

DETC2011-47( , %

## COMPLEXITY AS A SURROGATE MAPPING BETWEEN FUNCTION MODELS AND MARKET VALUE

**James L. Mathieson**  
Research Assistant  
Mechanical Engineering  
Clemson University  
Clemson, SC 29634-0921  
jmathie@clemson.edu

**Aravind Shanthakumar**  
Research Assistant  
Mechanical Engineering  
Clemson University  
Clemson, SC 29634-0921  
ashanth@clemson.edu

**Chiradeep Sen**  
Research Assistant  
Mechanical Engineering  
Clemson University  
Clemson, SC 29634-0921  
csen@clemson.edu

**Ryan Arlitt**  
Research Assistant  
School of Mechanical, Industrial  
and Manufacturing Engineering  
Oregon State University  
Corvallis, OR 97331-6001  
alitr@onid.oregonstate.edu

**Joshua D. Summers**  
Associate Professor  
Mechanical Engineering  
Clemson University  
Clemson, SC 29634-0921  
jsummers@clemson.edu  
(corresponding author)

**Robert Stone**  
Professor  
School of Mechanical, Industrial  
and Manufacturing Engineering  
Oregon State University  
Corvallis, Oregon 97331-6001  
rstone@engr.orst.edu

### ABSTRACT

The purpose of this paper is to investigate if early stage function models of design can be used to predict the market-value of a commercial product. In previous research, several metrics of complexity of graph-based product models have been proposed and suitably chosen combinations of these metrics have been shown to predict the time required in assembling commercial products. By extension, this research investigates if this approach, using new sets of combinations of complexity metrics, can predict market-value. To this end, the complexity values of function structures for eighteen products from the Design Repository are determined from their function structure graphs, while their market values are procured from different vendor quotes in the open market. The complexity and value information for fourteen samples are used to train a neural net program to define a predictive mapping scheme. This program is then used to predict the value of the final four products. The results of this approach demonstrate that complexity metrics can be used as inputs to neural networks to establish an accurate mapping from function structure design representations to market values to within the distribution of values for products of similar type.

**Keywords:** complexity, function structures, neural networks

### 1 COMPLEXITY IN ENGINEERING DESIGN

In engineering design, complexity is often addressed with the principle of “keep it simple” [1,2]. While this is a nearly

universal goal, the question remains as to what is complex. Metrics are needed to quantify the complexity of a design to enable comparison and decision making based on the complexity level [3]. Even when objective means of assessing the complexity of a design, process, or representation is available, there is not a clear mapping between these measures and meaningful measures of product performance [4]. However, it has been shown that such mappings can be established, such as using the metrics for surrogate models to predict assembly time on basic component connectivity graphs [5]. This paper seeks to establish such a mapping between complexity metrics on function structures and the market value of the end products which they represent.

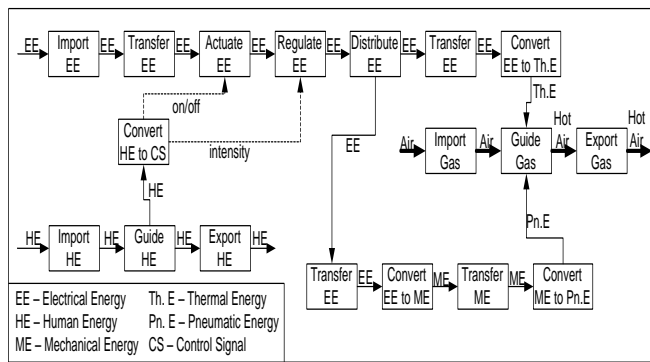
Previous studies have explored defining different views of engineering design complexity, such as size, connectivity, and solvability, and have shown that each of these views contribute differently to the understanding [6,7]. In this work, we seek to extend the use of complexity metrics beyond simply characterizing design product models by using the metrics to capture hidden and undefined knowledge. To this end, if a collection of metrics are found that can be used to accurately predict product performance from early stage design information models, then engineering tools can be developed to enhance the design process. Function structures are chosen as a good representation of early stage design information as they capture connectivity and intent at a high level of abstraction.

Twenty nine complexity metrics, offered in the literature, are used here to try and capture implicit information contained within the function structures that might predict sales price of

different products. These metrics are classified into four different groups: size, interconnection, centrality, and decomposition. A full discussion of these metrics can be found in [4,8]. These metrics are discussed further in Section 3.3.

## 2 FUNCTION STRUCTURES: EARLY STAGE DESIGN REPRESENTATIONS

A function structure is a graph-based model of mechanical product functionality, whose nodes are transformative actions and edges are flows undergoing transformations through a design product [2]. These graphs are discussed in design research as a useful representation to support early design activities, such as problem decomposition and understanding, solution search [2,9], concept generation [10,11], and design archival [12]. Figure 1 shows a function structure of a hairdryer, where the flow names are abbreviated as per the legend.



**Figure 1: Function structure of a hairdryer stored in the Design Repository<sup>1</sup>**

As a first step to formalize the terms used in these models, controlled vocabularies of functions and flow terms have been proposed. Notable examples are the terms discovered through engineering forensic studies of army helicopters [13], the vocabulary *motion, control, power, and enclose* [14], the NIST vocabulary [15], and the Functional Basis [16,17]. This last vocabulary contains 53 function verbs and 45 flow nouns, organized in a three-tier hierarchy. These terms were identified through systematic reverse engineering of consumer products [18] and have been used to catalog functional information of electromechanical products stored in the Design Repository [12]. The function structure shown in Figure 1 is obtained from this archive.

The function structure representation, and especially the design information stored in the repository, have been used to develop many tools for early design reasoning. Examples are the concept generator [10,19], functional similarity analyzer [20], the failure prediction and propagation analysis tool [21,22], a graph grammar tool that automatically synthesizes multiple decomposed version of a given black box function

[11], and a tool to generate component configurations for new design [23]. These tools rely upon historical reasoning, using the data from the Design Repository models to establish trends to be reused in new designs. The work presented in this paper also uses this historical based information and does not rely on any analytical and physics based reasoning within the models themselves.

## 3 DATA COLLECTION

The function-model of the consumer products for the research is obtained from the Oregon State University's design repository which contains 167 products and 6447 artifacts. The repository is a work of collaborative effort of researchers from Oregon State University, Missouri University of Science and Technology (formerly University of Missouri – Rolla), The University of Texas at Austin, and NIST. The repository provides a description, function structure, and connectivity graph for each product

A total of eighteen products were selected for this research, to cover the wide range of available product types and market value while maintaining a consistent level of detail and quality in the provided function structure. The cost range of the products analyzed range from less than \$10 to over \$200. The function-models of these products are converted into bipartite graphs, the properties of which form a complexity vector describing each product.

### 3.1 Market Values

The market value of each product is determined by identifying an equivalent product to that described in the design repository. While the design repository typically does not cite exact models and the product shown is not likely to still be on the market in new form, the manufacturer is identified. This allows a representative current product by the same manufacturer to be selected by matching product features to the function structure.

Quotes for each of these products are queried from the Google product search engine<sup>2</sup>. Five base-price quotes are taken spanning the price range for each product. Quotes originating from auction and marketplace sites as well as for used and refurbished products are not used. Many additional quotes are not used due to the limited number of vendors for some products. The used quotes are shown in Table 1, indicating the purchase price, but not including shipping, handling, and tax. Brand names have been omitted in this table. The quote orders are sorted by relevance as determined by the internal Google algorithm in product search.

<sup>1</sup> <http://repository.designengineeringlab.org/> accessed on December 31, 2010

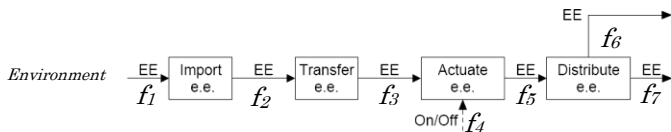
<sup>2</sup> Google Product Search <http://products.google.com>

**Table 1: Quotes for Products Selected from Design Repository**

No.	Product Name	Quote 1	Quote 2	Quote 3	Quote 4	Quote 5	Mean
1	Electric Toothbrush	\$ 8.16	\$ 6.99	\$ 7.79	\$ 11.98	\$ 11.08	\$ 9.20
2	Sander	\$ 49.99	\$ 39.99	\$ 49.00	\$ 43.34	\$ 46.42	\$ 45.75
3	Dog Toy	\$ 7.00	\$ 7.95	\$ 8.89	\$ 6.99	\$ 7.95	\$ 7.76
4	Garage Door Opener	\$ 179.99	\$ 179.99	\$ 179.99	\$ 179.99	\$ 179.99	\$ 179.99
5	Hot Air Popper	\$ 14.88	\$ 24.99	\$ 24.81	\$ 17.99	\$ 29.99	\$ 22.53
6	Robotic Vacuum	\$ 175.00	\$ 199.97	\$ 199.00	\$ 179.99	\$ 199.99	\$ 190.79
7	Juice Extractor	\$ 69.99	\$ 69.99	\$ 75.29	\$ 60.22	\$ 48.54	\$ 64.81
8	Mixer	\$ 35.94	\$ 28.39	\$ 39.99	\$ 28.39	\$ 42.49	\$ 35.04
9	Hair Dryer	\$ 22.39	\$ 24.99	\$ 24.49	\$ 29.99	\$ 19.01	\$ 24.17
10	Circular Saw	\$ 61.92	\$ 69.99	\$ 61.92	\$ 58.99	\$ 74.99	\$ 65.56
11	Foam Dart Gun	\$ 14.99	\$ 10.39	\$ 9.99	\$ 12.95	\$ 10.89	\$ 11.84
12	Nail Gun	\$ 66.00	\$ 59.00	\$ 69.00	\$ 69.95	\$ 79.99	\$ 68.79
13	Microwave	\$ 157.52	\$ 169.97	\$ 199.00	\$ 167.00	\$ 189.00	\$ 176.50
14	Lawn Mower	\$ 149.88	\$ 149.88	\$ 149.88	\$ 149.88	\$ 149.88	\$ 149.88
15	Jigsaw	\$ 49.97	\$ 59.99	\$ 52.99	\$ 49.99	\$ 54.34	\$ 53.46
16	Sewing machine	\$ 116.54	\$ 199.99	\$ 149.99	\$ 129.99	\$ 149.99	\$ 149.30
17	35 mm Man. P&S Camera	\$ 44.95	\$ 45.99	\$ 44.95	\$ 38.99	\$ 45.99	\$ 44.17
18	Video Cassette Recorder	\$ 176.40	\$ 228.84	\$ 249.99	\$ 229.99	\$ 174.99	\$ 212.04

**3.2 Bipartite Modeling**

The conversion of a function model into a bipartite graph begins with labeling the flows (energy, material, and signal [2]) as  $f_i$ . This is done to verify later that no ‘flow’ is missed in creating a bipartite model. Figure 2 shows an example of a function structure in which the flows have been labeled. This serves as a reference document for converting the function structure representation.



**Figure 2: Flow labeled for tracking while conversion into bipartite graph**

From this reference document, each function block in the model is listed in a column and assigned an element ID for each (Figure 3). The number of function-blocks in the list should match with the total number of function-blocks in the reference document. This is done to ensure that each function-block in the model is uniquely identified. This check in conjunction with ensuring that all flows in the model are also uniquely identified provides control over the quality of the document.

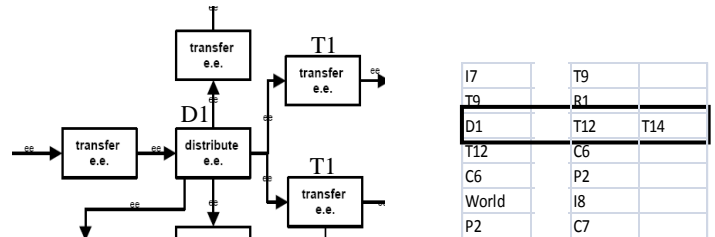
Next, two additional columns, labeled “source” and “sink”, are created to represent the origin and destination to the flow. Next, the function-block in the model from which flow

originates, such as  $f_2$  in Figure 2, is listed under the heading ‘source’ and the function-block where the flow ends is listed under the column ‘sink’. This new two column representation indicates that a flow originates from a particular source and ends at a specific sink as seen in Figure 3. This tabular coding of the bipartite graph is done to interface with the computational implementation of the complexity metrics.

Element ID	Description	Sources	Sinks
I1	Import EE	Environment	I1
T1	Transfer EE	I1	T1
A1	Flow ' $f_2$ ' in Figure 2	T1	A1
D1		A1	D1
R1	Regulate EE	D1	R1
C1	Convert EE to ME	R1	C1
C2	Change ME	C1	C2

**Figure 3: Representation showing element ID, description, and flow from source to sink**

This procedure is followed to cover all the function-blocks and flows in the function-model. The number of entries under the headings ‘source’ and ‘sink’ (i.e., total number of rows) should be equal to the total number of flows in the function-model. Additionally, if a flow has multiple sinks originating from a common source (flow branching), then all such sinks are listed under the column ‘sink’ in a single row next to each other as seen in Figure 4.



**Figure 4: Two function-blocks originating from the same flow are listed in a single row**

The same idea is carried over to represent flows that branch from multiple sources to merge into a sink wherein multiple sources are listed under the column ‘source’ in a single row next to each other. The source of the flow which enters the system from outside the system boundary is called “Environment” to represent its origin as the environment outside the system. Similarly, the sink to a flow that is outside the system boundary is also called “Environment” (see “Environment” in Figure 2 and Figure 3).

### 3.3 Complexity Analysis

The bipartite representation of each function structure is now subjected to analysis of its complexity properties. The properties derived from the graph include size, interconnection, centrality, and decomposition. These are taken from previously published work [4] and extended with additional measurements [8] for a total of 29 metrics. All of these are graph based metrics and depend only on the structure or topology of the graph and not on the vocabulary of the nodes and arcs. Thus, issues associated with vocabulary inconsistencies within functional representations is not a limiting factor in this analysis [24,25]. The analysis which determines these metrics is implemented computationally in MATLAB with a combination of self-developed functions and the MatlabBGL implementation of the Boost Graph Library [26,27].

In the size class of metrics, there are two sub-types. Dimensional size includes the raw counts of input elements and relationships, in this case functions and flows. Connective size addresses parametric degrees of freedom (number of interactions) and connectivity (number of graph connections). It should be noted that in a graph composed entirely of one-to-one connections the degrees of freedom value will be identical to the count of relationships and connectivity will be exactly twice this. In the context of function structures, these values are indicative of the existence of branched and merged flows.

Interconnection divides into the sub-types of shortest path length and maximum flow rate between any two elements. The analysis of these properties yields matrices of the value between each pair of nodes. These matrices are converted into four metrics each by taking the sum, maximum, mean, and density of the matrix. The density is the mean of the matrix divided by the relationship count, representing the impact of relationships on the sub-type.

This same suite of summary metrics is applied to the vector-based results of centrality and decomposition. In centrality, betweenness is the number of shortest paths which pass through each element while clustering coefficient is the percentage of elements a particular element is connected to which are also connected to each other.

In decomposition, in and out core numbers are the point at which an element's degree falls to zero in the given direction when all elements of that number are removed. Another value classified under decomposition is the Ameri-Summers algorithm [7]. This algorithm systematically breaks down the graph by the least connected relationships necessary to remove an element until the degree of all elements is zero. In this process, a score is built up based on the multiplication of iteration, remove set size, and the number of sets which can be removed without removing an element of higher degree.

Table 2 presents the results of this analysis for the first seven of the eighteen products selected. It can be seen here how metrics vary independently across the range of products. Also, it can be noted that the metrics span several orders of magnitude, both between metrics as well as within metrics.

**Table 2: Complexity Metrics for Product Function Structures**

Class	Type	Dir	Metric	Prod.1	Prod.2	Prod.3	Prod.4	Prod.5	...	Prod.18	
Size	Dim		Elements	17	27	32	34	18	...	63	
			Rel.	20	37	43	49	24	...	96	
	Conn.		DOF	21	37	44	49	25	...	97	
			Conn.	43	76	87	98	49	...	193	
Interconnection	Shortest Path		Sum	831	3379	5456	3217	271	...	28091	
			Max	10	13	14	10	7	...	18	
			Mean	3.06	4.81	5.50	2.87	0.89	...	7.19	
			Density	0.15	0.13	0.13	0.06	0.04	...	0.07	
	Flow Rate		Sum	226	742	1069	876	122	...	4052	
			Max	3	6	6	5	3	...	9	
			Mean	0.78	1.02	1.04	0.76	0.38	...	1.02	
			Density	0.04	0.03	0.02	0.02	0.02	...	0.01	
Centrality	Betweenness		Sum	633	2703	4464	2468	177	...	24185	
			Max	133	482	652	486	39	...	3314	
			Mean	37.24	100.11	139.50	72.59	9.83	...	383.89	
			Density	1.86	2.71	3.24	1.48	0.41	...	4.00	
	Clustering Coefficient		Sum	0.00	0.00	0.67	0.00	0.22	...	0.00	
			Max	0.00	0.00	0.50	0.00	0.08	...	0.00	
			Mean	0.00	0.00	0.02	0.00	0.01	...	0.00	
			Density	0.0000	0.0000	0.0005	0.0000	0.0005	...	0.0000	
Decomposition	Ameri-Summers			27	119	58	174	83	...	362	
	Core Numbers	In		Sum	29	53	64	56	18	...	126
				Max	2	2	2	2	1	...	2
		Out		Mean	1.71	1.96	2.00	1.65	1.00	...	2.00
				Density	0.09	0.05	0.05	0.03	0.04	...	0.02
	Core Numbers	In		Sum	34	54	64	67	18	...	126
				Max	2	2	2	2	1	...	2
		Out		Mean	2.00	2.00	2.00	1.97	1.00	...	2.00
			Density	0.10	0.05	0.05	0.04	0.04	...	0.02	

## 4 NEURAL NETWORK PARAMETERS

The vectors of metrics generated by the complexity analysis and quotes for each product are now used as inputs and targets respectively for neural network training. Our intent is not to present new neural network architectures or methods, but to explore their application as a pattern discovery tool in this domain. Here, we present the process of selecting the general neural network architecture, training parameters, and layer parameters. The goal of this process is to identify the network parameters which most consistently predict the relationship between the complexity metrics and the market values.

### 4.1 Network Architecture

The selection of a general architecture for the neural networks requires a consideration of the intended purpose of the network. Our intent is to establish a function (mapping) on 29 input variables (complexity scores for the products) to achieve a single continuous output (cost for each product). This problem lends itself well to the use of backpropagation network architectures. These architectures apply a generalization of the Widrow-Hoff learning rule to multiple-

layer networks with differentiable nonlinear transfer functions [28,29,30].

There are different types of backpropagation neural networks: feed-forward, Elman, cascade-forward, and dynamic networks. Dynamic networks are not considered for use because the input and output responses are both static cases. Similarly, Elman networks are not considered due to the time dependency of recurrent layers. This leaves feed-forward and cascade-forward networks for consideration. Cascade forward networks are an extension of feed-forward networks in that each layer is provided both the input vector as well as the results of all previous layers instead of only being exposed to the layer immediately preceding it. This additional access to information may improve the speed with which a cascade-forward network can learn complex relationships, an important property given the low training set size available.

**Thus, the justification for the choice in a cascade-forward network is motivated by our low training set size which is time-independent.**

#### **4.2 Training Parameters**

A backpropagation neural network is technically capable of fitting to any function given a sufficient number of neurons and training cycles. As such, the challenge in training such networks is in maintaining generalization of the network such that it will accurately predict new data. To this end, backpropagation networks use training methods with early-stopping. These methods require that the input data set be divided into training (60%), test (20%), and validation sets (20%). The training is terminated after the performance of the validation set does not improve after a set number of training epochs. However, this introduces an issue given the nature of the data provided here.

The neural network is presented with the same input for each of the five target values observed in the marketplace. The division of the input set will inherently bias the network's output toward those targets which remain in the training set. Further, the multiple presentation of identical inputs means that the internal validation of the network in training is not reliable.

To address the issue of training bias, an interleaving division algorithm is used. This algorithm selects a new random division of the input data into train, test, and validation sets on each training epoch. In this way, the network is both trained and internally validated against all of the input data (fourteen data sets). However, this requires that an external validation be performed on an entirely new set of data. For this reason, the final four products are omitted from the input data set for use in validation.

**Thus, the justification for the choice of interleaved training set division and exclusion of four products from the input set is motivated by multiple presentations of identical inputs with different target values.**

To further improve generalization of the network, regularization is used. This can be done either through Bayesian regularization or by modifying the performance

function. Bayesian regularization precludes early stopping and is much slower than other training functions and is thus not used [31,32]. Instead, a performance function which consists of the mean of the sum of squares of both network errors and network weights and biases is used. These two terms are combined as a weighted sum. This weight, or performance ratio, is set to 0.9, favoring mean square error.

**Thus, the justification for the choice in a weighted regularization of training performance function against means square error and network mean square weights and biases is motivated by a need to limit the computational demand.**

The final step in establishing training parameters is to select a training algorithm. The scaled conjugate gradient algorithm is used here due to performing well over a variety of problems and particularly in networks which have a very large number of weights, such as is the case with cascade-forward architectures. Additionally, the use of interleaving division for early stopping is best applied in training algorithms with smaller step sizes. This precludes the use of large step size training algorithms such as Levenberg-Marquardt.

**Thus, the justification for the choice in scaled conjugate gradient training is motivated by a need for broad performance and smaller step size.**

#### **4.3 Layer Parameters**

The particular arrangement of neurons, layers, and transfer functions can significantly impact the performance of the network. Specifying the output layer to always be a single purely linear neuron, the remaining layers are given a hyperbolic tangent sigmoid transfer function. This function 'squashes' the output to between -1 and 1. The hidden layers of the network define the shape of the function and are then scaled by the linear output function.

What is not well defined is the number of layers and neurons in each layer which will best model the relationship between the complexity metrics and market value. Given that it is desired that the network generalize well to new data, it is known that the number of neurons should be as low as possible while still being able to model the desired function. Further, additional layers may allow the network to learn complex relationships faster while a single hidden layer should be able to model any function given sufficient training. We consider networks with up to fifteen neurons and up to three hidden layers. This network design space is further reduced by limiting the number of neurons in any given layer to the maximum number of neurons divided by the number of hidden layers. This creates a design space with 189 different network structures: 1-15 neurons in a single hidden layer, all combinations of 1-7 neurons in each of two hidden layers, and all combinations of 1-5 neurons in each of three hidden layers.

**Thus, the justification for the choice in testing multiple neural network structures is motivated by the competing priorities of generalization and relationship complexity.**

The remaining issue in identifying the best architecture is the fact that, due to randomizations in the training and division algorithms, no two trainings of the same network architecture will yield exactly the same results. Thus, it is necessary to train many copies of the same network architecture and draw conclusions from the distribution of results. One hundred copies of each network structure are trained and simulated. The results of these simulations are then used to generate a kernel smoothing probability density estimate, bounded to positive values, for each product. The raw and normalized probability densities at the expected mean value are then taken for each product from these estimates.

**Thus, the justification for the choice in replication and probability density estimates is motivated by the stochastic nature of neural network training.**

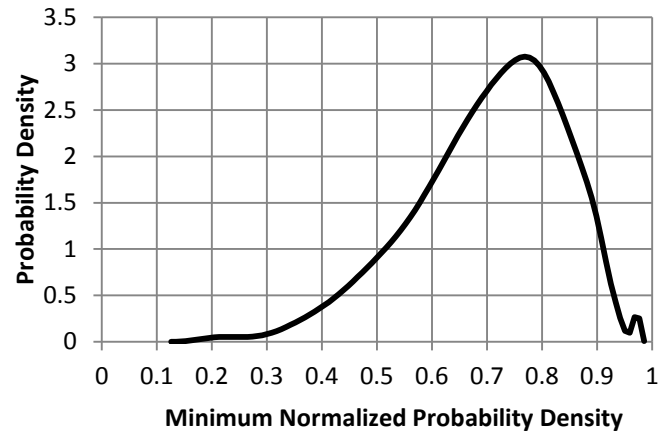
## 5 NETWORK PERFORMANCE RESULTS

The performance of the networks is evaluated in two stages. The performance of neural networks in general is addressed through an analysis of the distribution of responses given from all of the neural network structures in the design space. A performance measure is applied on these results to identify network structures which perform well in both accuracy and precision. The best network structure by this measure is then used to present output distributions for each other the four validation products.

### 5.1 Network Structure Evaluation

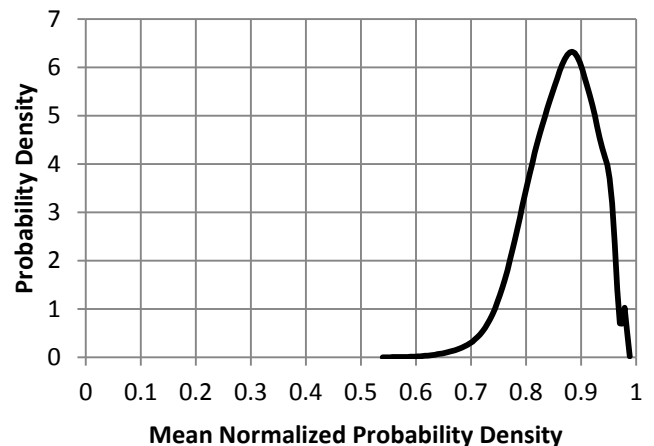
In assessing the performance of network structures we observe the probability density of the expected mean value in the distribution of outputs given by that structure for the four products not presented in training. The density value can be addressed in two different ways. The first of these is the value itself, which is indicative of the likelihood that particular value will be returned. The second approach is to normalize this value within its distribution, representing how close to the maximum density the target value is. For both of these approaches both the minimum and mean values for the validation set are of importance. Together, these values can be used to strike a balance between networks which with high confidence and those which predict with high accuracy.

When these measures are taken for each network structure, the distribution of the normalized probability density can demonstrate the general performance of the neural network approach in estimating the market value from function structure complexity metrics. Figure 5 shows the distribution of the minimum observed normalized probability density values from the external validation set. This shows that there is a 90% probability that all of the expected values from the validation set will have a probability density at least half that of the maximum density for the distribution of results.



**Figure 5: Probability Density Distribution of Minimum Normalized Probability Density Values for External Validation Products**

The performance of the neural networks is further supported when the mean normalized probability density is considered. It can be seen in Figure 6 that not only does the mean fall into a narrow band, but this band is near the upper end of the range. In fact, there is a 95% probability that the mean of the probability density values will be greater than three quarters of the maximum density in any given distribution. Together, Figure 5 and Figure 6 demonstrate the ability of neural networks to make accurate predictions of the market value from function structure complexity metrics.



**Figure 6: Probability Density Distribution of Mean Normalized Probability Density Values for External Validation Products**

The final step is to select a network structure which performs well in both accuracy and precision so that the output may be analyzed. To this end, the minimum and mean of both the probability density value and normalized density value are normalized across the results for all of the structures and given an equal weight to create a performance score for each network

structure. Table 3 shows the top performing network structures by this performance function. The networks identified are quite similar in that all but the third ranked network have three hidden layers. Further, there appears to be a pattern of having two or more hidden layers of the same size. It should be noted that the single network with two hidden layers shown achieves a much higher accuracy in terms of the expected value's relation to the maximum density, but apparently at the expense of precision. Similarly, the second ranked network achieves a higher minimum precision at the expense of overall precision.

**Table 3: Top Performing Neural Network Structures**

NN Structure	Probability Density		% of Max. Density		Perf.
	Min	Mean	Min	Mean	
[2,2,2]	0.0029	0.0091	0.90	0.96	0.0866
[2,3,3]	0.0034	0.0077	0.84	0.90	0.0838
[5,6]	0.0023	0.0085	0.97	0.98	0.0835
[1,5,1]	0.0028	0.0078	0.90	0.96	0.0826
[1,1,1]	0.0026	0.0083	0.90	0.95	0.0825
[4,4,5]	0.0030	0.0082	0.78	0.95	0.0818

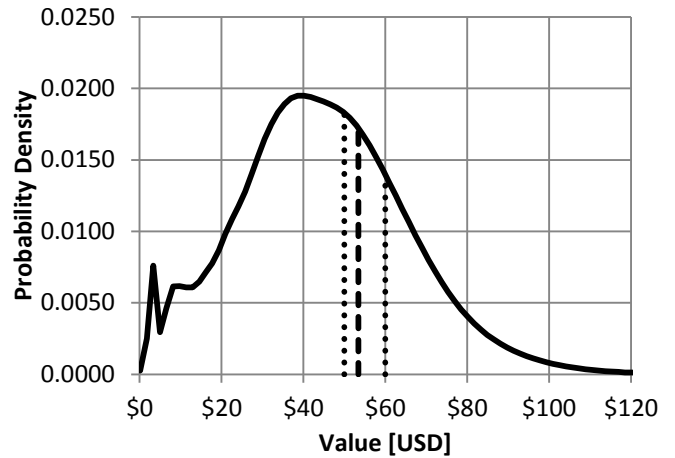
**5.2 Validation Results**

Taking the best performing network structure identified, a network with three hidden layers having two neurons each, we can now address the particular output of the neural networks. Again, this is approached in terms of probability density functions as no two trainings of the same network structure will be identical. The density functions presented here are derived from 1000 unique trainings of this network structure.

In Figure 7, the estimated probability density function for product 15, a jig saw, is shown. The dotted vertical lines denote the minimum and maximum value of the selected product observed in the marketplace, while the dashed line represents the mean of observed values. This same convention is repeated in all subsequent figures in this section.

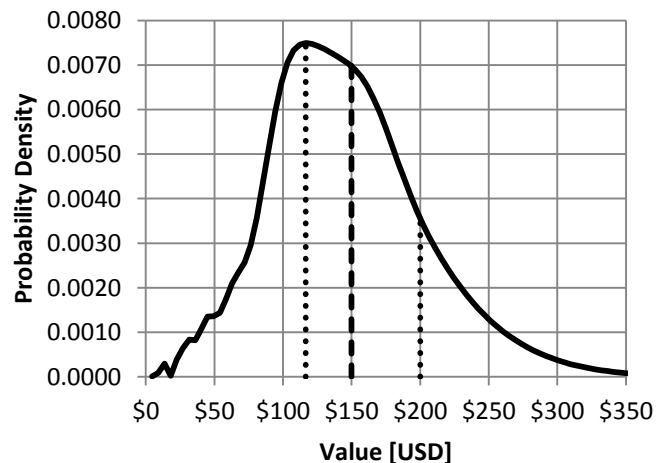
As is to be expected based on the results of network structure performance, the mean of observed values has a probability density 90% of the maximum density. The narrow range of market values observed for the product is apparent in Figure 7 and the integral of the observed market values which indicates only a 14% probability of returning a value within this range. However, it should be noted that this represents the market range for a single jig saw model. A broader observation of jig saws available on the market yields models of varying feature sets and manufacturer ranging from \$11 to several hundred dollars. This is quite telling, considering that this particular model represents an upper-end consumer product, lending itself to its position on the upper side of distribution. While commercial products may be valued at several hundred dollars, products intended for the consumer market are rarely valued above \$150. In this way, the neural networks are

providing a reasonably accurate picture of value range for the product type described, if not the exact model.



**Figure 7: Probability Density Function for Product 15 (Jig Saw)**

The nature of the validation distributions to describe the value range of the product type is continued in Figure 8. The broad range of values for the specific product model, a sewing machine, accounts for a 54.5% probability that the value returned will be within the observe range. However, as with jig saws, the range of consumer sewing machine values in general range from \$65 to \$350 at which point industrial and embroidery machines begin to dominate. The particular product model shown addressed here represents this manufacturer's highest-end sewing machine that does not include digital controls.

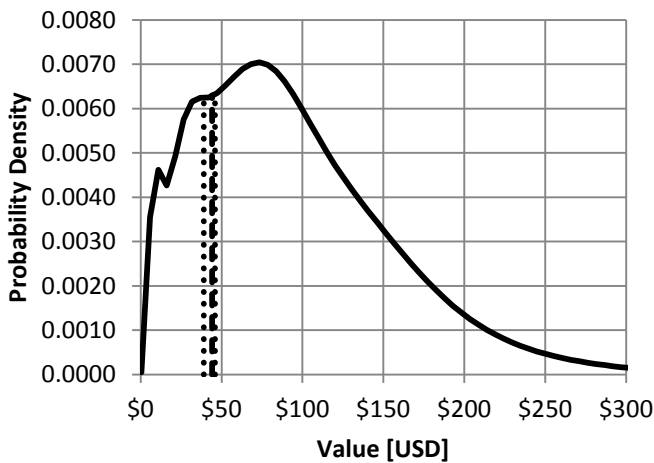


**Figure 8: Probability Density Function for Product 16 (Sewing Machine)**

It is interesting to note that the minimum, mean, and maximum observed value each fall on a change in the distribution. The minimum observed value corresponds with

the greatest density. This is to be expected considering the nature of vendors to compete on price, driving the majority of the observed market values toward the lower end. This is further illustrated by the steep decline in density following the mean and the inflection of the function following the maximum.

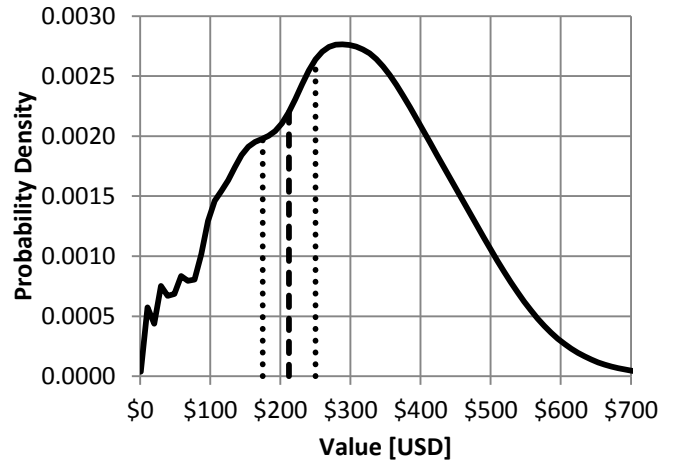
Figure 9 is an extreme case of a narrow observed range of values for the particular model while providing a fair representation of the product type. In this case, the product is a 35mm manual operated point and shoot camera. The particular model whose values are observed is a mid-range item which can be reloaded. However, the description provided could describe anything from a \$10 disposable to a \$250 professional unit. This is reflected in the distribution. While the probability of returning a value within the narrow \$7 range of observed values is only 3.5%, the distribution as a whole is a reasonable representation of the marketplace for a product of this type.



**Figure 9: Probability Density Function for Product 17 (35mm Manual P&S Camera)**

The final validation product, a video cassette recorder, presents an interesting case in terms of market value as this is an obsolete product in the consumer marketplace. As such, it would be intuitive if the value was considered to be less than that of the superseding technology, digital video discs. However, given that stand-alone video cassette recorders are no longer manufactured in consumer quantities, but rather primarily as professional videographer equipment, the observed market value is significantly higher than consumer digital video disc players. The particular model represented in Figure 10 is the least expensive stand-alone video cassette recorder which could be found in new production. Other products of this type now either use a digital storage format and are valued in the \$300 to \$600 range or are combined with a digital video disc player and are valued in the \$70 to \$200 range. In this way, the very wide distribution seen in Figure 10 is not unreasonable. It includes what few stand-alone VHS players remain on the market near the peak of the distributions, returning a value within the observed range for the given model 16.7% of the

time, while also covering the value of more modern products of similar construction.



**Figure 10: Probability Density Function for Product 18 (Video Cassette Recorder)**

These results demonstrate an ability to characterize the market range of products which have not been previously trained. Given the small training set size, this is an acceptable level of performance. This suggests that complexity metrics can be used as neural network inputs to effectively map between function structures and market values.

## 6 CONCLUSION

In this paper, we have explored a means for assessing the market value of a product from the function structures of those products using graph-based complexity metrics as a mapping through neural networks. Graph-based complexity metrics are used to interpret function structure design representations in the form of a consistent-length vector. This enables the training of neural networks on the relationship between this complexity vector and observed market values.

Results of this exploration have shown that neural networks are highly capable of establishing a relationship between graph complexity metrics on function structures and market value. The neural network structures tested here return a distribution of outputs which rank the expected output of all new data to be within 50% of the maximum probability density with 90% confidence. Further, the mean probability density of the expected value will be greater than 75% of the maximum probability density with 95% confidence. Thus, the use of graph complexity metrics as input to neural networks can be considered to have an acceptable degree of accuracy for this mapping.

The examination of outputs from a neural network displaying an equally weighted performance in accuracy and precision demonstrates that the precision of this approach and training set size is at the product range level. While the probability of returning a value within the observed range for a

specific product model ranges between 3.5% and 54.5%, the distribution for each new product presented is representative of the market for products of the same type. A product type level of market value prediction is an unexpected outcome given the training of the neural networks on specific model value ranges. This presents a potential alternative usage for this approach in forward design. However, if specific model value is desired, this may be achieved through the use of a significantly larger training set.

The work presented here uses unrefined function structure design representations of approximately similar fidelity. This offers a path to further research in presenting refined function structures of the same product at different fidelities. Such a study may be used to assess the sensitivity of the approach to the details of the function structure presented as well as to the specific graph properties reflected in the complexity metrics. Additionally, these function models are generated as part of a reverse engineering exercise and not as forward design. In that manner, the designed functionality may not be appropriately captured in these models. Thus, future research should include the exploration of complexity metric vectors of non-reverse engineered product structures as cost indicators. Preliminary research into these areas has suggested that there are particular graph properties that may have strong positive or negative correlations with market value. These effects are the subject of ongoing research with the goal of creating a finer set of complexity metrics for use in the surrogate modeling and to discover what underlying complexity properties are the driving forces behind the mapping.

## REFERENCES

- [1] N.P. Suh, "Axiomatic Design Theory for Systems," *Research in Engineering Design*, vol. 10, no. 4, pp. 189-209, 1998.
- [2] G. Pahl and W. Beitz, *Engineering Design: A Systematic Approach*, 3rd ed. London: Springer, 2007.
- [3] G. A. Hazelrigg, *Systems Engineering: An Approach to Information-Based Design*, 1st ed. New Jersey, NJ, USA: Prentice Hall Inc, 1996.
- [4] James L. Mathieson and Joshua D. Summers, "Complexity Metrics For Directional Node-Link System Representations: Theory and Applications," in *Proceedings of the ASME IDETC/CIE 2010*, Montreal, Canada, 2010, pp. IDETC2010-28561.
- [5] J. Mathieson, B. Wallace, and J. Summers, "Assembly Time Modeling Through Connective Complexity Metrics," in *International Conference on Manufacturing Automation*, Hong Kong, China, 2010.
- [6] J.D. Summers and J.J. Shah, "Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability," *Journal of Mechanical Design*, vol. 132, no. 2, p. 021004 (11 Pages), February 2010.
- [7] F Ameri, J.D. Summers, G.M. Mocko, and M. Porter, "Engineering design complexity: an investigation of methods and measures," *Research in Engineering Design*, vol. 19, no. 2-3, pp. 161-179, November 2008.
- [8] James Mathieson, *Connective Complexity Methods for Analysis and Prediction in Engineering Design*. Clemson, SC, USA: Clemson University, 2011.
- [9] D. G. Ullman, *The Mechanical Design Process*. New York, NY, United States of America: McGraw-Hill, 1992.
- [10] J. Vucovich et al., "Concept Generation Algorithms for Repository-Based Early Design," in *ASME 2006 IDETC/CIE*, Philadelphia, PA, USA, 2006.
- [11] P. Sridharan and M. I. Campbell, "A Study on the Grammatical Construction of Function Structures," *Artificial Intelligence for Engineering Design, Analysis & Manufacturing*, vol. 19, no. 3, pp. 139-160, 2005.
- [12] M.R. Bohm, R. B. Stone, and S. Szykman, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *Journal of Computing and Information Science in Engineering*, vol. 5, no. 4, pp. 360-372, 2005.
- [13] J. A. Collins, B. T. Hagan, and H. M. Bratt, "Failure-Experience Matrix - a useful Design Tool," *Journal of Engineering for Industry*, vol. B 98, no. 3, pp. 1074-1079, 1976.
- [14] C. F. Kirchman and G. M. Fadel, "Classifying Functions for Mechanical Design," *Journal of Mechanical Design*, vol. 120, no. 3, pp. 475-482, 1998.
- [15] S. Szykman, J. W. Racz, and R. D. Sriram, "The Representation of Function in Computer-Based Design," in *1999 ASME IDETC/DIE*, Las Vegas, NV, USA, 1999.
- [16] R.B. Stone and K. L. Wood, "Development of a Functional Basis for Design," *Journal of Mechanical Design*, vol. 122, no. 4, pp. 359-370, 2000.
- [17] J. Hirtz, R. B. Stone, D. A. McAdams, S. Szykman, and K. L. Wood, "A Functional Basis and Engineering Design: Reconciling and Evolving Previous Efforts," *Research in Engineering Design*, vol. 13, no. 2, pp. 65-82, 2002.
- [18] K.N. Otto and K. L. Wood, *Product Design Techniques in Reverse Engineering and New Product Development*. Upper Saddle River, NJ, United States of America: Prentice Hall, 2001.
- [19] C. R. Bryant, D. A. McAdams, and R. B. Stone, "A Validation Study of an Automated Concept Generator Design Tool," in *2006 ASME IDETC/CIE*, Philadelphia, PA, USA, 2006.
- [20] D. A. McAdams and K. Wood, "A Quantitative Similarity Metric for Design-by-Analogy," *Journal of Mechanical Design*, vol. 124, no. 2, pp. 173-182, 2002.
- [21] S. G. Arunajadai, S. J. Uder, R. B. Stone, and I. Y. Tumer, "Failure Mode Identification through Clustering Analysis," *Quality and Reliability Engineering International*, vol. 20, pp. 511-526, 2004.
- [22] R. B. Stone, I. Y. Tumer, and M. E. Stock, "Linking

- Product Functionality to Historic Failures to Improve Failure Analysis in Design," *Research in Engineering Design*, vol. 16, no. 2, pp. 96-108, 2005.
- [23] T. Kurtoglu, A. Swantner, and M. I. Campbell, "Automating the Conceptual Design Process: From Black Box to Component Selection," *Artificial Intelligence for Engineering Design Analysis & Manufacturing*, vol. 24, no. 1, pp. 49-62, 2010.
- [24] J. Schultz et al., "Limitations to Function Structures: A Case Study in Morphing Airfoil Design," in "ASME International Design Engineering Technical Conferences", vol. DTM, Montreal, Canada, 2010.
- [25] C. Sen, B. Caldwell, J. Summers, and G. Mocko, "Evaluation of the Functional Basis using an Information Theoretic Approach," *Artificial Intelligence in Engineering Design Analysis and Manufacturing*, vol. 24, no. 1, pp. 87-105, 2010.
- [26] David Gleich, MatlabBGL v4.0, 2008, Open-Source Software.
- [27] J. G. Siek, L.Q. Lee, and A. Lumsdaine, *Boost Graph Library, The: User Guide and Reference Manual*, 1st ed., B. Stroustrup, Ed.: Addison-Wesley Professional, 2002.
- [28] B. Widrow and M. E. Hoff, "Adaptive signal processing," in *1960 WESCON Convention Record, part IV*, 1960, pp. 96-104.
- [29] M.T. Hagan, H.B. Demuth, and M.H. Beale, *Neural Network Design*. Boston: PWS, 1996.
- [30] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," in *Parallel Data Processing, Vol. 1*, D.E. Rumelhart and J. McClelland, Eds. Cambridge, MA, United States of America: The M.I.T. Press, 1986, ch. 8, pp. 318-362.
- [31] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian regularization," in *Proceedings of the 1997 International Joint Conference on Neural Networks*, 1997, pp. 1930-1935.
- [32] D. J. C. Mackay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415-447, 1992.