The Introduction of a Design Heuristics Extraction Method

P C Matthews[a,*] L T M Blessing[b] K M Wallace[a]

[a]*Engineering Design Centre, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, United Kingdom*

[b]*Konstruktiontechnik und Entwicklungsmethodik, TU-Berlin, Straße des 17 juni 135, Berlin D-10623, Germany*

Abstract

This paper introduces a novel method for analyzing conceptual design data. Given a database of previous designs, this method identifies relationships between design components within this database. Further, the method transforms these relationships into explicit design knowledge that can be used to generate a 'heuristic-based' model of the design domain for use at the conceptual stage. This can be viewed as a knowledge extracting method for the conceptual design stage. Such a method is particularly interesting, as the conceptual stage typically lacks explicit models to describe the trade-offs that must be made when designing. The method uses either Principal Components Analysis or Self Organizing Maps to identify the relationships, and this paper describes all the elements required by the method to successfully extract and verify design knowledge from design databases.

Key words: Conceptual Design, Self Organizing Maps, Knowledge Extraction, Data Mining, Design Heuristics.

1 Introduction

In early stages of engineering design, designers have the greatest freedom to explore the solution space for a given problem. At the same time, designers wish to know if the lines of thought they are pursuing are worthwhile, or

* Corresponding author.
Email addresses: `pm131@eng.cam.ac.uk` (P C Matthews), `blessing@kt10.kf.tu-berlin.de` (L T M Blessing), `kmw@eng.cam.ac.uk` (K M Wallace).

if their efforts would be better spent following a different set of solutions. This requires an understanding of the complex relationships within the given design domain. At the conceptual design stage, this understanding can be in the form of underlying trends, existing solutions, trade-offs, or awareness of possible technological alternatives. These are used to explore the solution space for potentially feasible designs. Only the best solutions from this search should be taken forward for more detailed consideration.

The problem is that this understanding of the design domain is frequently the result of many years of experience within the domain. As a result, this knowledge tends not to be explicitly expressed, but rather resides tacitly in the form of a designer's experience. By assuming this knowledge is implicitly manifest within existing designs, this paper presents a novel method for extracting this knowledge and thereby making it explicit. However, this knowledge is likely to be quite coarse and hence the method is referred to as the Heuristics Extraction Method (HEM), thereby implying that this knowledge does not *guarantee* good results, but improves the chances of doing so.

The HEM is based on recognizing patterns in existing solutions. This pattern recognition triggers a first coarse model of the design space, providing a link from the design parameter space, the space describing the design concepts for a particular family of designs, and the evaluation space, the space describing possible evaluations for the design family. A neural network could be trained to learn these relationships and thus provide such a mapping. The initial aim was to use this mapping to support evaluation of new concepts. A new concept, defined in parameter space, could then be mapped into the evaluation space, thus providing a possible evaluation. However, designers in industry expressed that they could not trust such as system as they did not know the rules that had been applied and hence that would justify the evaluation results.

The aim then became to extract relationships that are implicit within trained neural networks (i.e. in the design concepts) to explicitly provide designers with domain knowledge that is useful for evaluation and generation of concepts. These relationships are expressed either in prose, e.g. 'As efficiency tends to 100%, there is a sudden large rise in EINO_x, hence, low efficiency implies low EINO_x', or as an algebraic expression, relating a number of design components. A *component* is defined in this paper as an element of the design description, e.g. the diameter of a hole, number of wheels, or the measurement of an evaluation criterion.[1] For example, a hollow cylinder design space can be described using the length (ℓ) and radius (r) and these can be combined to represent the mass, $m = C_1 \ell r + C_2 r^2$. Such an algebraic relationship is also frequently referred to as a *feature*.

[1] the use of the term *component* follows the terminology used in the neural network community, as opposed to the mechanical engineering community

The remainder of the paper is structured as follows: Section 2 reviews related work that is used by the analysis method; Section 4 describes the properties of design domains that are particularly suited to this method; Sections 5–9 describe the various elements of the heuristics extraction element of the method. Section 10 provides some empirical evidence on the success of the HEM. Finally, Sections 11 and 12 provide a discussion of the potential of the HEM and conclusions to the paper. The future work that is being undertaken is given in Section 13.

2 Data Analysis Methods

The nature of the design tasks carried out during the early design stages are unstructured; varied in type of solution; and with few, coarse details. During this part of the design process the designer should not be constrained in any manner that unduly reduces the solution space, e.g. by limiting creativity. This creates a challenge for producing a computational tool for this activity. A computational tool can be considered as the encoding of a method. For this tool to be useful, it must also have some rules or algorithms to help the users with their tasks. However, using such rules is in contradiction with the requirement that there be few constraints on designers. As a result, the aim is to explore the possibilities of automatically generating a non-restrictive model for the early design stages. As the relationships between elements of different design domains differ, these models will be different for each domain, so the method of generating them needs to be portable across domains. Finally, as the model needs to be non-restrictive, it should be viewed more as a method of guiding designers in the direction of better solutions when help is needed.

The approach taken in this research for extracting such a coarse model of the design domain has its origins in rule extraction and rule induction methods. These methods frequently are based on identifying class boundaries within a sample distribution and then expressing this boundary explicitly. There are a number of algorithms designed to partition a domain based on decision trees. These decision trees form (symbolic) rules about how to classify elements in a domain. An example of such a rule based induction method is ID3 (Rich and Knight, 1991; Luger, 1994). This aims to minimize the complexity of the tree it generates, thereby maximizing the information contained in each decision node. However, when the case space is large, the decision trees tend to get quite complicated and difficult for human interpretation.

A further example of using symbolic machine learning techniques for extracting rules is given by Reich and Travitzky (1995). This example generated a set of rules describing the corrosion properties for a set of materials. However, these were typically IF. . . THEN type rules, which can only separate the domain parallel to the component axes, that is, the domain is separated according to a single component at a time. More complex rules could produce general linear separations of the domain. These can only be successful in the event that the domain is known, or is likely to be, separable in such a manner. This approach will be of be little use where the separation rules are non-linear.

Another approach attempts to generate rules describing the distribution based on a neural network that has been trained on (and verified on) the distribution. Tickle et al. (1998) have devised a taxonomy for classifying rule extraction methods from artificial neural networks. This taxonomy has been applied to a cross section of neural network architectures. The aim of their work was not only rule extraction, but also rule initialization (where specific rules are inserted into a neural network) and rule refinement (where rule networks are 'tweaked' so that particular rules are slightly modified). The complexity of rules extracted from neural networks tends to provide a challenge when they are to be used by a human. Efforts have been made to optimize the extracted rules for parsimony so that they constitute a 'better' explanation of the distribution learnt by the neural network for a human reader (Corbett-Clark and Tarassenko, 1997; Corbett-Clark, 1998). Ultsch (1993) makes an initial attempt at extracting rules from Self Organizing Maps (SOM), by examining where the feature map needs to 'stretch' to cover the distribution. This suffers from only examining the overall structure of the SOM, rather than looking at individual components. This prevents the identification of relationships that occur between specific pairs of components that are too small to affect the overall SOM structure.

The above methods typically require large sample distributions to provide good results. For this research, two computational methods have been used to analyze design databases: Principal Components Analysis (PCA), a 'traditional' statistical procedure (Diamantaras and Kung, 1996); and Self Organizing Maps (SOM), a 'neural' computational method inspired by the structure of the biological brain (Kohonen, 1997). Both of these methods provide stable results with considerably smaller databases than required for traditional data-mining techniques. On overview of these will be given in Section 7.

3 Representation Issues

The aim of this research is to learn about relationships in a given design domain. The assumption is that designs and their evaluations can be parametrically represented, that is, as a set of attribute-value pairs. Due to the nature of conceptual design, these attributes are unlikely to be determined with great accuracy and might have to represent broad decisions which will require further detailing later in the design process. This also implies that the design and evaluation parameters/attributes are not necessarily of a quantitative nature.

3.1 Design space representation

Based on the above assumption, in the proposed HEM designs are represented in a vector format. This vector contains both the physical description and the evaluation parameters for a concept. The evaluations are either result from building the object and taking measurements, or they are estimations based on a model developed previously. There are 5 types of vector components:

(1) continuous valued (e.g. length measurement);
(2) ordinal (e.g. number of wheels);
(3) boolean (i.e. True/False);
(4) fuzzy valued; and
(5) 1-of-k (e.g. encoding a vehicle's driving wheels as one of 'front-wheel drive', 'rear-wheel drive', or 'all-wheel drive')

This representation permits 'coarser' data to be captured, which is useful as the design is only at the concept stage. Capturing designs using this format has the advantage of being readily usable by various computational methods.

The vector components must be determined prior to using the HEM. In first instance, a 'superset' of components should be captured. These are then processed to determine if any of the vector components are redundant. In this way, the set of most relevant components can be obtained (Matthews et al., 2000). For example, in some specific design domain, the color of an object may have no effect on its cost (and the remainder of the design). Initially color would be included in the component super-set, but would then be found to be redundant and therefore removed from further processing.

The following section describes an alternative means of describing the design space. This description is based on algebraic relationships between design and evaluation components, and hence these relationships are continuous. The purpose of such a representation is that these relationships explicitly express properties of the design space.

3.2 Feature Space

As seen in the previous sections, a simple way of describing multivariate objects parametrically is to express each measurable component by its value. However, for each distribution of objects (e.g. a particular design domain), there will be combinations of these components that represent relationships within the design. This representation has two advantages: firstly it reveals structure within the design space, and secondly it can provide a means for reducing the dimensionality of the design space which in turn will result in a faster and more accurate processing later. For example, consider the domain of similarly proportioned boxes: these can be described parametrically by (w, d, h, m, ρ); where w is width, d is depth, h is height, m is mass, and ρ is the planar density of the material used to construct the box. As all boxes in this domain are similarly proportioned, the three dimension components are linearly related:

$$\frac{w}{\alpha} = \frac{d}{\beta} = \frac{h}{\gamma} \tag{1}$$

These three components are combined linearly to form a feature, which in this case will represent the 'size' of the box:

$$f_1 = a_1 w + a_2 d + a_3 h \tag{2}$$

where (a_1, a_2, a_3) are chosen to 'maximize' the variation of f_1 with the variation of the box dimension components (which in this case will be the direction cosines of the line described by the box dimensions, given by (α, β, γ) in Equation 1). Similarly, the mass, m, of such a box is proportional to the surface area of the box:

$$m = 2(wd + dh + hw)\rho \tag{3}$$

Combining Equations 1, 2, and 3 a second (non-linear) feature ('weight') is produced, expressed in terms of ρ and the first feature:

$$f_2 = a_4 \rho f_1^2 \tag{4}$$

where a_4 is some constant of proportionality. These two features (f_1, f_2) form a two-dimensional description of the boxes that is equivalent to the 5-dimensional parametric description. In addition, the feature description represents the objects in what could be considered a more natural manner: namely a size component and weight component. This illustration has demonstrated both linear (Equation 2) and non-linear (Equation 4) features.

4 Relevant Design Domains

This research has developed methods for extracting knowledge to provide support within the early stages of the design process.

The HEM identifies relationships between design components. Therefore, this method is particularly useful for design domains where there are few *explicitly* known relationships. These domains will not have explicit models that can be used during the conceptual design stage or that are difficult to analyze analytically (e.g. designs requiring extensive computational fluid dynamics for evaluations). However, there must be a number of prior designs, complete with evaluation data, available to train the system with. It is assumed that this data is costly, either in terms of material or computational resources, and therefore will not necessarily be abundant. Also, the method can handle missing observations from within the data, which is advantageous as occasional measurement can be missing from designs.

For the purpose of validating the method, relatively mature domains were chosen and hence domain experts had a degree of understanding of these domains. These domains relied on empirical studies to build up their explicit knowledge bases. The aim was to determine if the HEM could automatically extract relevant relationships from a database of previous designs. It was therefore necessary for the domain to be known.

Three design domains were identified as case studies: one department based student design project and two aerospace projects. The department based case study was used to develop the representation used to encode the design space. The first aerospace project was the analysis of the preliminary design of gas turbine combustors. This case study aimed to examine the use of the HEM in a domain where measurements were missing from some of the examples used to train the system and where the relationships were unclear. The second aerospace case study examined preliminary wing design. This case study aimed to extract more complex relationships within the domain by incorporating novel data preprocessing techniques into the HEM.

The case studies provide illustrations for properties of design domains that determine how successful this HEM will be. The next sections describe the heuristics extraction method, and also provide further details as to what design domains are appropriate and inappropriate for the application of this method.

5 HEM Overview

The Heuristics Extraction Method consists of the full process from the representation of the conceptual design space through to the generation and verification of the relationships extracted from the sample design distribution. This represents not only the required computational elements, but also interfacing with domain experts.

Insert Figure 1 here

The overall structure of this method consists of five steps:

(1) Design concept space parameterization;
(2) Data collection, and if necessary preprocessing;
(3) Training;
(4) Post-processing (heuristic generation); and
(5) Verification of heuristics by domain experts.

The interaction with domain experts is an important part of this process. Without this, the extraction method might identify relationships that are of little or no use to designers. Domain experts are very important during the initial design space parameterization stage, as they are likely to be aware of which design parameters do play important roles in the design process. They can also help direct the method's searching by highlighting parameters they feel are particularly important.

The individual steps taken by the HEM are expanded below.

6 Data Preparation

The HEM is a data driven method, and therefore relies on the data collection stage prior to processing. This requires an initial design space analysis stage, where the conceptual design space is described parametrically. Once the design space has been parameterized, it must then be populated with examples. These are typically taken from collections of previous designs. Once such a database is available, a preprocessing algorithm is used to help the HEM identify more complex relationships, and a pruning algorithm to reduce the computational complexity incurred due to including irrelevant components. An overview of the computation element of the HEM is given in Figure 2.

Insert Figure 2 here

6.1 Design concept parameterization

Parameterizing design concepts appears initially to be contrary to the aims of conceptual design where the designer should not be restricted in any manner. However, when designing within a family of designs, there are a series of decisions that are prescribed (Matthews, 1998; Matthews et al., 2000). The outcome of these decisions can be parameterized, even if the outcomes are not numerical values (Ball et al., 1998). The aim of the parameterization is to capture a 'super-set' of design decisions, which will be pruned down at a later stage.

The parameterization is determined by examining all previous designs. These designs are stripped of details that can or need only be determined at a much more mature state of the design (e.g. wiring or pipe routing details, stress or other finite element details) and parameters that are known to have little or no effect on the evaluation (e.g. the color of the body work). From the remaining design details, parameters are grouped into classes: those that are constant throughout the family (e.g. external dimensions of a series of gas turbine combustors designed for a particular aero engine), and those that change within the family (and hence distinguish different designs). The class of constant parameters is then also discarded, as these parameters offer no information for distinguishing designs. The remaining parameters form the 'conceptual description' for the designs within this given family. This conceptual description is then augmented with the technical and any other relevant, e.g. aesthetic, evaluations for that design family.

6.2 Data collection

Once a conceptual description, including evaluation criteria, has been generated, all previous examples need to be encoded according to it. This forms a set of vectors that will be used to train the system (the 'training distribution'). In the event of a design lacking the measurement of a particular parameter, this parameter is left marked as unknown. It will then be up to the particular training algorithm to deal with this situation.

This is a relatively straightforward step. Provided that designs do not undergo large changes between final design concepts and final product, any changes that do occur can be considered as a small noise component. Such noise typically does not have a large overall effect on the final outcome.

6.3 Data preprocessing

The data collection stage covered the steps required to transform a set of previous designs into a numerical representation that can be processed by either the PCA algorithm or the SOM algorithm. To make this a *useful* representation, there must be intrinsic relationships that can be identified by either PCA or SOM. There are design representations where this is not the case. For example, consider the design of a rod: a possible parameterization for this would be the start and end points in space (a, b), and the mass of the rod ($m = \rho|a - b|$, where ρ is the linear density). The set of previous designs consists of a collection of random (a, b) values and appropriate m values. The difficulty here lies in that the methods investigated (PCA and SOM) can only learn relationships between pairs of components. In this case, there are no relationships between any pair of components and the training will not stabilize. However, if the training set used the parameterization $(|a-b|, m)$, a clear correlation emerges.

This example highlights the importance of generating a useful description of the design family. However, there are cases when it might not be possible to form a useful description in the first instance. In some data representations the relationships between components do not occur in pairs but rather in larger groupings, e.g. triples. The need for preprocessing is identified after an initial (exploratory) training iteration. Three ways of identifying the need for preprocessing are:

(1) components that are expected to play an important role (e.g. by a domain expert) are not being included in the final set of heuristics;
(2) there are far fewer heuristics than expected;
(3) training on different subsets of the original data yields little or no stability in the heuristics generated, or in intermediate steps.

Of these criteria, the first two require a domain expert to review the initial set of heuristics generated. The third criterion tests how stable the algorithm's results are, if different portions of the training data are used (e.g. 'k-fold cross verification'). This can be seen by examining the difference in the linear features generated by PCA or the component maps generated by the SOM. It effectively identifies 'noisy' components within the data. Assuming that if a set of components is believed to be useful, but appears only as random noise to either PCA or SOM, then these components might need to be combined in some manner to generate useful information about the design (although this information might not be directly useful to a designer) such that the PCA and SOM can stabilize.

In the first instance, components should be combined algebraically using whatever prior domain knowledge there is (for example, a set of related measurements might be able to be combined usefully by adding or subtracting them, as illustrated earlier). If this does not produce satisfactory results, components can be combined with a 'brute-force' method, that is testing all possible combinations. The basic algebraic operations (addition, subtraction, multiplication, and division) are applied to each pair of components resulting in 4 more components for each original component-pair (Matthews, 2001). Finally, in the event that this option is exhausted, it is possible that the design space is too chaotic within its defined boundary, and the initial assumption about the continuity of the mapping between design components is insufficiently true for this method to generate heuristics successfully. In this event, either important design components are being omitted in the representation or there are no continuous relationships between the components. In both cases, this heuristics extraction method cannot be applied.

7 Core Technology

The basic analysis algorithm, or ‘core technology’, is some form of numerical correlation. This core technology takes as input the (possibly) preprocessed design data, and performs some analysis or re-representation of this data. For comparison purposes, two data analysis algorithms were used: Principal Components Analysis (PCA) and Self Organizing Maps (SOM).

This section will first state the necessary properties a design domain must have for this to be successfully analyzed by this method. Following this, a brief overview will given to the backgrounds of the two algorithms used. Subsequent sections will then describe how the HEM implements these technologies as part of the overall analysis method.

7.1 Space assumptions

There are some basic assumptions that need to be made about the properties of the design space. As described in Section 6.3, designs are represented as a real-valued vector. If there are a total of N components to this design vector, the design space, $\mathbf{X}$, can be embedded into $\mathbb{R}^N$. Not all elements of $\mathbb{R}^N$ will be legal designs (for example, a negative length component could be represented in $\mathbb{R}^N$, but would not be a member of $\mathbf{X}$). This can be algebraically represented as $\mathbf{X} \subset \mathbb{R}^N$. It shall be assumed that $\mathbf{X}$ forms a piecewise connected set, that is for any design $\mathbf{x} \in \mathbf{X}$, there exists a small region centered around $\mathbf{x}$ that lies fully within $\mathbf{X}$. This assumption means that any design can be perturbed (modified) by a small amount, and still be a legal design (see Figure 3). Note that if the design space is the union of disjointed sets, i.e. $\mathbf{X} = \cup \mathbf{X}_i$, each $\mathbf{X}_i$ represents a cluster of similar designs.

Insert Figure 3 here

For practical matters, it will be necessary that any design sample distribution, generated from the database of previous designs, has sufficiently large ‘chunks’ in each cluster. This is because a region needs to be sufficiently large to be identified, and ideally, there needs to be several examples or elements (designs) from this region to be able to learn about designs in this area of the domain.

A key assumption needed is that the mapping between design parameter sub-space and evaluation sub-space is at least piecewise continuous. That is, for any point within the design parameter sub-space, the mapping to the evaluation sub-space will be continuous for at least a small region around the point in parameter space (so a small change in the design parameters will result in no more than a small change in the evaluation sub-space).

7.2 PCA overview

Principal Components Analysis (PCA) is a factor analysis type of statistical method for analysing multivariate data (Diamantaras and Kung, 1996). PCA identifies linear correlations within the data, and the variation of these correlations. The first component is the linear correlation that can describe the greatest variance within the dataset. These components are then ordered in descending magnitude of variation in a manner such that the new co-ordinate system remains orthogonal (see Figure 4). The new coordinate system, (f_1, f_2) in the figure, represents a set of features in decreasing order of importance (as determined by the component's variance within the dataset) as each new co-ordinate is a linear combination of the original variables, (x_1, x_2) in the figure. Hence the PCA based algorithm is a feature extraction method, albeit a simple one. This is usually used to re-structure the data representation in such a manner that the 'less' significant components can be identified and discarded, thereby representing the data in a more compact form. For example, the data in Figure 4 could be represented in one dimension by approximating it to a line (discarding the deviation component from this line, which could possibly be due to noise). With design data, this also provides a method of identifying which of the design parameters have a lesser effect on the design during the conceptual stage, and hence need not be precisely determined early on in the design process.

Insert Figure 4 here

For higher dimension spaces, let X represent the matrix of observations [2] (i.e. each column $\mathbf{x}_i$ represents observations for a particular parameter and the rows represent individual observations). First, the covariance matrix, $\mathrm{R} = (r_{ij})$, is constructed:

$$r_{ij} = \mathbf{E}[(x_i - \bar{x}_i)(x_j - \bar{x}_j)] \tag{5}$$

where $\mathbf{E}[\,\cdot\,]$ is the expectation operator and $\bar{x}_i$ is the mean of the i^{th} parameter. Note that by the construction of R, it is a real symmetric matrix. A property of real symmetric matrices is that they have real eigenvalues λ_i with associated eigenvectors $\mathbf{e}_i$ where $\mathbf{e}_i^{\mathrm{T}}\mathbf{e}_j = \delta_{ij}$, i.e. the eigenvectors for distinct eigenvalues are orthogonal. These eigenvectors represent the principal components of the data. Further, the size of eigenvalues gives an indication of how important the component is. It is therefore sensible to order the eigenvalues in descending order $(\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N)$.

The variable space can then be orthogonally transformed into feature space using the matrix spanned by the (normalized) eigenvectors:

[2] an observation is the numerical value for a parameter for a given design instance

$$\mathrm{Q} = (\mathbf{e}_1\,\mathbf{e}_2\;\cdots\;\mathbf{e}_N)^{\mathrm{T}} \tag{6}$$
$$\mathbf{f} = \mathrm{Q}\mathbf{x} \tag{7}$$

where $\mathbf{x}$ represents a vector in variable space, $\mathbf{f}$ represents a vector in feature space, and Q is the transformation matrix. This transformation matrix is a linear re-representation of the design domain. Section 8.1 describes how this matrix is analyzed to extract the design heuristics.

7.3 SOM overview

In its most simple interpretation, the Self Organizing Maps are an 'elastic network' of points fitted to some given distribution (Kohonen, 1997). This network has the interesting property that the network nodes (index vectors) maintain the global ordering with respect to the original distribution space, that is, there is no 'knotting' of the network. A knotted network would be troublesome as traveling in one direction along the network would not guarantee that the same point in design space would not be revisited. The topology preserving nature of this network results in the map produced forming a non-parametric model for the design space.

The SOM is an unsupervised learning algorithm: that is, it must learn the structure of the space without any prompts. The SOM is 'trained' by presenting it the sample data several times, with the SOM re-adjusting itself with each iteration. The SOM is made up of a set of M nodes each containing a codebook vector $\mathbf{w}_i$ that represents a point in the design space. Initially, these are set randomly. The data is then presented to the network in a random order. For each data sample ($\mathbf{x}$), a winning node c is the node whose codebook vector has the greatest similarity to the data vector, in this case determined by the Euclidean distance:

$$d(\mathbf{w}_c, \mathbf{x}) \leq d(\mathbf{w}_i, \mathbf{x}) \;\; \forall i \in \{1, \ldots, M\} \tag{8}$$

The codebook vector of this winning node is then adjusted so that its distance with this data vector would be decreased (where α is the 'learning rate' with $0 \leq \alpha \leq 1$):

$$\mathbf{w}_c' = \mathbf{w}_c - \alpha(\mathbf{w}_c - \mathbf{x}) \tag{9}$$

The neighboring nodes of c are also updated in a similar manner, but with a smaller value of α. This ensures that the neighborhood remains topologically equivalent to the neighborhood of $\mathbf{x}$ in variable space. This procedure is repeated for all data points, with α decreasing, until the codebook vectors stabilize. Once this is achieved, the SOM is said to be ordered (or trained).

Once the network has been ordered, the nodes can be considered to be 'classifiers' of their regions within the design space. When the network is given a design, the node that represents the area of domain that is most similar to this given design is activated. Due to the topological ordering of the network, moving to neighboring nodes represents moving to nearby regions within the design space. The network can be considered to represent features locally, that is, the network can be used to approximate the relationship between parameters for small regions represented by neighboring lattice points. In a similar manner to identifying and interpreting the linear features discovered by PCA, the SOM features will also need to be identified and interpreted by a domain expert.

8 Post-processing

The post-processing algorithms that identify potential relationships from the training data represent a novel method for knowledge elicitation and are a significant contribution arising from this research. Two significantly different post-processing algorithms are used to generate heuristics. This is due to the difference in the output of the PCA and SOM algorithms. The overall post-processing method involves the following steps: (1) analyze the output for relationships between components; (2) simplify these relationships by describing them in a concise manner; (3) check these relationships with the full sample distribution. The third step is required in the event the second step oversimplified a relationship to the extent that it is incorrect.

The next sections describe the methods for extracting heuristics from the PCA and SOM algorithms, and how these heuristics are checked against the sample distribution.

8.1 Principal Components Analysis

As described in Section 7.2, PCA is a statistical technique that identifies the linear dependence structure of a multivariate stochastic observation (Diamantaras and Kung, 1996). This dependence structure can be analyzed to extract a set of implicit linear equations that describe the relationships between design and evaluation parameters. These linear features are described as follows:

$$f_i = \sum_{j=1}^{N} e_{ij} x_j \qquad (10)$$

These features represent an orthogonal transformation of the axis (as described by Equation 7), and are too complex for extracting useful heuristics: so any terms with $|e_{ij}| < \Theta$, are removed for some threshold Θ. Hence, the simplified features are given by:

$$f'_i = \sum_{|e_{ij}| \geq \Theta} e_{ij} x_j \tag{11}$$

where Θ is chosen such that most features can be described using about 10 design parameters. Most of these simplified features can be given some physical interpretation. For example, consider the design of a small autonomous vehicle (see Section 10.1). Vehicles are characterized by five components: distance between sensor array and drive axle (d); drive wheel diameter (w); number of microswitches (n); construction quality, or 'duty' (D); and line following ability (L). Of these five components, the first three are design parameters (components that the designer can set) and the last two are evaluations (determined as a result of the design parameters). Using data from the case study, the first linear feature is given by:

$$f_1 = 0.9996d + 0.0016w - 0.0018n + 0.0139D + 0.0225L$$

Simplifying this by setting $\Theta = 0.01$ results in:

$$f'_1 = 0.9996d + 0.0139D + 0.0225L$$

This feature indicates that the largest variation occurs with d, the distance between the IR sensor array and the drive axle; and that increasing d increases the duty score (D) and improves line following (L). The remaining two components (wheel size and number of microswitches) appear to have little variation with respect to variation in d, and hence are removed.

These features need to be examined individually to extract heuristics. Recall that the factors, e_{ij}, are the eigenvector components of the covariance matrix generated from the training distribution. Hence, in the simplified features, f'_i, these will represent components that are either strongly correlated or anti-correlated (depending on the sign). This linear scaling permits us to predict how changing one component in the feature is *likely* to change the others and by what amount.

The features can be grouped into three categories: features containing only design parameters; features containing only evaluation parameters; and features containing both design and evaluation parameters. The features containing only design parameters indicate how various design parameters are related, and potentially provide an indication of how novel combinations could be made by breaking such relationships. The features containing only evaluations indicates how various evaluations are related (and therefore suggest trade-offs that might need to be made). Finally, the features containing both design components and evaluation components indicate how those designs and evaluations are directly related.

The limitation of this method is that PCA generates a set of orthogonal linear features. However, the actual design features are not necessarily either orthogonal or linear. Therefore it is desirable to use a non-linear method to produce further results.

8.2 Self Organizing Maps

SOMs provide a simpler (low dimension) representation of a distribution in a high dimensional space in a manner that ensures that the *topology* of the distribution is maintained. From this simplified representation, the aim is to extract knowledge about the distribution (i.e. heuristics about the particular design domain). The local non-linear correlation algorithm described in Section 7.3 provides the basis for the knowledge extraction method.

The SOM generates a two dimensional map of the distribution using a regular lattice of nodes. These nodes are representative points within the distribution. The advantage of using a two dimensional map is that the distribution can be easily visualized. This is accomplished by plotting the values of each component of the distribution on its own 2D map (the value being represented by shading). Components are correlated when their respective maps visually have 'similar appearances' (Figure 5(a)). Further, components can be locally correlated if they both have an area in which they appear similar, when globally they are dissimilar (Figure 5(b)).

Anti-correlations (Figure 5(c)) are identified using an edge detection technique, based on the gradients of the component maps. Due to the small size of the component maps, this is a coarse approximation to the true gradient (or the edges are difficult to identify, see Figure 5(d)), and so results from this method tend to be weaker. Both correlations and anti-correlations are identified here.

Insert Figure 5 here

Because of the large number of pairwise comparisons that have to be made, clustering methods are used to identify smaller groups of component maps that are worth more detailed comparison. Clustering is performed based on the 'visual' similarity between maps, measured using a Tanimoto metric and stored in a matrix (Matthews, 2001). The $(i, j)^{\text{th}}$ element of this matrix represents the similarity between component maps i and j. Originally, this matrix was rearranged so that similar components were placed together in blocks, which is a similar clustering procedure to that of Ling (1973). Figure 6 is a graphical representation of the re-arranged similarity matrix. The two axes represent the component indices, with the shading of each cell representing the value of the corresponding matrix entry where black is no similarity ranging through to white which represents complete similarity. Elements that are below a given threshold are reset to zero. The blocks are similar to the features generated by the PCA method, except that the similarity measure is qualitative rather than quantitative as is the case with the factors generated with PCA. However, the main purpose is to generate small groups of components that demonstrate similar characteristics. It is worth noting that this method only discovers positive correlations, as it identifies similarity in matching values. As mentioned earlier, anti-correlations can be detected using the same method on the second order method (edge detection) matrices.

Insert Figure 6 here

A shortcoming of using the re-arranged similarity matrix is that it is difficult to generate accurate groups of similar components. This is due to not being able to generate mutually exclusive sets of components. From Figure 6, it can be seen that the group structure is more complex than a set of mutually exclusive sets, namely that there are overlapping regions.

There are various clustering techniques available, most of these generating hierarchical partitions (Kaufman and Rousseeuw, 1990; Matthews et al., 2001). One non-hierarchical method was mentioned (Jardine and Sibson, 1968), and this was implemented to generate 'clumps' of components. These clumps represent a more natural grouping of components as each component can be in a number of clumps, which can represent the (partial) similarity a component has to each of these clumps. This is illustrated in Figure 7. On the left of this figure, is an example of the shape a typical re-ordered similarity matrix will have. Next to this, is a possible hierarchical representation of how the components could be structured. It should be noted that for the elements from the overlapping region ($\mathbf{A} \cap \mathbf{B}$), a decision must be made as to which group they shall belong to. A non-hierarchical clustering method does not have this problem, as it is possible for groups to overlap when necessary (as shown with the rightmost tree structure). However, it should be noted that the algorithm implemented is computationally inefficient (memory-wise), and there will be difficulties with analyzing high-dimensional distributions.

Insert Figure 7 here

8.3 Component pruning

As discussed in Section 6.1, when initially parameterizing the design space, the experimenter is encouraged to include more parameters than are believed to be necessary. This is good practice, as it is more difficult to identify missing parameters than to identify redundant ones. Also, the preprocessing algorithm introduces a large number of new combined components. However, over-parameterization of the design space can present a problem when training neural networks if coupled with relatively sparsely populated data training points, e.g. when there are only a couple of data points per dimension. It is therefore desirable to reduce the dimensionality of the input data. The redundancy identification (pruning) method can be summarized as identifying components with little effect on the remainder of the design space.

A simple method to determine the number of components to be pruned is by using the eigenvalues from PCA. As the eigenvalues decrease, the importance of the associated feature (and hence dimension in the feature space) decreases. Therefore, a threshold is set and the linear features with associated eigenvalues below this threshold are discarded. This determines the *intrinsic dimensionality* of the design space, n_i, and this number of original components is selected.

The SOM is used to identify component pairs that behave similarly, based on the component maps. Component maps that have little similarity to any other can be assumed to represent redundant components, as these are not involved in any relationships and therefore do not reveal any structure of the domain (Matthews, 2001). This redundancy becomes clearer when the components are grouped using a non-hierarchical clustering method, described in Section 8.2. These redundant components will only occur in singleton groups.

It must be noted when pruning components that these components might not be totally redundant: they might be components that need to be combined with others in a preprocessing phase to identify meaningful relationships within the design space. Therefore, it is recommended that before any components are discarded, a domain expert should confirm that these components are not likely to play an important role.

9 Generating and Evaluating Heuristics

The previous two sections involved the numerical processing of design data and identifying sets of components that have some form of relationship between them. This section describes how this information is interpreted to generate linguistic (or algebraic, where appropriate) heuristics that can be readily used by designers.

9.1 Principal Components Analysis

The final result of the PCA algorithm was a set of simplified linear relationships between components (recall Equation 11). It was argued that these features represent design characteristics. Hence, these features can be transformed into linguistic heuristics by describing the trends or correlations encoded by the linear relationship. Attention must be paid to the orthogonality of the linear features, namely, if two features appear to contradict each other the more important (according to the associated eigenvalue) should be used.

9.2 Self Organizing Maps

Analyzing the component similarity matrix results in a set of clumps of components that share similar behaviors. Each of these clumps is taken in turn. For each clump, all the member component maps are plotted side-by-side (see Figure 5). These plots are visually inspected to determine the nature of the similar behavior, namely, is the behavior shared globally or does it only occur locally? If it occurs locally, what are the conditions for the shared behavior? These relationships are expressed in a simple manner that reflects the component maps (e.g. Figure 5(b) would be High EICO *implies* High EINO_x). Under certain conditions, the Tanimoto metric can falsely identify two maps as similar. This occurs when one map is mainly covered by average values and the other map has a significant number of its nodes at extreme values, thereby causing the averages to be similar. An example of such a pair of maps is given in Figure 8. This highlights the need to visually compare the component maps to filter out such examples.

Insert Figure 8 here

9.3 Checking against training data

The interpretation of the PCA and SOM outputs involves a ‘smoothing’ or simplification process. This process can result in the over-simplification of some relationships to the degree that they are no longer correct. Hence, it is necessary to check the relationships generated against the original training data.

The checking of the heuristics is achieved by plotting the components of the relationship against each other. These plots fall into three categories:

(1) Continuous valued against continuous valued: this can be verified by a scatter-plot (see Figures 9(a) and 9(b));

(2) Boolean against continuous valued: this can be verified using two frequency plots (see Figures 9(c) and 9(d));
(3) Boolean against boolean: this can be verified using two frequency plots (see Figures 9(e) and 9(f)).

Insert Figure 9 here

The first of these verification methods is a traditional scatter-plot. In the event that the points appear to lie on a curve, an exact relationship can often be found by investigating the curve that best fits through the points (see Figure 9(a)). If the scattering does not converge to any form of curve, it can be inferred that there is no direct relationship between the two given components (see Figure 9(b)).

Boolean components can arise from two types of component: either an originally boolean component (e.g. a member of a 1-of-k choice) or from a continuous component that is split into cases, e.g. component values below and above a certain threshold. A Boolean-continuous relationship will take the form of: if component x lies in a given range, component y is more likely to take the value y_0 otherwise y is more likely to take the value y_1. Note that this is a probabilistic relationship rather than a definite one. This is a characteristic of design heuristics: they do not hold in all cases, just a significant majority. The inferred relationship can be verified by plotting (1) the frequency of elements being classified as True according to the continuous component, and (2) the frequency of elements being classified as False according to the continuous component. If these two plots have most of their 'mass' at opposite ends of the continuous component's range, it can be inferred that this heuristic is correct (see Figure 9(c)); if the mass of both plots lie in the same place, it can be inferred that the heuristic is incorrect (see Figure 9(d)).

The Boolean-Boolean relationship can be considered as a restricted version of the Boolean-continuous relationship, namely the range has now been replaced with a True/False outcome space. In a similar manner to the Boolean-continuous, such relationships can be verified by being able to infer (with relatively high confidence) class membership (i.e. the one component) based on the condition of the second component (see Figures 9(e) and 9(f), for positive and negative examples respectively).

Heuristics that fail this self-verification stage are removed. The remaining heuristics are now known to be accurate as far as the supplied training distribution is concerned. The process of checking pairs of components might appear to make the use of PCA and SOM redundant, however, this checking process is considerably more expensive than generating the groups. The clustering greatly reduces the number of component pairs that will need checking.

It is not known how useful or novel these heuristics are, and therefore it is necessary to have a second verification round with a group of domain experts.

9.4 Expert verification of the HEM results

The final verification is done by a group of domain experts. This verification of the heuristics serves three purposes: (1) how accurate are the heuristics according to an expert's opinion; (2) how novel are the heuristics; and (3) how important are the heuristics. The first purpose aims to widen the context beyond the training distribution by using the expert's tacit domain knowledge. This is also important to prove the heuristic generating method is capable of identifying relationships accurately, and serves to validate the method. The second purpose aims to measure how 'interesting' the heuristics are. This is useful for targeting the heuristics to a particular expertise level, and ultimately can be used to identify novel relationships that have been discovered from the distribution. The third purpose is to gauge how important the heuristics are for the domain.

The heuristics are evaluated using a paper method: each heuristic is reported with an evaluation scoring table that is to be completed by the expert. The experts are also encouraged to comment on the heuristics. Once this has been completed, the heuristics that have scored well are kept. The heuristics that have scored poorly according to the domain experts require further investigation, as these relationships were verified as accurate with respect to the training distribution. After further consideration, they can either be rejected or included, in a modified form, in the final set.

Finally, this verification process provides an accuracy measure subjective to the experts' opinions. Therefore, it is possible that even where an expert believes there is no relationship, such a relationship does indeed exist.

Other methods exist for the purpose of evaluating machine learning, for example Arciszewski (1997) and Reich and Barai (1999). However, for the purposes of this research, such methods are not necessary as it is sufficient to demonstrate that valid heuristics have been identified and extracted from the training distribution.

10 Application of HEM to Case Studies

The scope of this research was to generate accurate relationships from a given design domain. Hence, the method was able to be validated through the verification of the accuracy of the outputs of the method applied to three case studies. The novelty and importance of the heuristics were also measured to gain an indication of how applicable the HEM would be for industrial uses. These case studies are reviewed, highlighting the conclusions that can be drawn from each.

10.1 Integrated Design Project

The Integrated Design Project (IDP) involves teams of six students (2 will become mechanical specialists, 2 electronic, and 2 software). These teams are required to design, build and test a semi-autonomous vehicle that is able to navigate a course (marked out by a white line) and perform various pallet handling tasks, usually of the format fetch-classify-place. Each vehicle is constructed from a fixed kit of parts (with no restrictions on bulk materials such as steel or wood) and must be completed in four weeks.

The IDP case study was based within the Department which permitted easy access to prior design examples, their evaluations, and domain experts. This case study represented a product that involved the interaction of different modules (electrical, software, and mechanical). Only the mechanical module was represented, as it was not possible to acquire previous design data, and hence model, the electrical and software elements. While these are important factors, the benefits of having both data and domain experts locally available made this a valuable case study.

The PCA method produced a set of orthogonal linear features which proved to be difficult to interpret into useful heuristics. The results generated from the SOM method were considerably simpler to transform into heuristics. Further, it was shown that 17 out of the 22 heuristics identified using the PCA method were also identified using the SOM.

A total of 42 heuristics was generated. The heuristic accuracy verification resulted in a total of 12 (28.6%) scoring as highly accurate, 13 (31.0%) scoring as moderately accurate, and 15 (35.7%) scoring as not very accurate (leaving 2 (4.8%) undecided). The novelty scores for these heuristics were roughly opposite to the accuracy scores and the importance scores roughly followed the accuracy scores. One of the shortcomings of this case study was the lack of inclusion of the electrical and software modules. These interact with the mechanical module, and can have either a positive or negative influence on the design's final performance.

10.2 Combustor

The combustor provides a core function of an aircraft gas turbine engine. Each combustor design represents a very costly development, and hence there is the desire to maximize the amount of information extracted from combustor rig tests. Due to this expense, there were only a few examples to analyze. Further, some of the examples had measurements missing. It is also known that the combustor domain is non-linear, and hence that linear methods will not be very successful. These properties hindered the use of the PCA method, as it requires all measurements to be present. The SOM on the other hand, can be trained with incomplete datasets.

Nevertheless, both PCA and SOM were used to process the data. To use the PCA, the data elements with missing measurements were removed leaving 79 data points for 37 dimensions: i.e. about 2 data points per dimension. The linear features were compared to the heuristics generated from the SOM. A total of 24 (51.1%) of the heuristics generated using the SOM could not have be identified in the PCA results. The remaining heuristics that were potentially identifiable from the PCA processing would require greater effort to be identified. This is because the SOM post-processing reports potential relationships in a manner that is considerably easier to check. A rapid initial checking is possible by comparing the relevant SOM component maps side-by-side prior to further testing. There is no parallel method for rapidly checking small sets of components for potential relationships with the PCA. No heuristics were identified from the PCA processing that had not been identified using the SOM.

A total of 47 heuristics was generated. The heuristic verification resulted in 27 (57.4%) scoring as highly accurate, 6 (12.8%) scoring as moderately accurate, and 11 (23.4%) scoring as not very accurate (leaving 3 (6.4%) undecided). Although a large portion of these heuristics did not score highly on novelty, this case study demonstrated the potential of using the heuristics extraction method as a means of generating a coarse design model for a given domain.

10.3 Wing

The wing case study represented the second industrial design project. This differed from the combustor design domain in two ways. Firstly, the wing designs used were a result of a genetic algorithm search of the design space rather than actual wing designs. Secondly, the design space was represented by a small number (13) of components in comparison to both the IDP (84) and the combustor (37). A direct application of both PCA and SOM on this dataset revealed no potential relationships. Hence, it was necessary to perform the additional preprocessing step in the method.

A total of 25 heuristics was generated. These were represented as trade-offs that would be expected in the design space and were phrased as algebraic proportionalities, i.e. omitting all constants and coefficients, for example one of the heuristics takes the form of $A - B \propto C \times D$ where A, B, C, D represent wing design components. However, the domain experts were unable to verify the heuristics for a two reasons. Firstly, the representation of the relationships was difficult to follow and most of the relationships were not considered to be relevant. This can be addressed by improving the formatting of the heuristics to be less algebraic and to flag design components that the heuristics extraction method should focus on. The second difficulty was that the wing design space was not fully encoded. This resulted in the identification of overly general relationships that were thought to be of little use to designers. More specific relationships that take design context into account are needed.

A self verification method was developed to provide some measure of the accuracy of the heuristics by measuring the 'goodness' of each heuristic. This was achieved by measuring the variance of the data about the relation and the overlap of these variances at the extreme ranges of the relationship. The aim was to provide some means of scoring the heuristics that would be comparable to the other two case studies. This scoring was possible as all the heuristics were algebraic type relationships. However, no comment or measurement could be provided on either the novelty or importance of these heuristics as these can only be supplied by domain experts. This self-verification resulted in 8 (32%) heuristics being scored as highly accurate, 7 (28%) scoring as moderately accurate, and 8 (32%) scoring as not very accurate (leaving 2 (8%) undecided).

11 Discussion

This paper has described the research and development of a novel design data analysis method aimed at the generation of heuristics. This heuristics extraction method used PCA and SOM at its core to analyze design data. These techniques required the assumption that there was some form of piecewise continuous mapping between the design components (as discussed in Section 7.1). This was wrapped by a methodology for collecting and preparing the design data for the two numerical techniques. A method was developed to interpret the results of these numerical techniques into relevant design relationships. These relationships were then re-represented in the form of 'design heuristics', with the aim of verifying the relationships using domain experts. There were also developments for interfacing between the major elements of the HEM. These were the preprocessing and filtering ('component pruning') algorithms prior to the core processing, and the clustering method used to rapidly identify groups of components sharing similar behavior (recall the overall structure as shown in Figure 2).

In a more abstract sense, the HEM generates a set of 'interesting questions' to ask a domain expert. The full process of analyzing the design data, generating the relationships that form the basis of the questions, and using the experts' answers forms the heuristic extraction method. The main challenge this method faces is ensuring that the design space is sufficiently well described. It was demonstrated that over-described design spaces could be reduced to the relevant set of design components, but design spaces where important components were missing were unlikely to provide useful results. The case studies investigated in this paper illustrated these issues.

11.1 Exploitation

Once the heuristics have been extracted, there are several uses for them. The most immediate is to inform designers of implicit relationships that exist between components in the design domain. Although there has been expert intervention, this method is useful for four reasons: (1) the expert is given a set of relationships (without explanation) rather than having to generate these; (2) the expert might not be consciously aware of some of the relationships and would not expressly write these down if asked; (3) some of the relationships might be totally novel, and therefore unknown; and (4) these explicit relationships can be used by novices to learn about the design domain. This represents a relatively efficient use of an expert's time for documenting and acquiring knowledge about a design domain, as it only requires a single 'interview' with the expert. Other methods typically require several interviews (for example, the SPEDE method described in Ahmed (2001) requires between 5 and 12).

The heuristics can also be used to identify regions in the design space that have not been explored. Such regions can be identified where there appear to be relationships between components with no good reason, resulting in an extra (implicit) constraint on the design space. Once these have been identified, designers can either experiment in this region or rationalize the constraints, which again leads to greater understanding of the design domain.

One final use of the heuristics is *trend identification*. This is where the relationships are extended beyond the training distribution to generate estimates of how a design family would behave if their design parameters were set beyond the range of what has currently been tried. It should be noted that this can be risky, as the relationships identified have only been verified within the training distribution, and there might be some physical phenomena or constraints that prevent the design distribution extending in this manner.

The development of the Heuristics Extraction Method has provided a novel methodology for extracting design knowledge from databases of previous designs. This methodology includes a method for presenting the computational results to design experts, and using this process to focus their attention on potentially unknown relationships that occur within the given design domain. This analysis method potentially impacts the manner which empirical knowledge is gathered, by automatically generating hypotheses (i.e. the relationships identified). Researchers will then be able to focus on the interesting relationships and further test them, either empirically or analytically.

11.2 HEM Requirements and Limitations

The HEM has a number of requirements which in turn imposes certain limitations. These were given in Section 3.1 as a set of abstract mathematical properties that a design space must have for the HEM to successfully identify relationships. In design terms, these requirements state that:

(1) the designs used to train the HEM must all come from the same family, i.e. a single representation can be used to describe all designs of this family;
(2) there must be some continuity in this space, i.e. designs can be modified by a small amount with no more than a small effect on the overall design; and
(3) the design space must have been partially explored in some manner to provide the training data for the HEM.

The three case studies satisfied these requirements. Each case study had a representation that covered all design examples. From these representations, continuous relationships existed between various components, even though these were contained in more complex relationships in the wing case study. Finally, each case study was based on a database of previous examples.

Of the above requirements, the continuity requirement provides the greatest limitation. The HEM is not able to generate heuristics relating to components that do not have continuous relationships with other design components. For example, designs where the placement of some resonating element is used would not have the continuity property: e.g. the element needs to be located *exactly* for resonance to occur. This breaks the continuity requirement as a small change in the position results in a large change in the overall design. This is, for example, a property of designs where the reduction of audible noise is an issue. Hence, it is not expected to be possible to analyze such domains with the HEM.

12 Conclusions

The primary aim was to develop a method for extracting design heuristics from a given set of previous design examples. This was achieved by identifying relationships between various design components, which could be either design parameters or evaluations. The overall heuristics extraction method used two different data analysis algorithms at its core: Principal Components Analysis and Self Organizing Maps. The post-processing of the results of these two methods provided a novel means for identifying these heuristics. Two methods were used to provide a comparison between the mature PCA method and the more recent SOM. This demonstrated how the SOM's flexibility could identify more general relationships than PCA linear analysis.

This research aimed to extract *accurate* relationships from a given design domain by analyzing prior design examples. The three case studies demonstrated this accuracy, and through this it can be inferred that the method developed has reasonable validity for this purpose.

Two main contributions were provided for the mechanical design domain. The first was a novel means of analyzing design domains using databases of previous examples from the domain. This analysis method provides designers with a set of explicit heuristic-based relationships that provide greater understanding of the domain. The second contribution arises from the first: the relationships generated reflect any implicit rules or patterns designers follow, even if they are not aware of doing so. In following such patterns, designers restrict their search space. By highlighting such behavior, designers can ensure they escape from such patterns and perform a more complete search of the design space. Overall, the extracted heuristics provide an explicit description of the behavior of the design space. This can be used by industry to help better understand the design relationships of a current product line.

13 Future Work

As a result of this work, various aspects of the algorithm and technique have been identified as requiring greater attention. These aspects can be divided into the three main computational elements of the HEM: the preprocessing, the numerical processing method, and the post-processing.

The preprocessing method could benefit from investigation into more sophisticated methods to explore the relationship space, as opposed to a 'brute-force' recombination. A potential method being investigated is the use of genetic programming techniques for searching the relationship space.

This paper describes two specific numerical processing methods (PCA and SOM). It would be beneficial to investigate other such methods. Work is planned on experimenting with using larger lattice dimensions for the SOM. While this has the advantage of being able to more closely represent the design space, it will no longer be possible to generate two dimensional projection maps, and therefore visualize the design space. This will require a more accurate similarity metric to replace the Tanimoto metric currently being used.

Finally, the post-processing methods need to be improved. The non-hierarchical clustering method implemented is too inefficient for very high dimensional spaces. This clustering method is being reviewed to reduce the overheads it incurs. Further, the method with which the final heuristics are generated and presented to domain experts is also being reviewed, based on the results of the wing case study.

Acknowledgments

This work was fully funded by the *University Technology Partnership*, a collaborative research project between the universities of Cambridge, Sheffield and Southampton; and with industrial partners BAE SYSTEMS and Rolls-Royce.

References

Ahmed, S. (2001). *Understanding the Use and Reuse of Experience in Engineering Design*, PhD thesis, Cambridge University Engineering Department.

Arciszewski, T. (1997). Engineering semantic evaluation of decision rules, *Journal of Intelligent and Fuzzy Systems* **5**: 285–295.

Ball, N. R., Matthews, P. C. and Wallace, K. M. (1998). Managing conceptual design objects: An alternative to geometry, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design '98*, Kluwer, Dordrecht, Lisbon, Portugal, pp. 67–86.

Corbett-Clark, T. A. (1998). *Explanation from Neural Networks*, DPhil thesis, Department of Engineering Science, Oxford University.

Corbett-Clark, T. A. and Tarassenko, L. (1997). A principled framework and technique for rule extraction from multi-layer perceptrons, *Proceedings of the 5th International Conference on Artificial Neural Networks*, IEE Conference Publication No 440, pp. 233–238.

Diamantaras, K. I. and Kung, S. Y. (1996). *Principal Component Neural Networks: Theory and Applications*, Adaptive and Learning Systems for Signal Processing, Communication, and Control, John Wiley, New York, NY.

Jardine, N. and Sibson, R. (1968). The construction of hierarchic and non-hierarchic classifications, *The Computer Journal* **11**(2): 177–184.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, Probability and Mathematical Statistics, John Wiley, New York, NY.

Kohonen, T. (1997). *Self-Organizing Maps*, number 30 in *Springer Series in Information Sciences*, second edn, Springer-Verlag, Berlin.

Ling, R. F. (1973). A computer generated aid for cluster analysis, *Communications of the ACM* **16**(6): 355–361.

Luger, G. F. (1994). *Cognitive Science: The Science of Intelligent Systems*, Acadmic Press, San Diego, CA.

Matthews, P. C. (1998). *Preliminary evaluation of conceptual mechanical designs*, First year report, Cambridge University Engineering Department.

Matthews, P. C. (2001). *The Application of Self Organizing Maps in Conceptual Design*, PhD thesis, Cambridge University Engineering Department.

Matthews, P. C., Langdon, P. M. and Wallace, K. M. (2001). New techniques for design knowledge exploration: A comparison of three data grouping approaches, *in* S. J. Culley, A. H. B. Duffy, C. McMahon and K. M. Wallace (eds), *Proceedings of the 13th International Conference on Engineering Design*, Vol. 2, IMechE, Professional Engineering Publishing, London, Glasgow, pp. 107–113.

Matthews, P. C., Wallace, K. M. and Blessing, L. T. M. (2000). Design heuristics extraction: Acquiring engineering knowledge from previous designs, *in* J. S. Gero (ed.), *Artificial Intelligence in Design 2000*, Kluwer, Dordrecht, pp. 435–453.

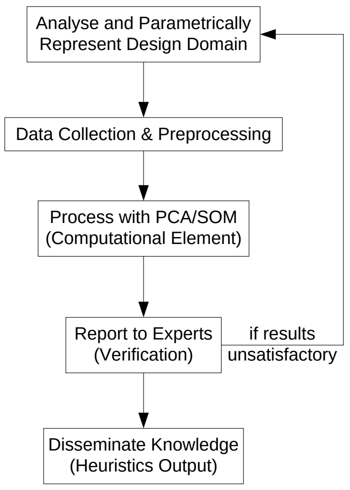
Reich, Y. and Barai, S. V. (1999). Evaluating machine learning models for engineering problems, *Artificial Intelligence in Engineering* **13**(2): 257–272.

Reich, Y. and Travitzky, N. (1995). Machine learning of material behaviour knowledge from empirical data, *Materials & Design* **16**(5): 251–259.

Rich, E. A. and Knight, K. (1991). *Artificial Intelligence*, second edn, McGraw-Hill, Inc, New York.

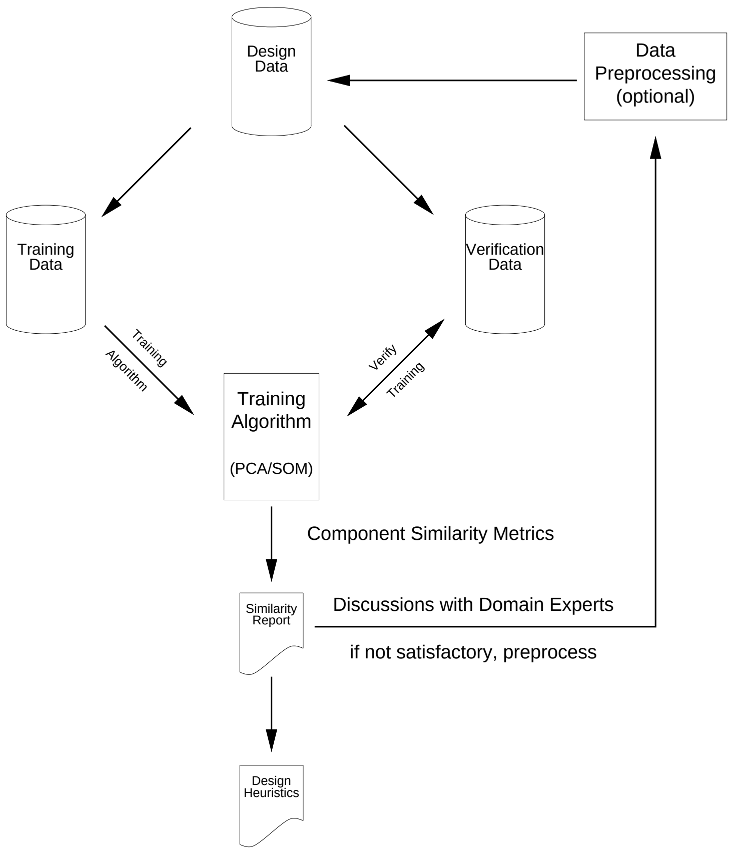
Tickle, A. B., Andrews, R., Golea, M. and Diedrich, J. (1998). The truth will come to light: Direction and challenges in extracting the knowledge embedded within trained artificial neural networks, *IEEE Transactions on Neural Networks* **9**(6): 1057–1068.

Ultsch, A. (1993). Self organized feature maps for monitoring and knowledge aquisition of a chemical process, *in* S. Gielen and B. Kappen (eds), *Proceedings of the International Conference of Artificial Neural Networks*, Springer-Verlag, London, pp. 864–867.



Analyse and Parametrically Represent Design Domain
Data Collection & Preprocessing
Process with PCA/SOM (Computational Element)
Report to Experts (Verification)
if results unsatisfactory
Disseminate Knowledge (Heuristics Output)

Figure 1. Overview of the HEM process



Design Data
Data Preprocessing (optional)
Training Data
Verification Data
Training Algorithm
Verify Training
Training Algorithm (PCA/SOM)
Component Similarity Metrics
Similarity Report
Discussions with Domain Experts
if not satisfactory, preprocess
Design Heuristics

Figure 2. Overview of the heuristic extraction algorithm

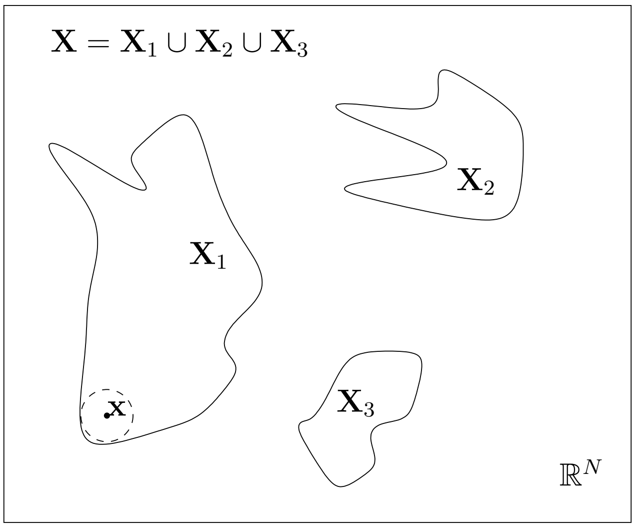


Figure 3. Illustration of a piecewise connected set

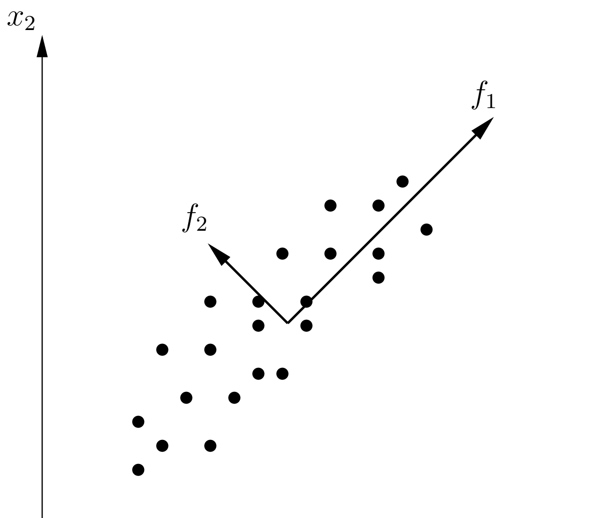
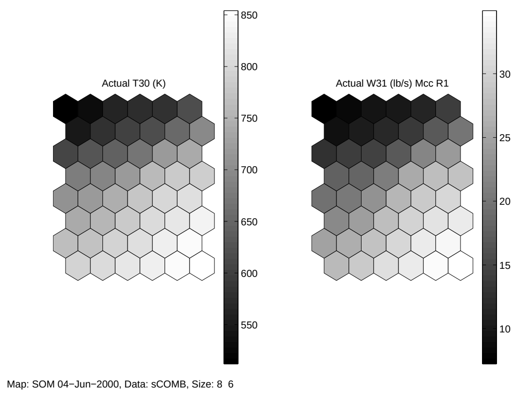
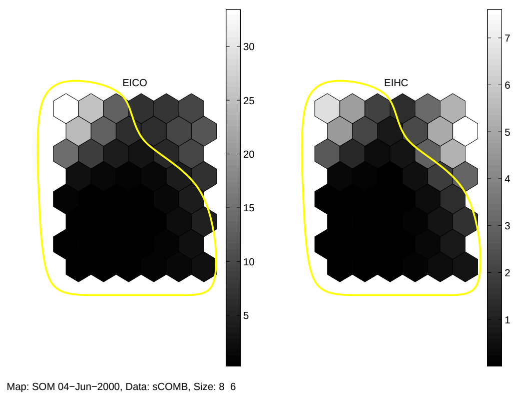


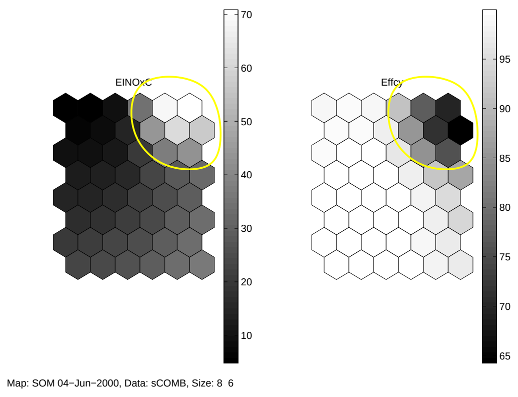
Figure 4. The Principal Components of some two-dimensional data



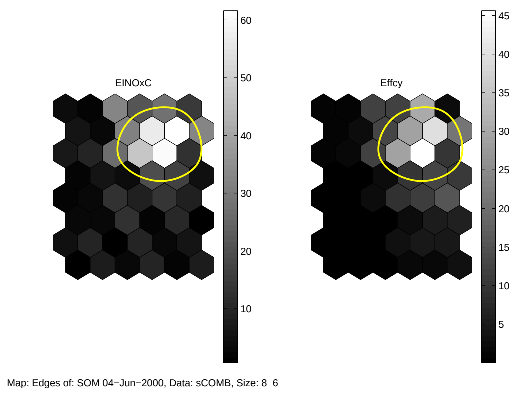
(a) Basic (global) correlation



(b) Local correlation



(c) Anti correlated maps



(d) Edge detection on anti-correlated maps

Figure 5. Interpreting the Self Organizing Maps for a variety of cases

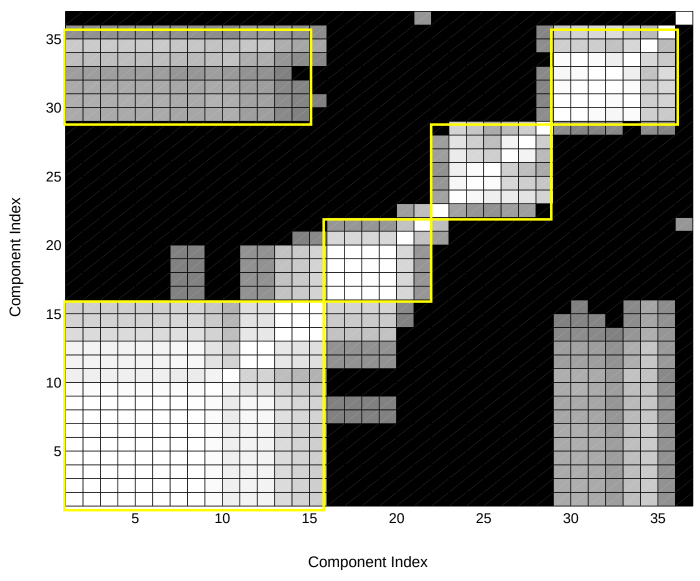


Figure 6. The sorted Tanimoto similarity matrix: black = no similarity, white = similarity (taken from the Combustor dataset)

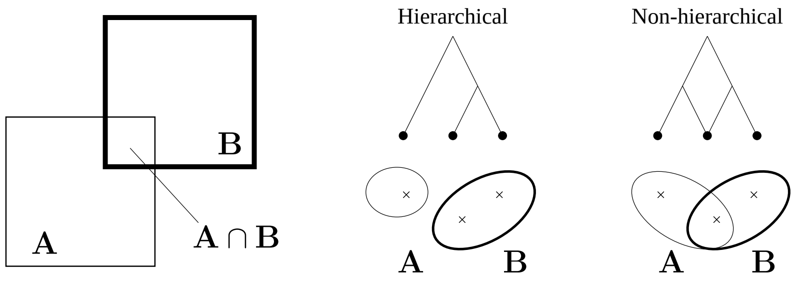


Figure 7. Illustration of hierarchical versus non-hierarchical clustering

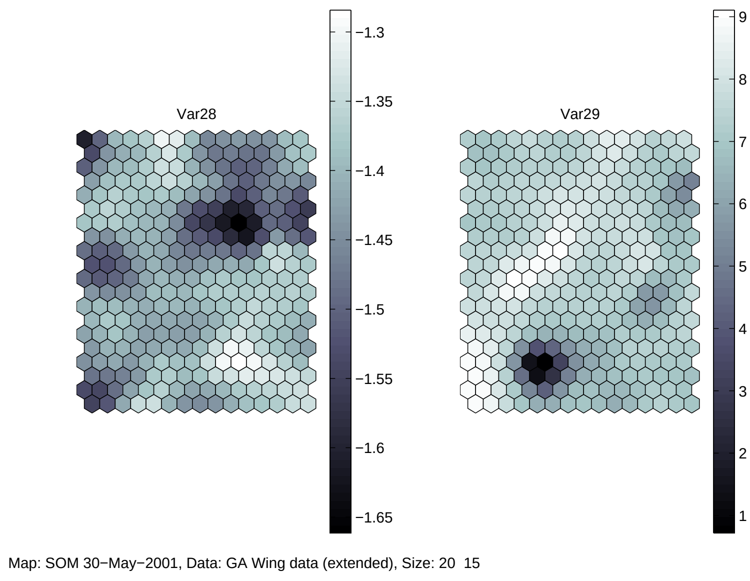
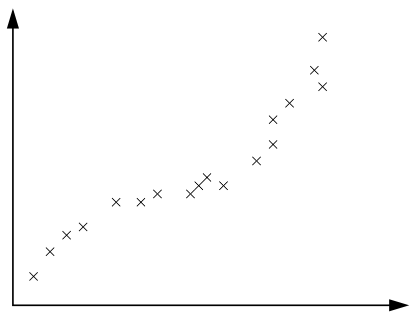
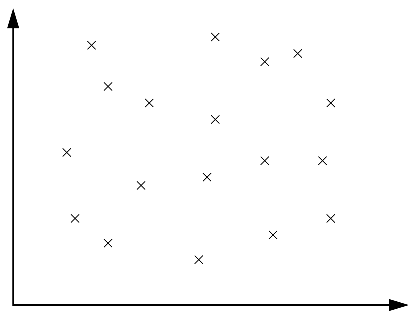


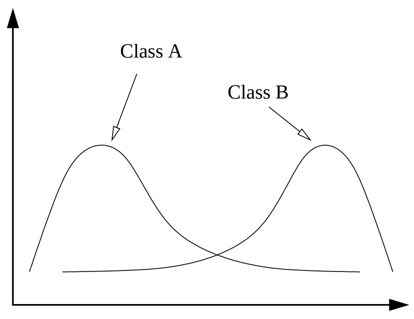
Figure 8. Example of where the Tanimoto metric has incorrectly identified two maps as similar



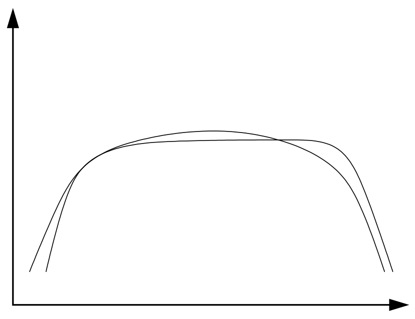
(a) Continuous–continuous data: example of positive evidence for a relationship



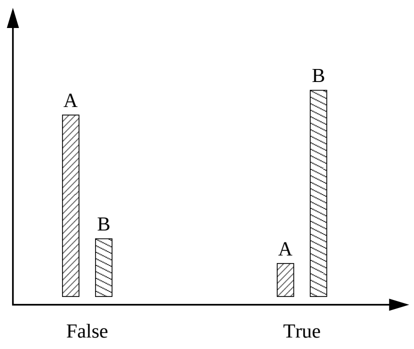
(b) Continuous–continuous data: example of negative evidence for a relationship



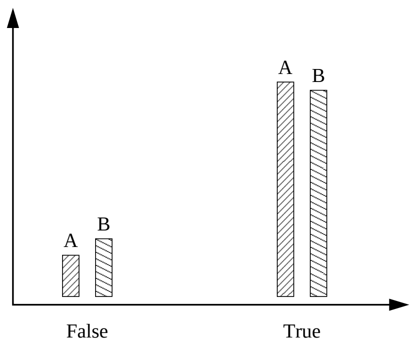
(c) Boolean–continuous data: example of positive evidence for a relationship



(d) Boolean–continuous data: example of negative evidence for a relationship



(e) Boolean–Boolean: example of positive evidence for a relationship



(f) Boolean–Boolean: example of negative evidence for a relationship

Figure 9. Plots used to check heuristics against data: contrasting when a relationship is present between components (left hand side) and when there is no relationship (right hand side)