain one contextual label. This is a result of having fewer data points than map nodes. With this coloring scheme it is still possible to interpret the map, furthermore it is possible to view and train a map with a limited data set.

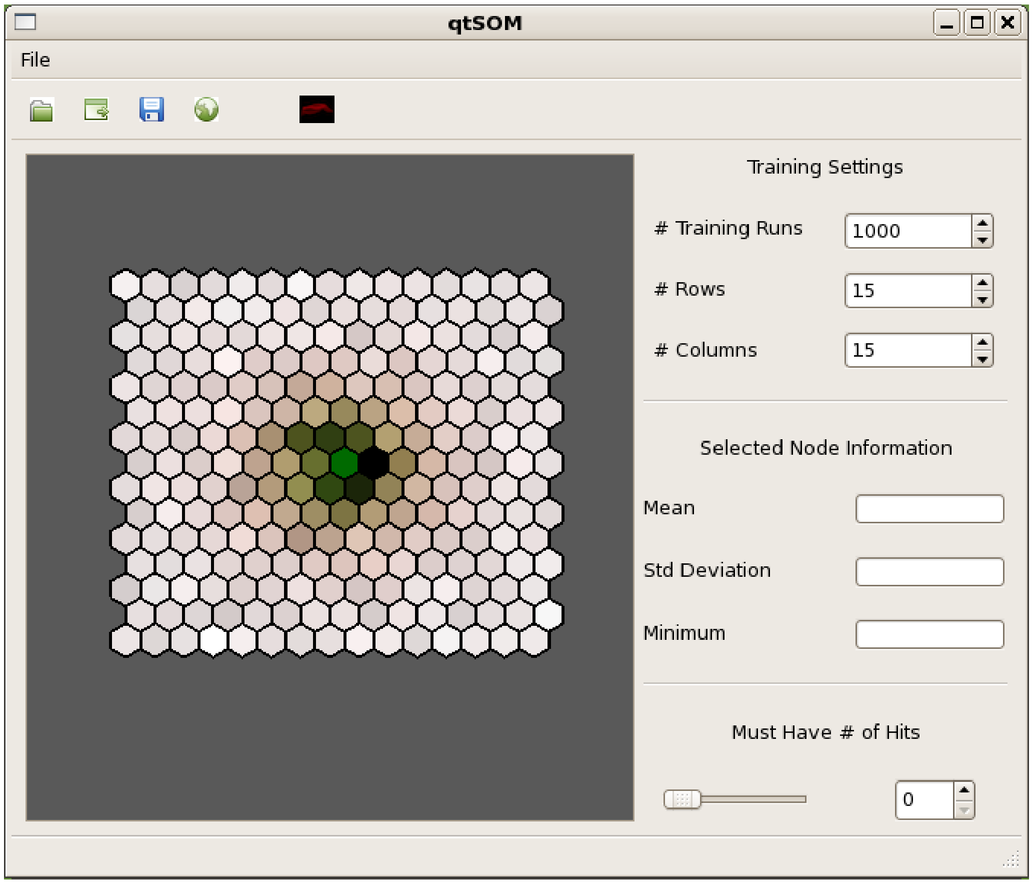
In order to accommodate all of the customization to the standard self-organizing map, it was necessary to create a desktop application. The desktop application would allow a user to train, display, and save the results of this research in ways that the Matlab toolbox is un-capable. The SOM Visualizer application, which was created to accomplish this research, is described below.

3.3.3 SOM Visualizer Application

While there are widely available tools to generate self-organizing maps such as the Matlab Neural Network Toolbox [29] and Peltarion Synapse Neural Network Software [30], this research requires a tool that could provide complete customization of the algorithm and display method. To accommodate these requirements, a C++ application was created to fit the research needs and allow for a customizable experience.

Software

This application was developed to manage data, create a map, and train the created map. In addition to training the map on a given dataset, the OpenGL graphics library [31] was used to visualize training results. OpenGL allows the program to display a variety of outputs from the map. Finally a graphical user interface (GUI) grants additional advantages so that other users can use the software without a list of instructions. This GUI was created using Qt by Nokia [32], an open source library available under the LGPL license. All of these libraries are open source and cross platform so the resulting application, Figure 35, (SOM Visualizer) runs on Windows, Mac, and Linux.



Training Runs 1000
Rows 15
Columns 15
Mean
Std Deviation
Minimum
0

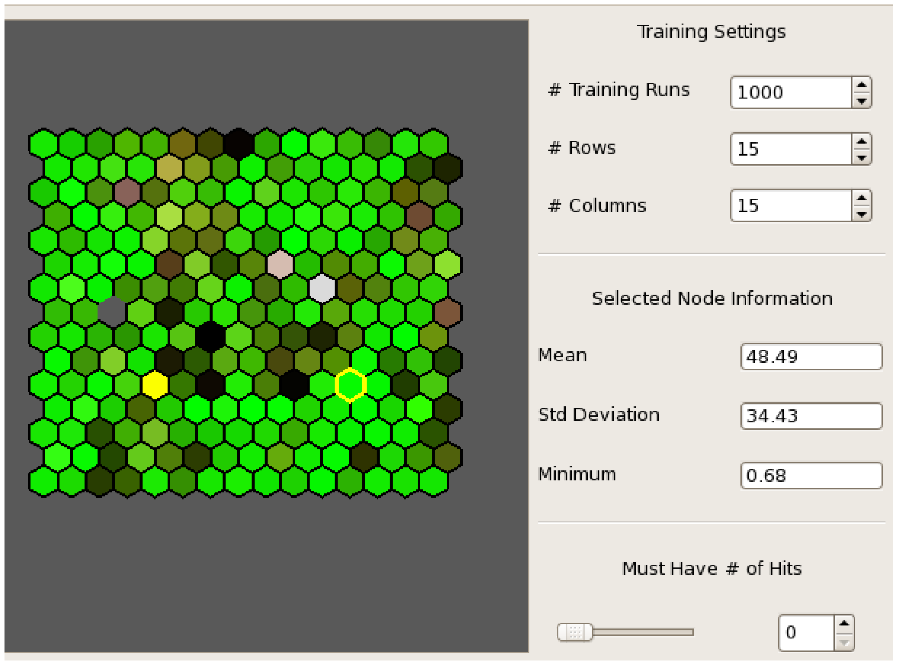
Figure 35 – SOM Visualizer application running on RedHat Linux.

Interface

The SOM Visualizer application, Figure 35, provides the user three options to customize the SOM and its training. These are the number of training runs that the map will undergo and the size of the network, specifically the number of rows and columns in the map. Beyond these three options there are many other training variables to specify, but these are less frequently adjusted and thus not put in the interface for reasons of simplicity.

In addition to the training settings, the user is able to load data, train the map, save out the trained map, and load a previously trained map. These options are chosen by clicking on designated toolbar buttons along the top of the window. The user can identify the appropriate button by reading the tool tips (messages) that display when the user hovers over a button. The workflow of these options is also displayed by only allowing subsequent options to be clicked once their predecessor has been executed. One example of this is: without loading a data set the map will not allow the user to click train. This method of sequencing commands prevents the user from attempting actions which are not yet available.

Lastly, the application allows for extensive examination of the resulting map through many output methods. The first step to map comprehension is the visual display of the map once training is completed. This map is shown in the OpenGL widget in the center of the window, and updates when there is new information to display. The user can manipulate the OpenGL display by selecting to filter the network display based upon the number of times a node ‘won’ or was activated during the contextual training. Additionally, the user has the ability to select nodes within the map for further examination. When a node is clicked by the user, it becomes highlighted and the details of this node are displayed in the “Selected Node Information” panel, an example can be seen in Figure 36.


Training Settings

Training Runs 1000

Rows 15

Columns 15

Selected Node Information

Mean 48.49

Std Deviation 34.43

Minimum 0.68

Must Have # of Hits 0

Figure 36 - A trained SOM with a node selected for investigation.

When a node is selected, the node’s mean, standard deviation, and minimum context value are displayed so that the user or designer can better understand the topology of the map. Finally, this application has the ability to save two different outputs to text files: node contextual information and the map topological information. The contextual information that gets saved to a text file displays the context values for each node as well as the node’s mean, standard deviation, and minimum. The topological information that gets saved is the entire trained map including the map size, weight vectors, and contextual labels. This map can then be loaded later for further examination.

Data

This application can load in a data set of any size; this includes both samples (rows) and variables (columns). The data is loaded into the SOM in a specific format so that the application correctly trains the map. For an example of a five dimensional data set see Table 1.

Table 1 - Input data format

Contextual Label (F)	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5
F1	X1	X2	X3	X4	X5
F2	X1	X2	X3	X4	X5
F3	X1	X2	X3	X4	X5

The contextual labels, F, column in Table 1 are the location of the objective values, which will be omitted from the training of the map and applied as contextual labels. The adjacent columns (X1, X2,.., X5) are the independent design variables used to generate the objective value, F. These design variables are the basis of the training of the SOM, and the dimensionality of the design variables determines the dimensionality of the self organizing map's node weight vectors.

4 Results and Discussion

4.1 Optimization Test Suite

For the evaluation of this method, three published optimization problems were chosen: Dixon and Price [33], Rosenbrock's Valley [34], and Ackley's Path Function[34]. These functions were chosen for their topological characteristics. These functions were also chosen because they have the ability to scale to as many independent variables as desired, providing viewable three dimensional plots as well as higher dimensional capability for the SOM to be trained upon. All of these optimization problems can be written in the standard optimization problem statement that was introduced in Section 1.3.

Dixon and Price

This function provides a topology that contains a very shallow optimal area, making convergence difficult for a formal solution method. Additionally, the edges of the design space have steep gradients causing most points in the design space to reside relatively close to the optimal value. The equation and bounds for this function are given in Equation 7.

$$\begin{aligned} &F(X) = (X_0 - 1.0)^2 + \sum_{i=2}^{n} (i) \times \left(2 \times X_i^{\,2} - X_{i-1}\right)^2 \\ &-10.0 \le X_i \le 10.0, \quad i = 0 : n \end{aligned} \tag{8}$$

The optimal value for this function resides at $F_{\min}(x) = 0.0$ and $x_i = 0.0$. Matlab was used to generate a three dimensional plot of this function by generating a two design variable version of the function and using Matlab's surface plotting to plot the two design variables and the objective value in three dimensional space. The result is displayed in Figure 37.

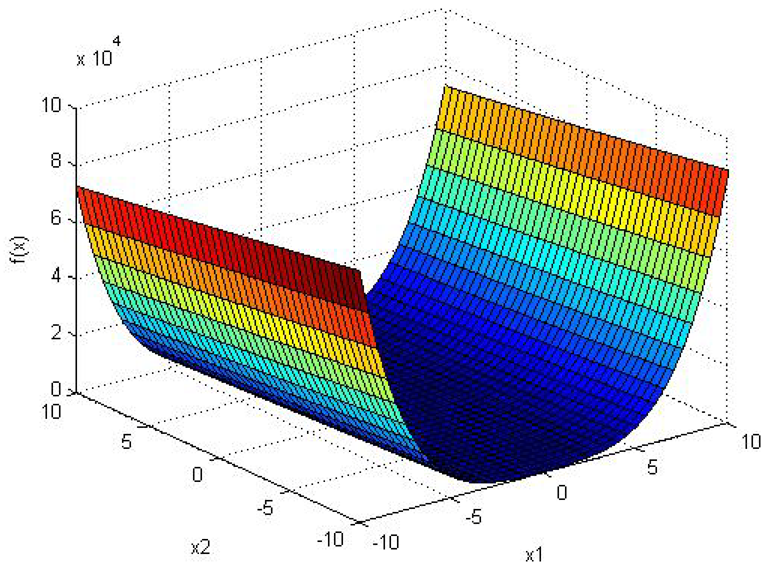


Figure 37 - The Dixon and Price Function in two dimensions.

Figure 37 displays the characteristics of this function, and provides a unique topology for the self-organizing map to visualize. This function is also greatly different than the subsequent functions, Rosenbrock's Valley and Ackley's Path function because it has a much simpler design space.

Rosenbrock’s Valley

This function is similar to the Dixon and Price function because it is uni-modal, but its complexity increases because the optimum lies within a long, narrow, parabolic valley. This adds additional complexity to the Dixon and Price function, while retaining the unimodal aspect. Equation 8 provides the equation and bounds of Rosenbrock’s Valley Function.

$$F(x) = \sum_{i=1}^{n} 100 \bullet (x_{i+1} - x_i^2) + (1 - x_i)^2$$
$$-2.048 \le x_i \le 2.048, \quad i = 0 : n \qquad (9)$$

The optimal value for Equation 8 is $F_{\min}(x) = 0.0, \quad x_i = 1, \; i = 1 : n$. Again, Matlab was used to generate a three dimensional plot of this function by creating a two design variable function and plotting the objective function value as the third dimension. This plot can be seen in Figure 38.

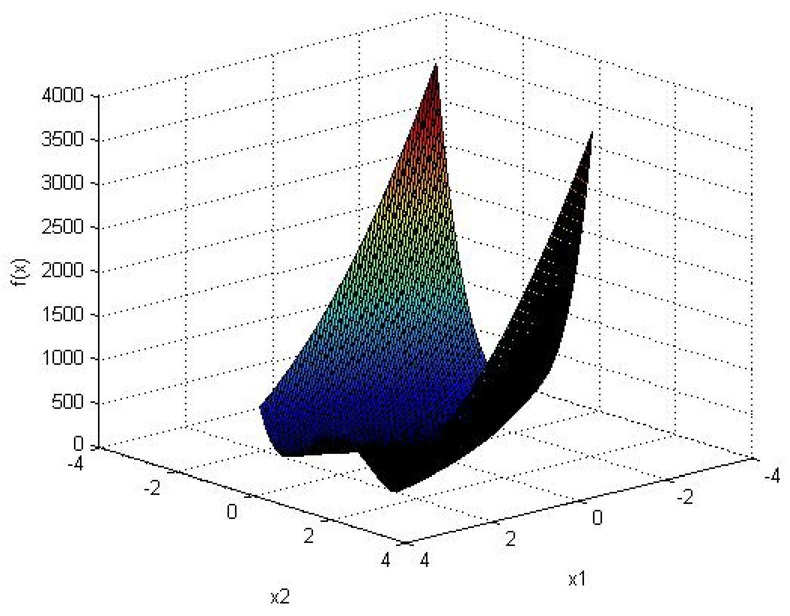


Figure 38 - Rosenbrock's Valley (Banana) Function.

As described, this function has a long narrow parabolic shaped optimal area surrounded by large slopes to the edges of the design space. Although the function is unimodal, there are areas that are relatively flat and often trap solution algorithms before they can reach the true optimum value.

Ackley's Path Function

This function varies quite a bit from the previous functions, primarily because it is a multi-modal function. This function has a generally flat looking topography with small peaks and valleys until immediately around the global optimal area. The multi-modal nature of the function provides additional complexity due to the multiple solutions present. Equation 9 was used to generate this function, which includes its objective function and bounds.

$$
\begin{aligned}
&f(x) = \frac{\pi}{n}\left\{ \begin{array}{l} k\sin^2(\pi y_1) + (y_n - a)^2 + \\ \sum_{i=1}^{n-1}\left[(y_i - a)^2\left(1 + k\sin^2(\pi y_{i+1})\right)\right] \end{array} \right\} \\
&y_i = 1 + 0.25(x_i - 1) \\
&-10 \le x_i \le 10,\ i = 1:n
\end{aligned}
\qquad \textbf{(10)}
$$

The optimal value for Equation 9 is $F_{\min}(x) = 0.0,\ \ x_i = 0.0$. A plot of the function is shown in Figure 39, which was generated similarly to the previous functions.

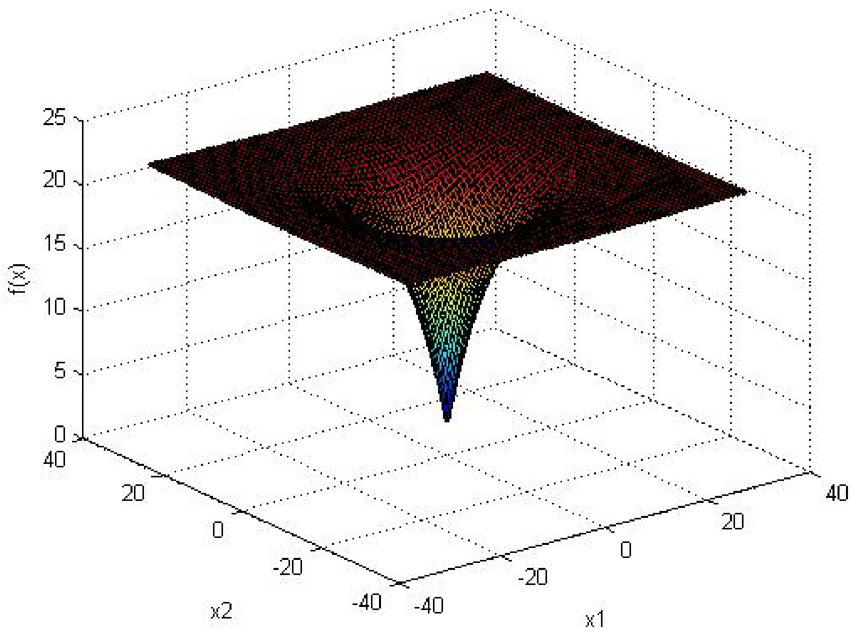


Figure 39 - Ackley's Path Function.

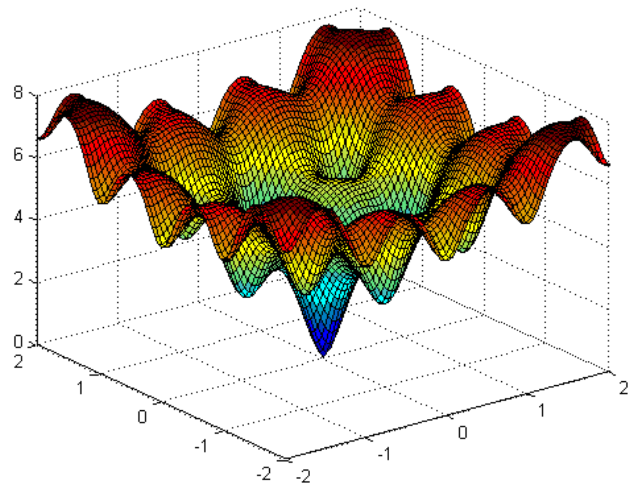


Figure 40 - Ackley's Path Function focused on the global optimum.

Figure 39 displays the characteristics described such as its multi-modal nature, generally flat nature across the map until the region surrounding the optimum is reached. This topology provides a unique function for the self-organizing map to visualize and then extract relationships.

4.2 Contextual SOM Results

With such a diverse and distinct test suite, it is expected that the features outlined in Section 4.1 are prominent in the contextual map results. In order to adequately test the functionality of this method, results were generated for a variety of test cases within each test problem. Each test problem was run under the conditions listed in Table 2.

Table 2 - Contextual SOM test cases

	100 Sample Points	1,000 Sample Points	10,000 Sample Points
2 Design Variables	15x15, 1000 iterations	15x15, 1000 iterations	15x15, 1000 iterations
5 Design Variables	15x15, 1000 iterations	15x15, 1000 iterations	15x15, 1000 iterations
10 Design Variables	15x15, 1000 iterations	15x15, 1000 iterations	15x15, 1000 iterations
	15x15 describes the map size, 15 rows and 15 columns		
	1000 iterations specifies the number of ordering phase iterations		

When examining each figure the training of the SOM algorithm, which initializes the map randomly, must be considered. The result of this random initialization is a map that can vary every time it is trained, even on the same data set. The maps displayed in the following section are a representation of what may result from training the SOM. By manipulating the training parameters such as map size or training iterations the designer has the potential to reach slightly different results.

Dixon and Price

Beginning with the Dixon and Price objective function, the first step is to verify the problem characteristics of the two design variable (three dimensional) plot, Figure 37, with the contextual map results. The resulting contextual SOMs for the two design variable case shown with 100, 1,000, and 10,000 sample points can be seen in Figure 41.

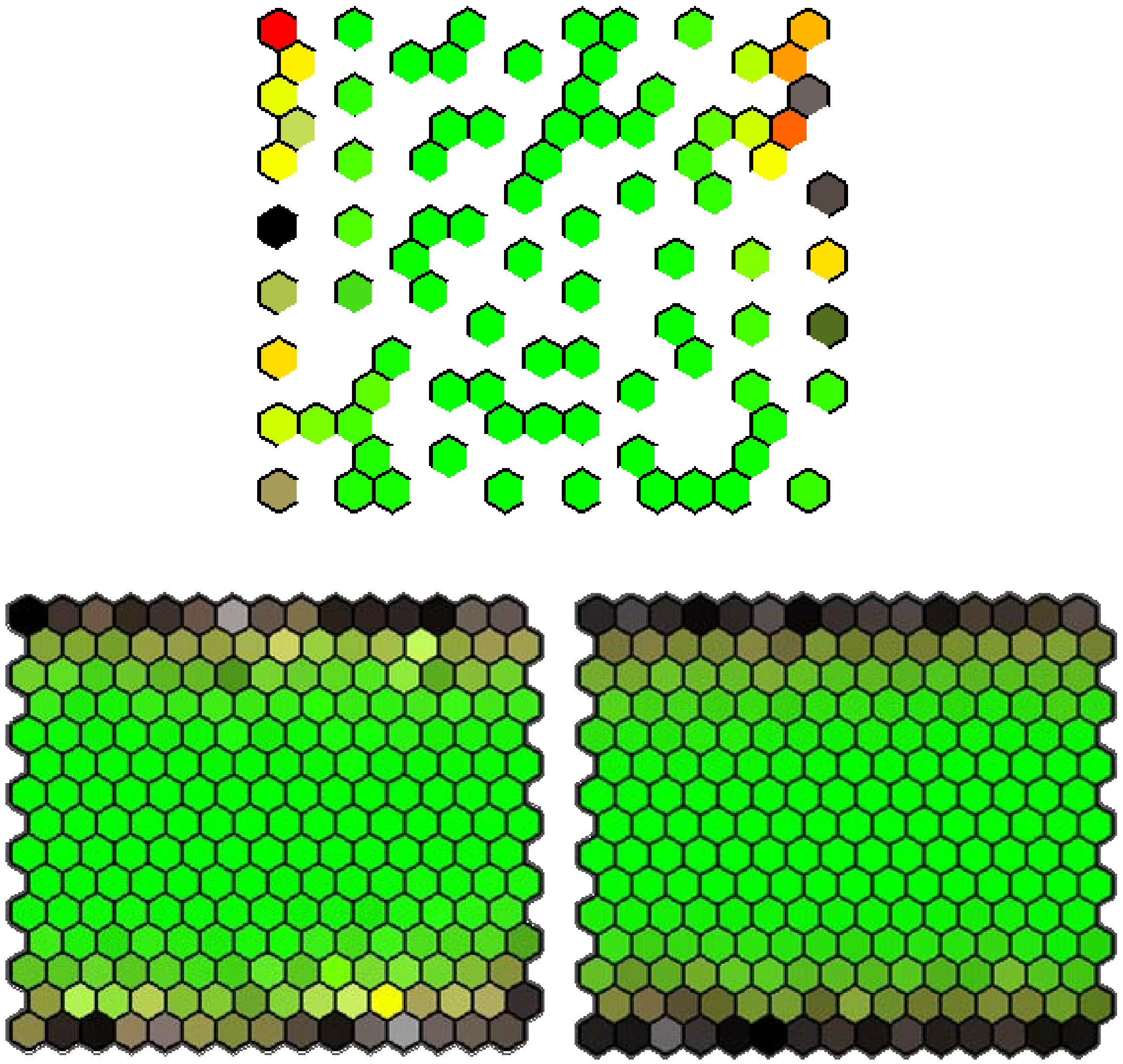


Figure 41 - Dixon and Price two design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

The results displayed in the case of 100 sample points used to train the SOM show features consistent with the three dimensional Matlab plot for Dixon and Price, Figure 37. Specifically, this function displays a large optimal area and steep curvature at the boundaries of the design space. This map primarily utilizes only the color interpolation rather than the full HSV method because there are more nodes than data points, so some nodes have one label and other nodes have no labels. The 1,000 and 10,000-point maps displayed in Figure 41 contain the characteristics of the three dimensional Matlab plot, Figure 37. Both the 1,000 and 10,000-point maps contain a shallow optimal area (constant green area) and steep curvature on the edges of the map (dark colored areas). The trained map provides an adequate comprehension of the design space by conveying its uni-modal nature, shallow optimal area, and high curvature on the edges of the design space. The entire middle green area would most likely be a good initial search region for a formal solution algorithm to start from as the points therein are low objective function values.

The next case in the training regime is the five design variable case of the Dixon and Price function. Therefore, the trained map should display similar characteristics to that of the two dimensional case. The results of training the 100, 1,000 and 10,000 cases are displayed in Figure 42.

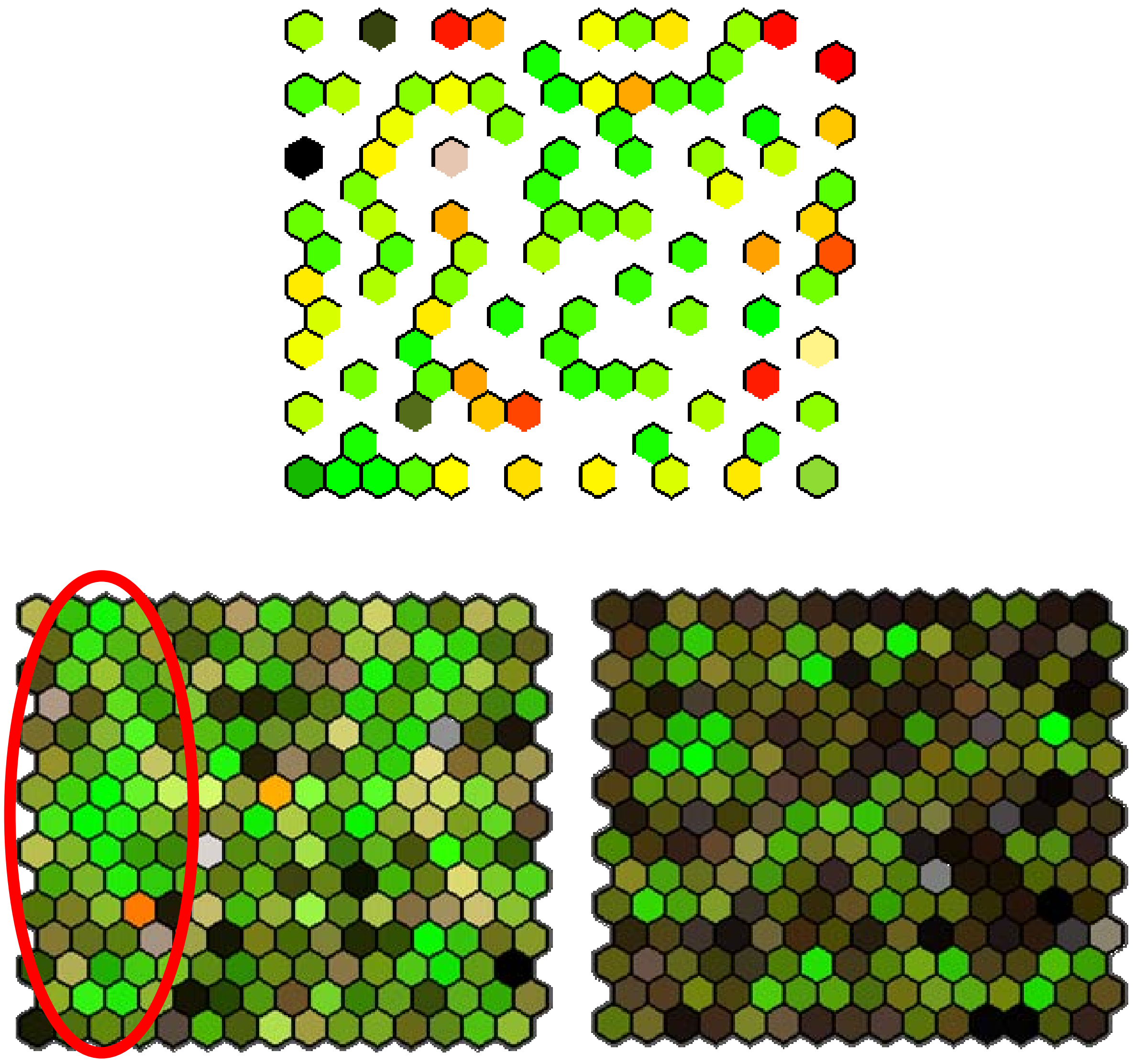


Figure 42 - Dixon and Price five design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

The mapping of the 100-sample point data set provides more conclusive results than the 1000 and 10000-point data sets. This figure displays a low central area, consistent with the three dimensional plot of Dixon and Price. This is apparent by the many green nodes contained within the center of the map, and more variable nodes to the edges of the map. A more conclusive map may have resulted from the smaller data set because the training parameters such as map size, the learning rate, and number of training iterations allowed for this data set to be more accurately trained. Since the training parameters for all the maps are kept constant, some maps may not be training sufficiently. The results from Figure 42 are less conclusive than the two design variable case. The 1000-sample point map (left) appears to have a slightly brighter central strip (circled) through the map which would equate to a lower standard deviation. This feature is not extremely evident, but could signify the shallow area in the center of the map with the edges of the map displaying a higher standard deviation and therefore a greater curvature. The 10000-sample point case (right) does not display any significant trend at all. However, this map does contain areas highlighting the optimal area. Unfortunately, there are several groupings that lead one to believe that the problem is multi-modal when in fact it is uni-modal. Even without an evident trend, the larger optimal area in the 10000-sample point plot could be a beneficial initial starting location for an optimization algorithm.

The last test case for the Dixon and Price function is the ten design variable data set. Again, the ten design variable case has sets of 100, 1,000, and 10,000 samples.

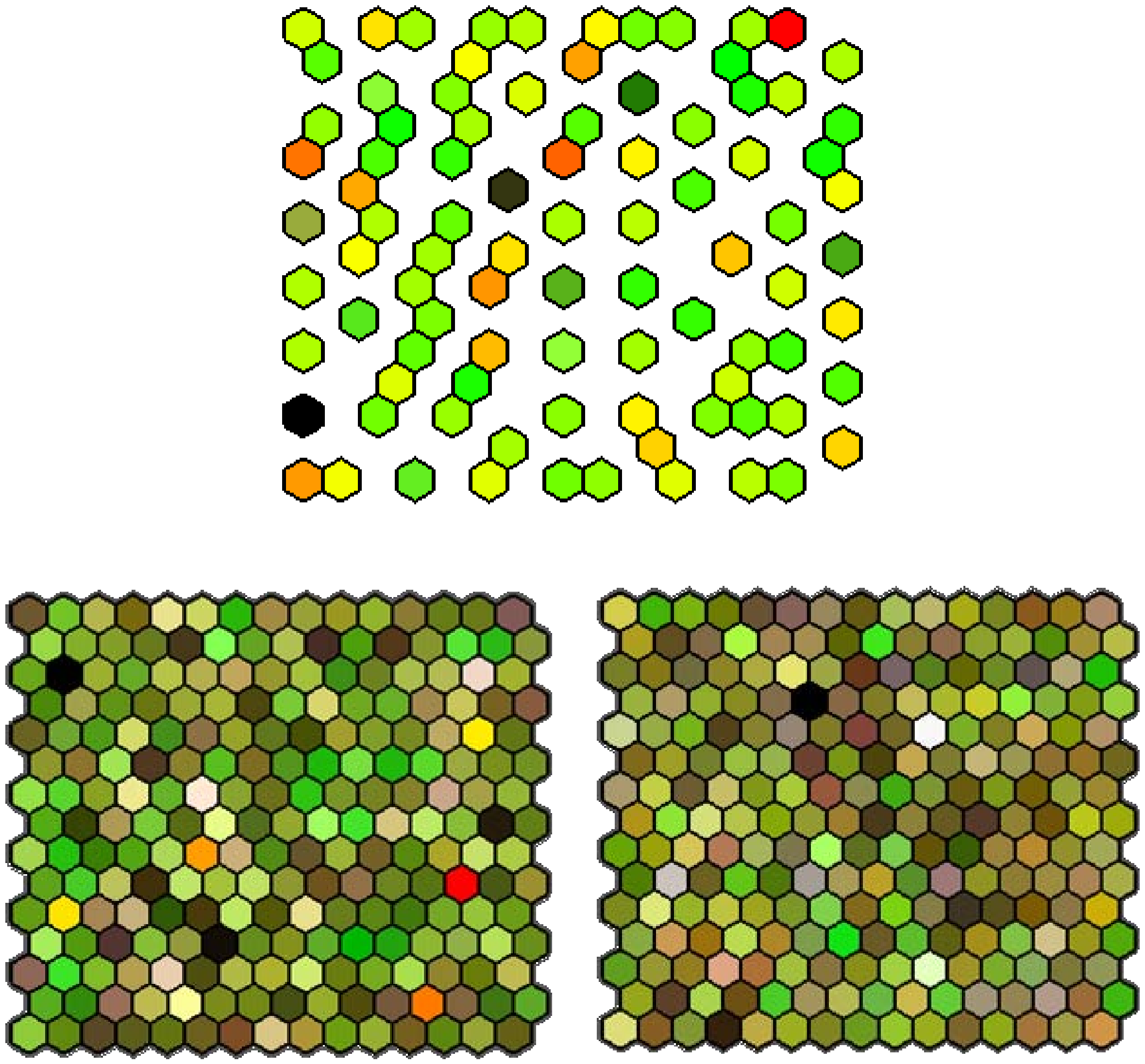


Figure 43 - Dixon and Price ten design variable SOM trained using 1,000 (left) and 10,000 (right) sample points.

The 100 point map in Figure 43, similarly to Figure 42, displays no significant correlation between the trained map and the characteristics of the Dixon and Price function. This is not surprising with the result of the other ten design variable trained maps, Figure 43. The trained maps in Figure 43 contain similar features to the trained maps of the five design variable maps, Figure 42. Unfortunately neither Figure 42 nor Figure 43 explain the characteristics of their respective design spaces. Again, the cause of these inconclusive maps may be related to their non-ideal training parameters. The resulting maps appear to have high standard deviations across the map and therefore the SOM nodes may not be grouping the data appropriately. Possible improvements for this function could come from a display of the individual design variable values within a node, modifications to the training parameters, and other items in the Future Work Section (5.2).

Overall, the contextual self-organizing maps trained on the two design variable Dixon and Price function provided a complete understanding of the problem characteristics. Additionally, the 100-point sample of the five design variable Dixon and Price function gave a similar understanding. Unfortunately the large five dimensional data sets (1,000 and 10,000) and the entire ten dimensional data sets resulted in inconclusive maps.

Rosenbrock’s Valley

Recalling the characteristics of Rosenbrock’s Valley function, Figure 38, the problem is uni-modal and contains a large shallow parabolic (banana) shaped optimal region. The surrounding area around the optimal has relatively low objective function values in comparison to the corners of the design space where the objective function value grows large very quickly. This section presents trained maps of the Rosenbrock’s Valley function in accordance with the training structure of Table 2.

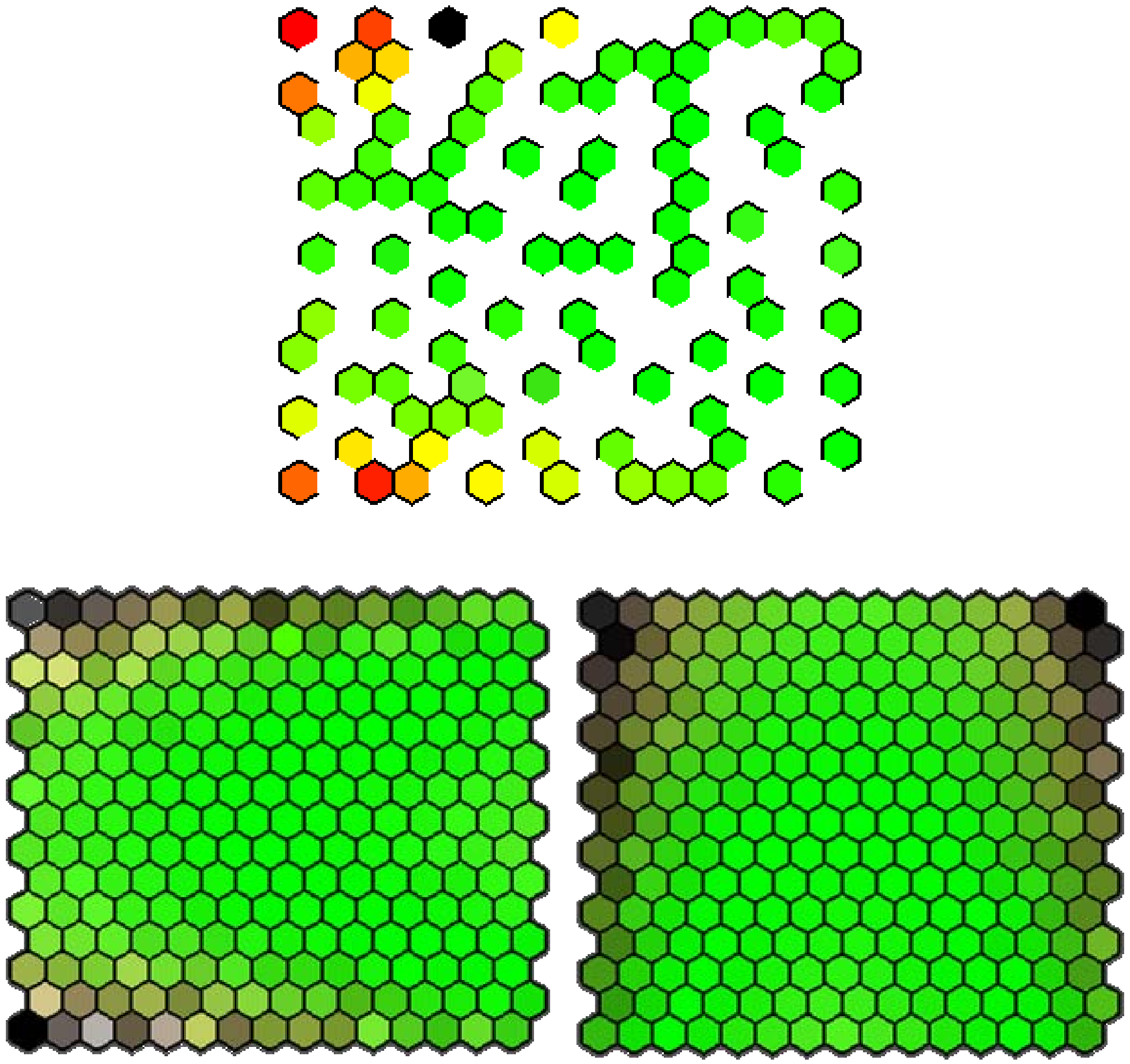


Figure 44 - Rosenbrock's Valley two design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

All of the maps in Figure 44 display an accurate representation of the design space and the characteristics described for Rosenbrock’s Valley, Figure 38. Figure 44 maps the 1,000 and 10,000 sample point data sets of the two design variable Rosenbrock’s

Valley function very well. These maps display the characteristics of the problem described in Figure 38 such as the parabolic optimal area, unimodal nature, and steep curvature at the corners of the design space. In addition, there are two things to consider upon inspecting these maps: the orientation and color variation. Recall that the map is randomly initialized so the orientation of the nodes in the map is irrelevant. Secondly, the majority of the map is colored green because the variation in mean node value from the optimal area to its surrounding nodes is much less than the difference between the optimal area and the corners of the design space.

As the dimensionality of the design space increases, the goal is to retain the ability to map the features of the design space. The next set of maps are trained on a five design variable data set with 100, 1000, and 10000-sample points, in Figure 45 respectively.

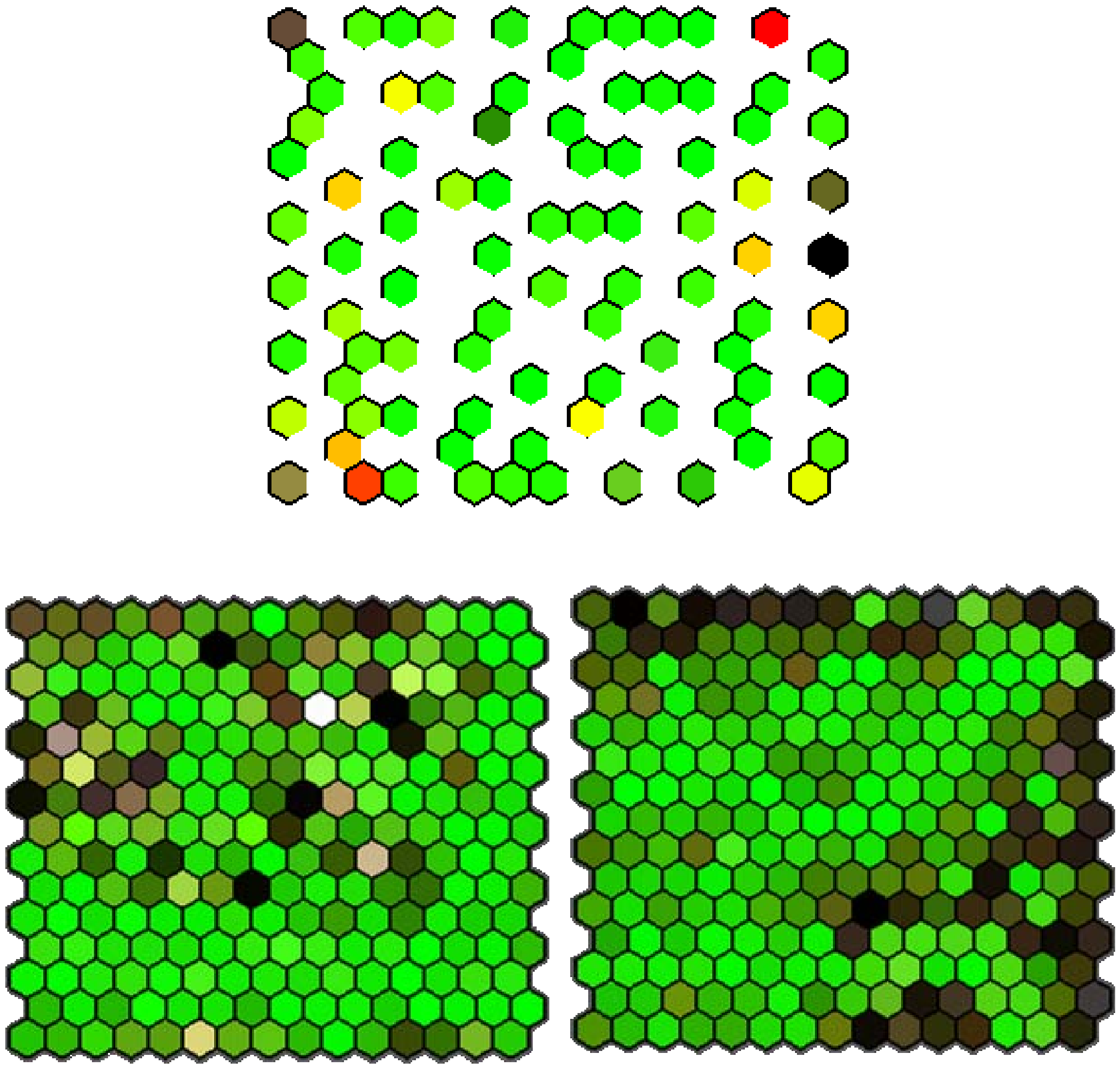


Figure 45 - Rosenbrock's Valley five design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

The top map in Figure 45 appears to contain the characteristics of a uni-modal problem with large curvature at the corners of the design space; however, the number of unused nodes in the map limits the information that can be gathered. Unfortunately, the unique shape of the optimal area cannot be extracted from the 100 point map in Figure 45 as it can from the other two maps because of the gaps in the map. Fortunately, it would be possible to identify an initial search area using this map, because of the exploration features of the application interface. Through inspection of the trained maps, Figure 45, it is noticeable that the characteristics of the design space are retained in this display. These maps retain the shallow optimal area with the edges of the map containing high curvature areas displayed through the darker colored nodes. Since the majority of the optimal area (green area) is connected, it could be surmised that the problem contains uni-modal characteristics too. Finally, these maps clearly display bright green nodes denoting close-to-optimal contextual values which could be useful for an initial search region of an optimization algorithm. This could be utilized by selecting the brightest green nodes, and saving the design variable values of the selected nodes for use as a starting location in an optimization routine.

The final map generated from the Rosenbrock’s Valley 10D function with 100 sample points. Ideally, the characteristics that have displayed through the two and five dimensional maps would also be displayed in Figure 46.

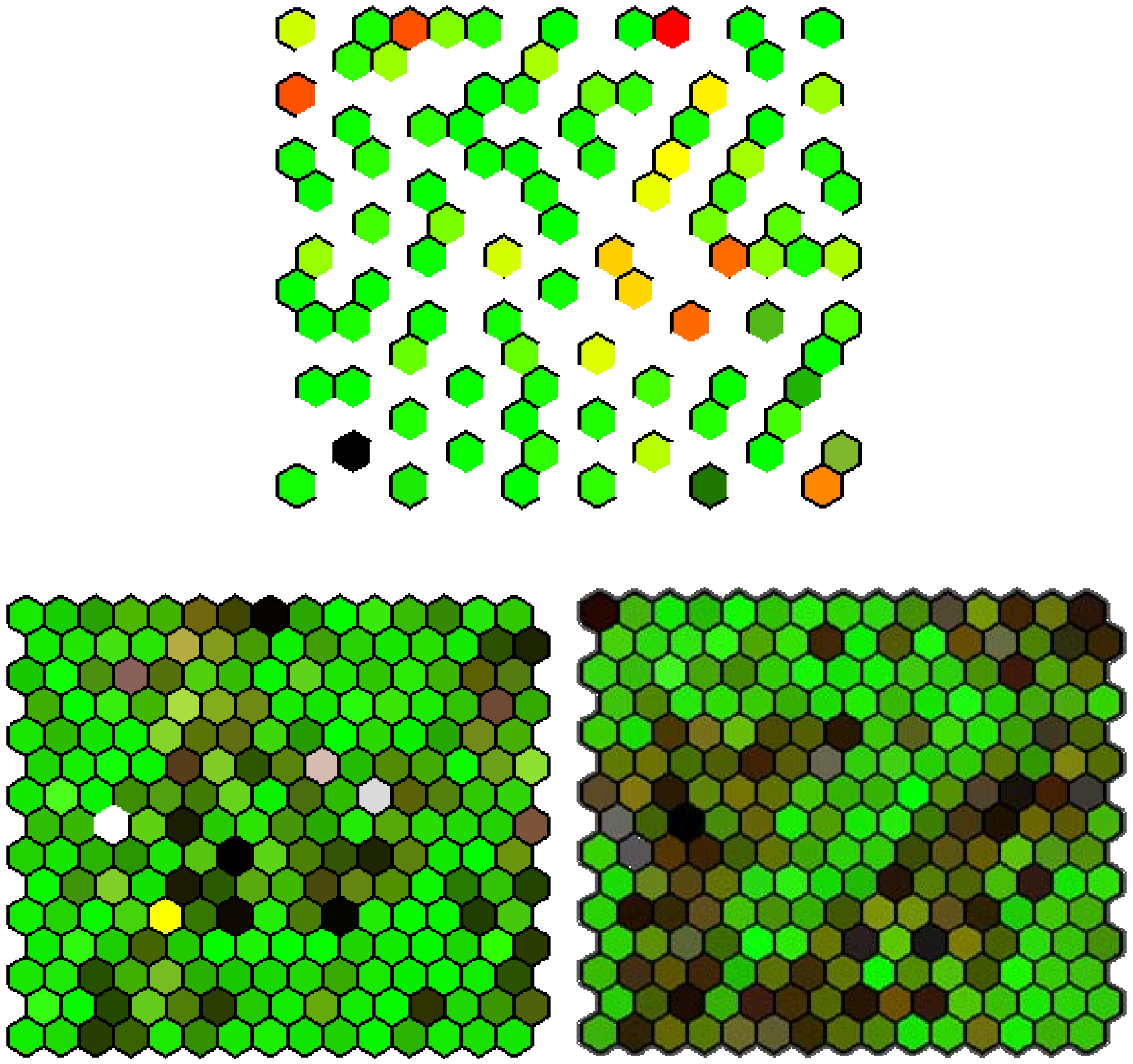


Figure 46 - Rosenbrock's Valley ten design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

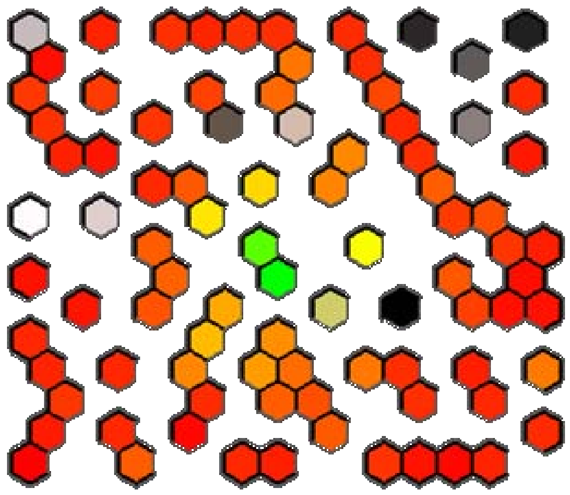
The holes within the 100 point map of Figure 46 make it difficult to interpret, but the large number of green nodes suggests a flat region of the design space around the optimum. The problem of node coverage can be solved by either decreasing the map size, or finding a method to interpolate between the nodes on the map and display colors on the now empty nodes. This will be discussed further in the Future Work

Section (5.2). The results from the 1000 and 10,000 point maps in Figure 46 show two maps that have unique optimal areas, but without the ability to picture a ten design variable space it is hard to verify their correlation back to Figure 38, the Matlab plot. It is important to consider that self-organizing maps are a dimensionality reduction method and there is no correct pattern to the output neurons. Two characteristics of the design space that are apparent in the left and right map of Figure 46 are the large optimal area and steep curvature within the design space. These two properties were present in the Matlab plot, Figure 38, and should remain properties of the design space as the dimensionality is increased. With a large number of bright green nodes, it is likely that the map did fit the input space effectively because the standard deviation of individual nodes in the map is not high.

The maps in Figure 44, Figure 45, and Figure 46 that were trained on the Rosenbrock's Valley function are promising in their ability to map the design spaces of two, five, and ten dimensional problems. The resulting maps can be utilized to discover problem characteristics such as the curvature of the problem, modality of the problem, and an initial search area for optimization algorithms. The 100 point maps at five and ten dimensions in these figures were not as effective in displaying the design space, but with some work to interpolate between node colors, these maps could become extremely informative. Interpolating between the colors would build upon the values of surrounding nodes; it would be possible to assign a color value to an empty node by calculating a value that falls between the color values of the surrounding nodes.

Ackley’s Path

The Ackley’s Path function is different than both Dixon and Price and Rosenbrock’s Valley because it is a highly multi-modal problem. The general shape of this function, Figure 39 is that of a cone with peaks and valleys covering the entirety of the cone. This section contains the resulting maps that were generated by following the training test cases in Table 2. With an understanding of the training capabilities of the contextual SOM on various data sets, it can be expected that the SOM adequately maps the two design variable data sets.



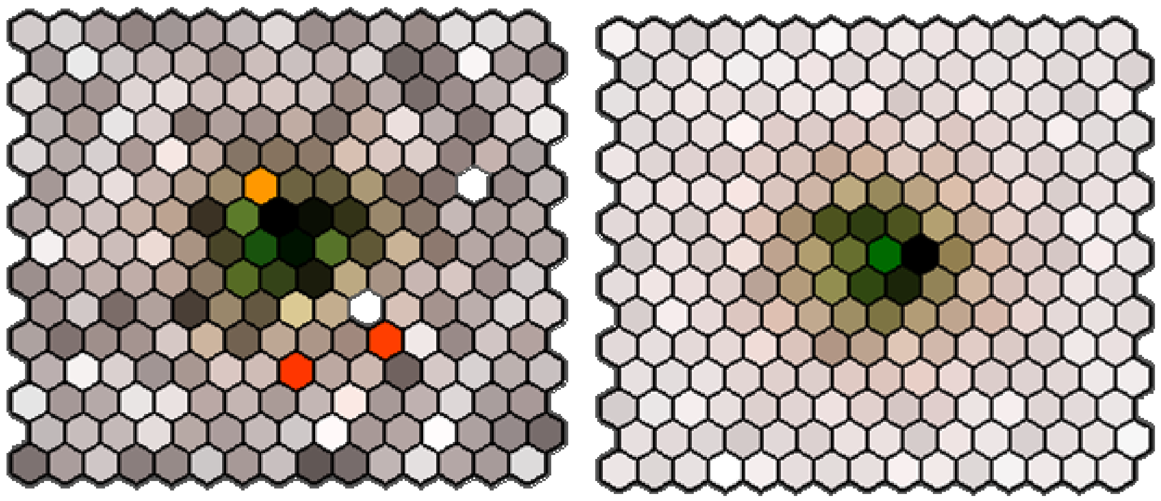
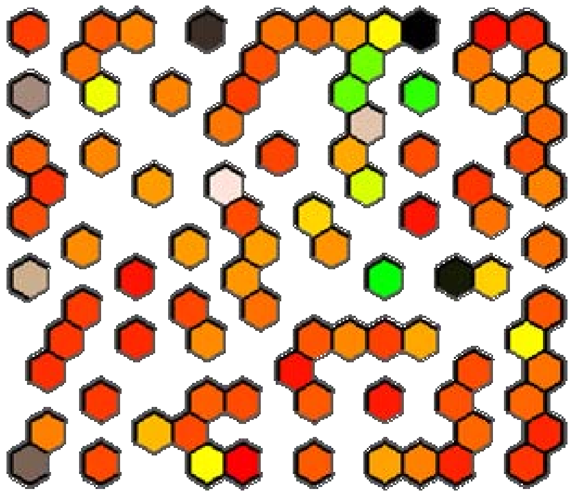


Figure 47 - Ackley's Path two design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

As expected, the 100 point map in Figure 47 displays a topology consistent with the expectations for Ackley's Path. There is a definite optimal area, which is very different from the rest of the map. There also appears to be local minima across the map, represented by the pockets of high standard deviations. Unfortunately due to the limited number of nodes in this map, it is difficult to decipher the modality of the function. Figure

47 fit the design space well and provided beneficial visual representation of the design space. As with the previous maps trained on two design variable data sets, the SOM is capable of very closely mapping a two design variable data set. The characteristics that can be extracted from these maps are that is a multi-modal problem and that it has an area of the design space close to the global optimum. The multi-modal nature is identified by the change in colors in the white/grey area of the maps, because these changes in color signify that a node will represent a set of value that is slightly higher or lower than the node next to it.

After completing the training and examination of the two design variable data set for Ackley’s Path, the next data set is a five design variable data set. This set will be represented by a 100, 1000, and 10000-sample point data set in Figure 48.



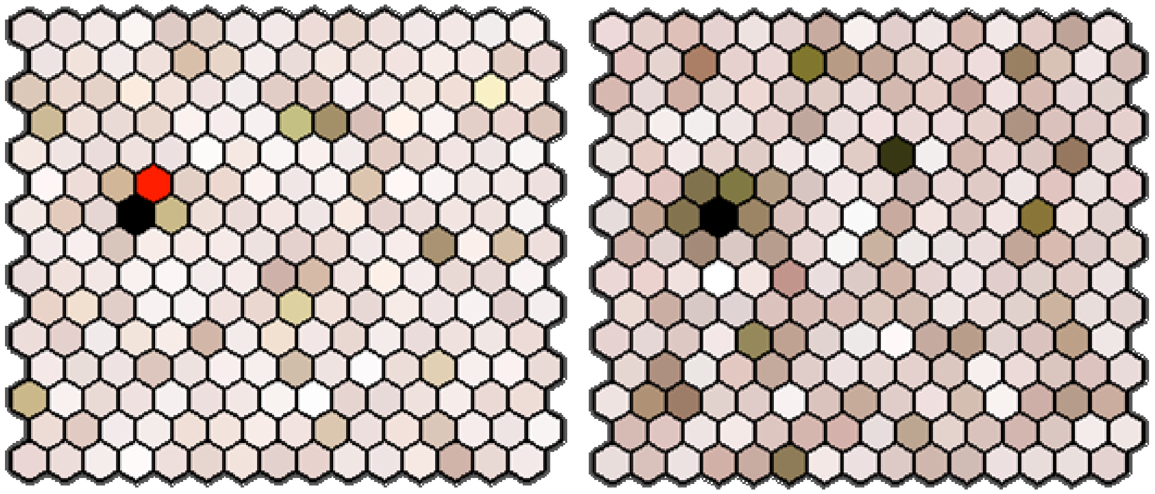


Figure 48 - Ackley's Path five design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

This map 100-point map displays the characteristics of a multi-modal design space because it contains such a scattered variety of orange and red nodes. The dispersion of orange nodes correspond to high objective function values and the red nodes are even higher objective function values. When these colors are next to each other repeatedly, this suggests peaks and valleys within a small range of objective function values, or more simply a multi-modal behavior. This map unfortunately suffers from a lack of nodes to effectively display the optimal area. However, there are three distinctly green nodes on the map, which would provide a beneficial starting location for a quicker path to optimization convergence. Figure 48 also potentially shows the global optimum region. Normally, this would be a green area on a map, but due to the nature of the problem, the global optimum area is also one where there are large gradients. Thus, the node(s) would appear black, as they contain both low objective function values and high standard deviations. This black node can be seen in both the 1000 and 10000-point maps shown in Figure 48. The maps with 1000 and 10,000 data points also suggest a multi-modal characteristic of the function through the high variability in white and pink and brown values over the majority of the map.

The last group of data sets to be examined is the ten design variable sets of data with data sets containing 100, 1,000, and 10,000 sample points. The first maps that will be presented is the 100-point data set and then the 1000 and 10000-sample point maps. The first map, 100-samples, will most likely not be full enough to utilize for proper analysis, but will display the multi-modal nature of the problem.

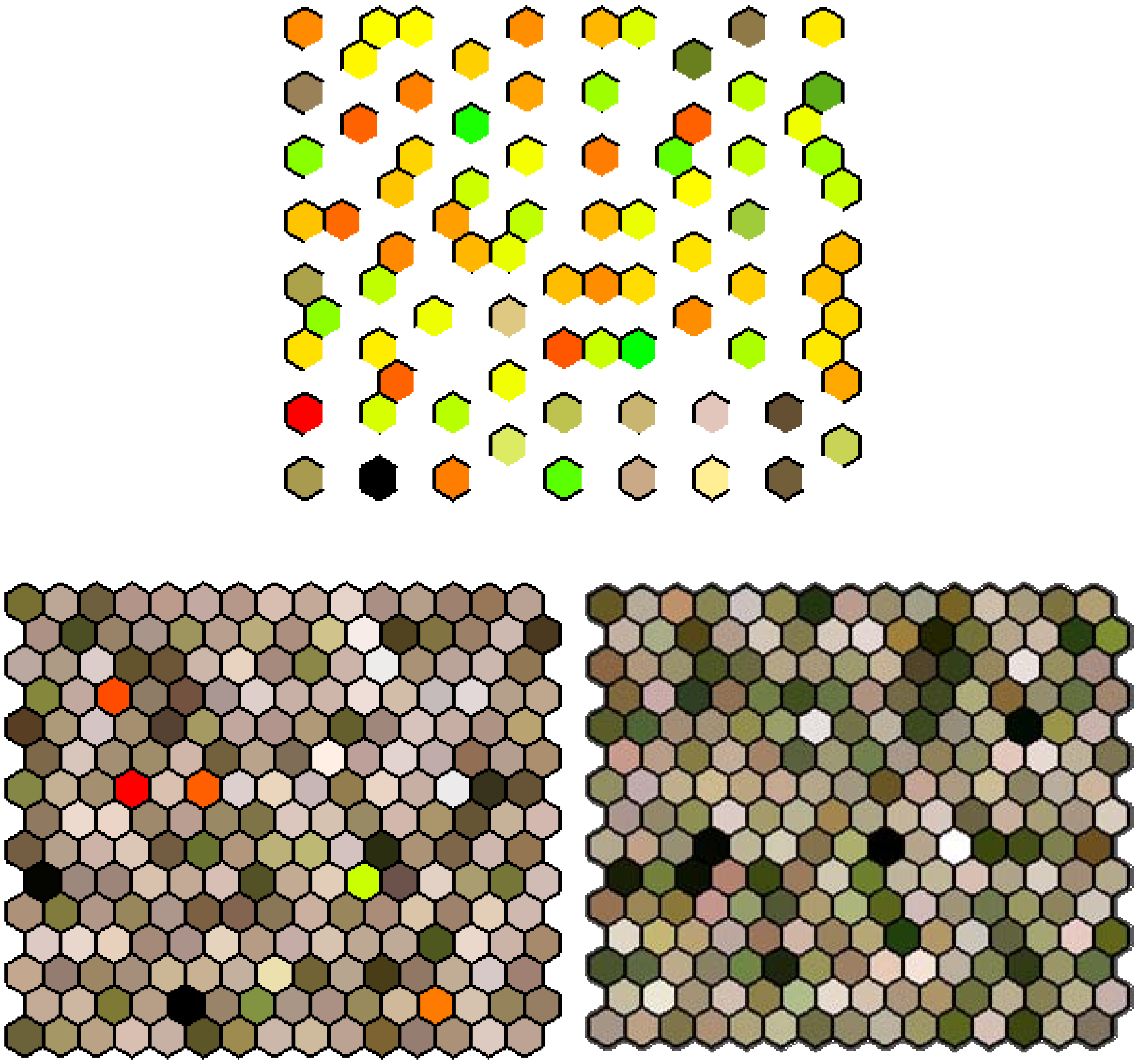


Figure 49 - Ackley's Path ten design variable SOM trained using 100 (top), 1,000 (left) and 10,000 (right) sample points.

The top map in Figure 49 contains a map that shows the multi modal nature of the Ackley’s Path function, but more than likely does not have a sufficient representation of the design space. There are several green nodes across the map, which means that the 100-sample point data set did not include a point that was located near or in the optimum region because the green nodes across the map are all in local minima. The multi-modal nature of this function is an important feature, but with a more encompassing data set it may be possible have been possible to discover an optimal area. While the resulting maps in Figure 49 may appear to be chaotic and unorganized, they provide significant information regarding Ackley's Path ten dimensional design spaces. The first observation in both maps is that the function is highly multi-modal, this is again because of the large area of the map which contains many peaks and valleys in the form of browns and greens. The second important piece of information from the 1000-sample point map on the left is again the initial search region; this map contains one bright green/yellow node which resides near the optimum. Unfortunately, the 10000-sample point map on the right does not clearly display any optimal area, but this may be due to an insufficient training length.

The maps trained from the Ackley's Path function, Figure 47 - Figure 49 all sufficiently display the multi-modal nature of the function. The maps that were unable to identify an optimal region of the map most likely suffered from the lack of an input vector near the optimum of the function. Assuring that the design space is sufficiently explored with the data set is difficult as the dimensionality increases because a function with a small optimal region, such as Ackley's Path, can decrease the chances of picking a point near the optimal region.

4.3 Discussion of Results

The trained contextual self-organizing maps that were presented in Section 4.1 provided valuable insight into the capability of contextual SOMs in design space visualization. As expected, when the dimensionality of the design space decreased, the comprehensibility of the trained map increased; alternatively, when the dimensionality of the design space increased, the comprehensibility of the trained map decreased. This paradigm is currently present in other research methods, and exemplifies the need for a capable solution to design space visualization.

Dimensionality reduction methods specialize in providing an abstract visualization of a high dimensional space; therefore, the two dimensional representation of a high dimensional problem cannot be compared to the two dimensional representation of a low dimensional problem. For example, a strict comparison between the maps in Figure 49 and the maps in Figure 47 will not make feasible sense because the two dimensional data set does not require dimensional reduction where as the ten dimensional data set requires dimensionality reduction to display the resulting design space. This property of self-organizing maps makes verification of the results difficult.

Another important consideration is that for every design variable that is added to the problem, the size of the design space increases. Due to the increasing design space, the probability of adequately covering the design space with a set number of sample points (100, 1,000, and 10,000) decreases. This is especially evident in the Ackley's

Path function, because the optimal value falls within -5 and 5 for every design variable, but the range of each design variable extends from -32 to 32. So, as the dimensionality increases, the likelihood of all the design variables falling within the -5 to 5 range decreases because of the added area to the design space. In order to compensate for this, it would be necessary to generate a design space of more points such as 100,000 or one million points. Generating this large of a data set will impact the training time of the map, but this could be enhanced by implementing a multi-core training algorithm which is described in further detail in the Future Work, Section 5.2.

While not all the results provided an easy to interpret map, the SOMs in Figure 42 and Figure 43, most of the maps did provide some information about their respective design spaces. For example, all of the maps trained on Rosenbrock's Valley, Figure 44, Figure 45, Figure 46, and provided the general shape of the optimal area as well as an indication to the location of the optimal region. Additionally, the maps trained on Ackley's Path function, Figure 47, Figure 48, and Figure 49 displayed the multi-modal characteristics of the design space. All of the maps in Figure 47-Figure 49, with the exception of the 10000-sample point map in Figure 49, provided a location of the optimal region.

This knowledge of design spaces is important for the progression of optimization because it will guide to a more efficient choices when trying to solve the problem. This solution can be reached by utilizing knowledge such as the modality of the problem, because some optimization methods specialize in highly multi-modal methods. An initial search area can also prove to be invaluable to convergence to a true optimum because it will help avoid getting caught in many local minima if exploration of the entire design space were necessary.

Lastly, as mentioned after the results from the Dixon and Price function, the training parameters that were used for these test cases could always be improved to fit the test case better. These parameters: map size, training iterations initial learning rate, and initial neighborhood size all affect the outcome of a trained map, and can each be adjusted to provide a more ideal map. With a further understanding of these parameters, it may be possible to generate more meaningful maps for all data set ranges including high dimensional data sets with a low number of samples.

5 Conclusions and Future Work

5.1 Summary and Conclusions

While the use of self-organizing maps in optimization and design space visualization is not a new concept, the application of contextual maps to display the results of a self-organizing map trained on an optimization is a novel approach. This approach has many benefits over other methods, such as the ability to function with or without an objective function, and the capability of training on a data set of any size. This method can also display the resulting information in a two dimensional plot, but with more details of the node network's contextual information such as mean, standard deviation, and minimum value. With this information, users will be able to discover design space characteristics and apply that knowledge to benefit the resulting solution.

This method provides an intuitive display that utilizes colors to convey the results of the trained self-organizing map. The colors are manipulated so that the optimal values in the map stand out by being the brightest colored nodes. The nodes that are more variable or have worse contextual labels do not attract the user's attention like the bright colored nodes, but are there to provide additional characteristics of the problem. The characteristics that can be extracted from a SOM representing a design space are the curvature of the design space, the modality of the design space, and the optimal region of the design space.

The results in this thesis displayed the ability for the contextual self-organizing maps to effectively display a high dimensional design space in the two dimensional plot, see Figure 46. The benefit of adding the contextual information to this plot is shown to convey more information about the nodes along with their node value. The knowledge of a node's standard deviation can allow the designer to discover areas in the design space with high curvature. The minimum values of nodes across the map can be used to discover the modality of the problem, for example when there are minimum values scattered around a map the problem is multi-modal. The last piece of information, the node mean objective value, is pertinent to discovering the optimal region of the map. These three characteristics are combined together in the contextual map so that the designer can easily extract characteristics of the design space.

While the results showed successful attempts at design space visualization for the Ackley's Path function and the Rosenbrock's Valley function, the Dixon and Price function did introduce problems with the method. The problems that arose from this method are most likely attributed to non-ideal training parameters. The training parameters in question are the size of the map, number of training iterations, initial learning rate, and initial neighborhood width. With a better understanding of these parameters, the Dixon and Price function might result in a more comprehendible map.

The use of this method for design space visualization and comprehension will lead to improved solutions of optimization problems. Convergence will occur more quickly by utilizing the knowledge gained of problem characteristics such as the modality, curvature, and the initial search region. The results of better optimization will lead to significant savings in time, project resources, and overall cost.

5.2 Future Work

The results of this thesis gave examples of contextual self-organizing maps providing crucial information about the design space, but all of the results were unfortunately not as comprehendible and useful. Therefore, there is a wealth of avenues to explore in order to advance this method. The future work tasks fall into four areas: the training process, results improvement, results verification, and method exploration.

Given the results for the Dixon and Price function, Section 4.2, the first step to improving the results of this work would focus on the training algorithm. The training parameters for this method were set based upon recommended values from a neural network textbook, but some of the maps, Figure 43, appeared undertrained. Modifications to the training parameters could result in improved training and mapping, which would provide better visual results. Secondly, the training duration could greatly decrease in time if the algorithm were modified to be multi-threaded. Another area for training improvement is to calculate the principle components of the data (PCA), and use these principle components to initialize the weight vectors rather than generate the initial map with random values. The batch SOM [35] can decrease training time and therefore allow for larger data sets or more training iterations.

The second area for improvement of this method is in the results of the contextual self-organizing map. While the results convey a meaningful display to the user, there are modifications that could be included to improve the resulting displays. For example, it would be possible to interpolate color values between the empty nodes if a network is not completely labeled. This is accomplished by examining the color of an empty node's neighbors and assigning it an intermediate value that falls between the surrounding colors. If this were implemented, the small data set maps with 100 sample points would become more useful. Additionally, given a two dimensional node lattice it would be possible to plot that lattice in three dimensions, similar to the Matlab plots, Figure 37, Figure 38, and Figure 39. The process for plotting this would be similar to that of the SOMO paper, by which the lattice is simply shown in three dimensions with the third dimension being the node value. Lastly, this method could expand its capabilities into multi-objective visualization. This task could be achieved through a variety of means, for example the color display could change, or the number of output maps could increase.

Another area for further work is in the realm of result verification. While it is difficult to visualize a high dimensional design space, the resulting maps could be further analyzed for problem characteristics. In order to accomplish this, the application would need to allow for multiple node selection and output of the design variable values contained in those nodes. Also, this optimal region of the map could be inputted to an optimization algorithm and verify that the location was beneficial.

Finally, the last area for improvement on this concept is to explore other methods that are built upon self-organizing maps. For example, the generative topographic map or the equalized orthogonal mapping method could insure proper training and therefore provide a more meaningful output map.

5.3 Acknowledgements

This thesis would not have been possible without the help and support of many individuals to whom I owe my successes. I would like to take this opportunity to express my gratitude and thanks to all of my supporters.

First and foremost, I would like to thank my family for their love and support throughout my college career. I would not be here today without your words of encouragement and your continued support. I would also like to recognize my fiancé, Kathleen Mettel, for the influence that she has had on my life. I would not be the individual that I am today without her behind me.

Next I would like to thank my advisor, Dr. Eliot Winer, who not only provided me with this tremendous opportunity, but helped guide me through the past three years of learning and growing. I want to also thank Dr. Amy Kaleita for her guidance on the research that led me to this point, her consistent enthusiasm for my work, and her support through the many challenges that arose during research.

Furthermore, I owe the success of this thesis to my research group of Kristin Crawford, Stephanie Kaphingst, Linda Geiger, Joe Goering, and Trevor Richardson. Without your continued effort on our research project, I would not have gained the knowledge to create this thesis.

To my colleagues: Eric Foo, Bethany Juhnke, Vijay Kalivarapu, Andrew Koehring, Kenny Kopecky, Marisol Martinez, Brandon Newendorp, Christian Noon, Joanna Peddicord, Catherine Peloquin, Brice Pollock, Levi Swartzentruber, Mike VanWartenhuzen, and Ruquin Zhang thank you for your continued help when I ran into the many bugs in my code. I would also like to mention Joseph Holub for being my right hand man, as we accomplished many things from classes and projects to a marathon.

Last but not least, I want to thank the faculty and staff of the Virtual Reality Applications Center (VRAC) at Iowa State University for your dedication to the students and the lab. You have provided me a wonderful experience and a great place to learn.

6 References

[1] M. Molinari. (2006, April) Engineering Design Centre. [Online]. http://www-edc.eng.cam.ac.uk/research/processmanagement/pm3/designspace/

[2] Ozen Engineering, Inc. Ozen Engineering. [Online]. http://www.ozeninc.com/images/Pareto4.gif

[3] J. Nocedal and S.J. Wright, *Numerical Optimization*. New York, USA: Springer, 2000, ISBN: 978-0387987934.

[4] R.L. Rardin and R. Uzsoy, "Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial," *Journal of Heuristics*, vol. 7, no. 3, pp. 261-304, May 2001.

[5] D.H. Wolpert and W.G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, April 1997.

[6] M.E. Tipping and C.M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611-622, 1999.

[7] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1-6, November 1998.

[8] J. Eddy and K. Lewis, "Multidimensional Design Visualization in Multiobjective Optimization," in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and*

Optimization, Atlanta, GA, 2002, p. 5621.

[9] E.H. Winer and C.L. Bloebaum, "Development of visual design steering as an aid in large-scale multidisciplinary design optimization," *Structural and Multidisciplinary Optimization*, vol. 23, no. 6, pp. 412-424, July 2002.

[10] E.H. Winer and C.L. Bloebaum, "Visual Design Steering for Optimization Solution Improvement," *Structural and Multidisciplinary Optimization*, vol. 22, no. 3, pp. 219-229, 2001.

[11] P-W. Chiu, A.M. Naim, K.E. Lewis, and C.L. Bloebaum, "The hyper-radial visualization method for multi-attribute decision-making under uncertainty," *International Journal of Product Development*, vol. 9, no. 1-3, pp. 4-31, 2009.

[12] D.F. Swayne, D. Cook, and A. Buja, "XGobi: Interactive dynamic data visualization in the X window system," *Journal of Computational and Graphical Statistics*, vol. 7, no. 1, p. 113, 1998.

[13] G. Stump, T.W. Simpson, M. Yukish, and L. Bennett, "Multidimensional Visualization and Its Application to a Design by Shopping Paradigm," in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Conference*, Albany, NY, 2002.

[14] G.M. Stump, M.A. Yukish, J.D. Martin, and T.W. Simpson, "The ARL Trade Space Visualizer: An Engineering Decision-Making Tool," in *10th AIAA/ISSMO Multidiciplinary Analysis and Optimization Conference*, Albany, NY, 2004.

[15] M. Fleischmann and W. Strauss. (2008, December) Digital Sparks - Semantic Map. [Online]. http://netzspannung.org/digital-sparks/

[16] S. Haykin, *Neural Networks A Comprehensive Foundation*, 2nd ed.: Prentice Hall Publishing, 1999, ISBN: 978-0132733502.

[17] J. Hollmen. (1996, March) U-Matrix. [Online]. http://www.cis.hut.fi/jhollmen/dippa/node24.html#SECTION00524100000000000000

[18] C.M. Bishop, M. Svensen, and C.K.I. Williams, "GTM: The generative topographic mapping," *Neural Computation MIT Press*, vol. 10, no. 1, p. 215, 1998.

[19] C.M.E. Holden and A.J. Keane, "Visualization Methodologies in Aircraft Design," in *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, 2004, pp. 1-13.

[20] Z. Meng and Y-H Pao, "Visualization and Self-Organization fo Multidimensional Data through Equalized Orthogonal Mapping," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, pp. 1031-1038, July 2000.

[21] M.C. Su, Y.X. Zhao, and J. Lee, "Som-based Optimization," in *IEEE International Conference on Neural Networks*, 2004, pp. 781-786.

[22] M. Milano, P. Koumoutsakos, and J. Schmidhuber, "Self-organizing nets for optimization," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 756-758, 2004.

[23] S. Obayashi and D. Sasaki, "Visualization and Data Mining of Pareto Solutions Using Self-Organizing Maps," *Evolutionary Multi-Criterion Optimization*, vol. 2632, p. 71, 2003.

[24] P.C. Matthews, "The Application of Self Organizing Maps in Conceptual Design," Engineering Department, Cambridge University, PhD Thesis 2001.

[25] T.T. Tanimoto, "An elementary mathematical theory of classification and prediction," IBM Corporation, New York, NY, Technical Report 1958.

[26] A.C. Clark and E.N. Wiebe. (2000) NC State University. [Online]. http://www.ncsu.edu/scivis/lessons/colormodels/color_models2.html

[27] T. Streeter. (2006, Fall) SelfOrganizingMap. [Online]. http://www.tylerstreeter.net/#SelfOrganizingMap

[28] G. Schneider and P. Schneider, ""Promiscuous" Ligands and Targets Provide Opportunities for Drug Design," in *Systems Chemistry*, Bozen, Italy, 2008.

[29] Mathworks. (1998) Neural Network Toolbox. MATLAB.

[30] Peltarion. (2008) Synapse. Software.

[31] Khronos Group. (2009, August) OpenGL. API.

[32] Nokia. (2010, January) Qt. API.

[33] A. Hedar. Global Optimization Methods and Codes. [Online]. [http://www-optima.amp.i.kyoto-

u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

[34] (2006) GEATbx: Genetic and Evolutionary Algorithm Toolbox. [Online]. http://www.geatbx.com/docu/fcnindex-01.html

[35] F. Mulier and V. Cherkassky, "Self-Organization as an Iterative Kernel Smoothing PRocess," *Neural Computation*, vol. 7, pp. 1165-1177, 1995.

[36] S. Kasi. (1997, September) Neural Networks Research Centre. [Online]. http://www.cis.hut.fi/research/som-research/worldmap.html

[37] A. Hedar. Global Optimization Methods and Codes. [Online]. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2056.htm

[38] E.L. Koua and M.J. Kraak, "Geovisualization to support the exploration of large health and demographic survey data," *International Journal of Health Geographics*, vol. 3, no. 21, 2004.

[39] R. Der, G. Balzuweit, and M. Herrmann, "Building Nonlinear Data Models with Self-Organizing Maps," *Lecture Notes on Computer Science*, vol. 1112, p. 821, 1996.