Conceptual Clustering with Numeric-and-Nominal Mixed Data — A New Similarity Based System

Cen Li and Gautam Biswas
Department of Computer Science
Box 1679, Station B, Vanderbilt University, Nashville, TN 37235.
Email: cenli, biswas@vuse.vanderbilt.edu

Abstract

This paper presents a new Similarity Based Agglomerative Clustering(SBAC) algorithm that works well for data with mixed numeric and nominal features. A similarity measure, proposed by Goodall for biological taxonomy[13], that gives greater weight to uncommon feature-value matches in similarity computations and makes no assumptions of the underlying distributions of the feature-values, is adopted to define the similarity measure between pairs of objects. An agglomerative algorithm is employed to construct a concept tree, and a simple distinctness heuristic is used to extract a partition of the data. The performance of SBAC has been studied on artificially generated data sets. Results demonstrate the effectiveness of this algorithm in unsupervised discovery tasks. Comparisons with other schemes illustrate the superior performance of the algorithm.

1 Introduction

The widespread use of computers and information technology has made extensive data collection in businesses, manufacturing, and medical organizations a routine task, but the primary challenge that drives the relatively new field of database mining or knowledge discovery from databases is the extraction of potentially useful information by careful processing and analysis of this data in a computationally efficient and sometimes interactive manner[17]. Frawley et al.[10] define knowledge discovery to be *"the non trivial extraction of implicit, previously unknown and potentially useful information in data."* This suggests a generic architecture for a discovery system, illustrated in Figure 1. At the core of the system is the discovery engine, which computes and evaluates groupings, patterns, and relationships using a relevant set of features selected in the context of a problem solving task[1]. Depending on the discovery engine employed in the system, the results can be further analyzed to derive models as rules, analytic equations, and concept definitions under the chosen context. Typically, the discovery engine is powered by one of two mechanisms:

- *supervised classification schemes*, which assume that class labels of the data objects being analyzed are known. The process extracts rules that identify groups of objects that have the same class label and differentiates among groups of objects that have different labels. The goal is to classify new data observations into one of a set of known categories.
- *unsupervised classification or clustering schemes*, which make no assumptions or have no knowledge of the category structure. They use objective *criterion functions* to define similarity or dissimilarity among objects. The goal is to find "natural groups", i.e., to partition the data into groups so that objects that are more similar tend to fall into the same group, and objects that are relatively distinct tend to separate into different groups.

Both classification and clustering schemes require data objects to be defined in terms of a predefined set of features. Features represent properties of the object that are relevant to the problem solving task. For example, if we wish to classify automobiles by speed and power, body weight, body shape, and engine size are relevant features, but the color of the car body is not. Selection of appropriate features is an important task in clustering and classification applications.

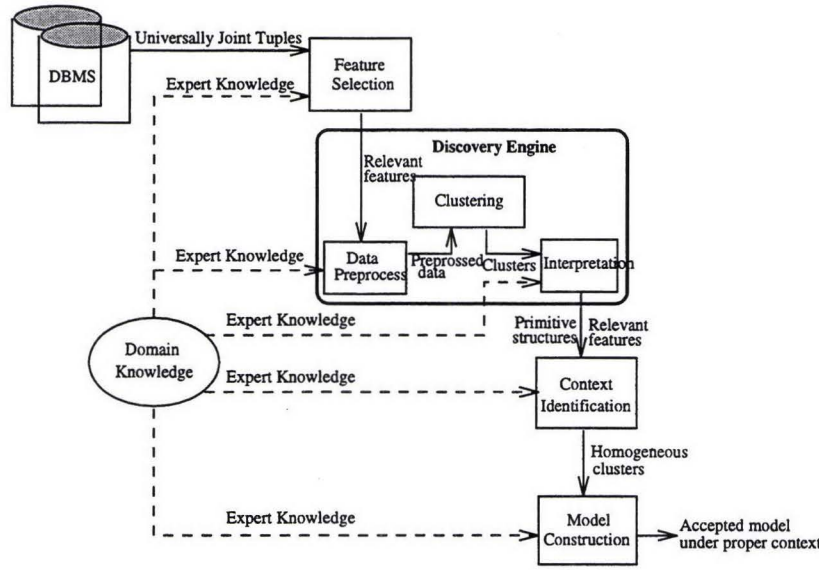

Figure 1: Discovery system architecture with unsupervised classification discovery engine

Traditional clustering methodologies[5, 15] assume features are numeric-valued, but as applications extend from scientific and engineering domains to the medical, business, and social domains, one has to deal with features, such as sex, color, shape, and type of diseases, that are nominal-valued. Schemes for processing nominal-valued data, called conceptual clustering, combine clustering with the concept formation and interpretation tasks[8].

Clustering methodologies incorporate three main steps[1]: preprocessing, clustering, and explanation. Data preprocessing involves the selection of relevant features for the analysis task and the characterization of individual features at the right level of granularity. The clustering step constructs a flat or hierarchical partitioning of the objects based on an objective criterion function and a control algorithm which can be (i) *agglomerative* or (ii) *divisive*. The explanation step assigns meaning to the groups or structures discovered based on the feature-value definitions associated with each group. The explanation process is often governed by characteristics such as parsimony and completeness.

In the real world, a majority of the useful data is described by a combination of numeric and nominal-valued features. For example, if we look at geological data, features such as age, porosity, and permeability are numeric-valued, but other features such as rock types, crystalline structure, and facies structure are nominal-valued. Though effective methods do not exist for clustering data sets with mixed numeric and nominal data, symbolic and numeric clustering methods have by themselves approached their maturity[7, 15]. In knowledge discovery tasks, it becomes crucial that clustering and discovery systems deal with combinations of numeric and nominal-valued data. Section 2 discusses criterion functions used for numeric and symbolic clustering, and illustrates their primary differences. Attempts to develop criterion functions for mixed data have not been very successful mainly because of the differences in the characteristics of these two kinds of data. To overcome this problem, pragmatic approaches have been adopted. Most solutions attempted fall into one of the following categories:

- Encode nominal attribute values as numeric integer values, and apply distance measures used in numeric clustering for computing proximity between object pairs. In many cases, the translation from nominal data to numeric does not make sense, and the proximity values are hard to interpret.
- Discretize numeric attributes and apply symbolic clustering algorithms. The discretization process often causes loss of important information especially the relative (or absolute) difference between values for the numeric features.
- Generalize criterion functions designed for one type of feature to handle numeric and non numeric feature values. The primary difficulty in doing this can be attributed to the nature of the criterion

functions used. Criterion functions used in symbolic clustering are based on probability distribution functions. For nominal-valued attributes, probability distributions can be approximated by simple counting schemes. Computing and bounding density estimates for numeric-valued attributes are a much more difficult task. This method has been implemented in some systems, but experimental results discussed later in this paper show they produce much better results for pure nominal-valued data than they do for mixed or pure numeric-valued data. Criterion functions used in numeric clustering are based on distance metrics that can not be extended to nominal attributes except in a very superficial way.

The focus of this paper is on developing unsupervised discovery techniques that exhibit good performance with mixed data. The core of the methodology is based on a similarity measure from biological taxonomy proposed by Goodall[13]. The measure processes numeric and nominal valued attributes within a common framework and can be conveniently coupled with a simple control strategy that agglomeratively constructs a hierarchical concept tree, from which a distinct partition is extracted by a simple procedure that trades off distinctness for parsimony.

The rest of the paper is organized as follows. Section 2 reviews existing criterion functions and clustering systems that have been developed. Section 3 presents our clustering system SBAC, with its two components: (i) the similarity measure and (ii) the agglomerative control structure. Section 4 demonstrates the effectiveness of the SBAC system by a comparative empirical study of its performance on artificially generated and standard data. General discussion and the conclusions of this work follow in Section 5.

2 Background

Traditional approaches to cluster analysis (numerical taxonomy) represent data objects as points in a multi-dimensional metric space and adopt *distance metrics*, such as Euclidean and Mahalanobis measures, to define similarity between objects[5, 15]. On the other hand, conceptual clustering systems use conditional probability estimates as a means for defining the relation between groups or clusters. Systems like COBWEB[6] and its derivatives use the *Category Utility(CU)* measure[12], which has its roots in information theory, to partition a data set in a manner that maximizes the probability of correctly predicting a feature-value given group C_k versus the same probability given the distribution for the entire data set. Systems like WITT[14] use *correlation* measures to define groupings. These measures are tailored for nominal attributes, though variations, such as COBWEB/3[18] and ECOBWEB[20] use modifications of the CU measure to handle numeric attributes. AUTOCLASS[3] uses a fundamental *finite mixture model*, and derives groupings of objects that locally maximize the posterior probability of individual clusters given the feature distribution assumptions. The COBWEB/3, ECOBWEB, and AUTOCLASS systems are discussed in greater detail below.

2.1 COBWEB/3

The category utility function[12] defines a probability matching strategy to measure the usefulness of a class in correctly predicting feature values:

$$CU_k(nominal) = P(C_k) \sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2, \quad (1)$$

where $P(A_i = V_{ij})$ is the unconditional probability of attribute A_i taking on value V_{ij}, and $P(A_i = V_{ij} \mid C_k)$ is the conditional probability of $A_i = V_{ij}$ given class C_k. The difference, $\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2$, represents an increase in the number of attribute values that can be predicted correctly for class k versus the expected number of correct predictions given no class information. The partition score, i.e., the utility of a partition structure made up of K classes, is defined as the average CU over the K classes: $\frac{\sum_{k=1}^{K} CU_k}{K}$. COBWEB/3 combines the original COBWEB[6] algorithm with the methodology defined in CLASSIT[11] to handle numeric attributes in the CU measure. For numeric attributes, probabilities are expressed in terms of the probability density function(*pdf*) defined for the range of values that can be associated with the attribute. COBWEB/3 assumes that the numeric feature values are normally distributed, and computes the

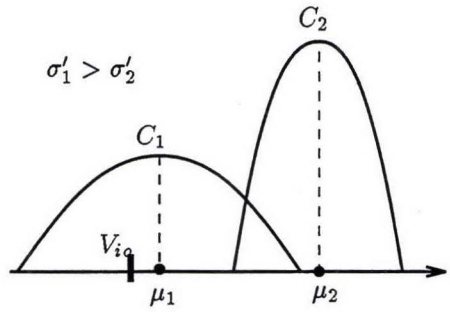

Figure 2: A special case scenario for COBWEB/3

$\sum_j P(A_i = V_{ij} \mid C_k)^2$ term in the following manner:

$$\int_V P^2(A_i = V \mid C_k)dv \equiv \int_{-\infty}^{\infty} \frac{1}{2\pi{\sigma_{ik}}^2} e^{-(\frac{x-\mu_{ik}}{\sigma_{ik}})^2} dv \equiv \frac{1}{2\sqrt{\pi}\sigma_{ik}},$$

where μ_{ik}, and σ_{ik} are the mean and standard deviation of feature i in class k, respectively. Similarly, the unconditional probability, $\sum_j P(A_i = V_{ij})^2$, is computed as $\frac{1}{2\sqrt{\pi}\sigma_i}$, where σ_i is the standard deviation of values of A_i. With the direct interpolation of the criterion function, the counting scheme used for estimating probabilities for nominal features is switched to estimating mean and standard deviation for a numeric features.

A problem arises with this approach as the range of possible values for a feature A_i narrows, i.e., σ_{ik} (or σ_i) becomes very small. In the extreme case, when all objects in class C_k have a unique value, $\sigma_{ik} = 0$, and $\frac{1}{\sigma_{ik}} \rightarrow \infty$. Therefore, $CU_k \rightarrow \infty$. COBWEB/3 introduces a practical solution to this problem in the form of an additional parameter called *acuity*, $\frac{1}{\sigma}$, which is usually set to 1.0. This value bounds the value of σ to be used in the CU_k computation. When σ_{ik} for a numeric attribute i and class k becomes less than $1 (\sigma_{ik} < 1)$, the acuity threshold is applied, and $\frac{1}{\sigma_{ik}}$ is set to 1.

For the set of numeric features, CU is defined as:

$$CU_k(numeric) = \frac{P(C_k)}{2\sqrt{\pi}} \sum_i \frac{1}{\sigma_{ik}} - \frac{1}{\sigma_i},$$

and for a set of mixed nominal and numeric features, the overall CU is the sum of the CU contributions from nominal and numeric features:

$$CU_k = CU_k(nominal) + CU_k(numeric).$$

In review, the nominal form of the CU function describes a well defined systematic information theoretic measure for evaluation of the goodness of a partition structure. However, the numeric form of the CU measure has a number of limitations. First, it assumes that feature-values are normally distributed. This may not be true in general. Second, the mean and variance are estimated by using the sample means and variances. In many cases, the sample populations are very small, therefore, the accuracy of the estimate is suspect. Also, when feature values converge, σ_{ik} becomes small, but the computation is bounded by the acuity measure. For these reasons, it becomes hard to bound the accuracy of the CU measure for numeric features.

From another viewpoint, the numeric CU measure does not take into account an important piece of information, the actual distances between object values, in determining class structure. This can lead to counterintuitive results. A special case is shown in Fig. 2 to illustrate the point. For two intermediate clusters C_1 and C_2, μ_1, μ_2 are the means and σ_1, σ_2 are the standard deviations of a numeric valued attribute A_i in each cluster respectively, with $\mu_1 < \mu_2$ and $\sigma_1 > \sigma_2$. A new object to be evaluated has value V_{i0} for A_i. Tentatively putting the new object into cluster 1 or 2 changes their standard deviations to σ'_1 and σ'_2, respectively. If $\sigma'_1 > \sigma'_2$, then $CU_1 < CU_2$, so the object will be placed in cluster 2 despite the fact that $|V_{i0} - \mu_1| < |V_{i0} - \mu_2|$.

Method	Interval Value
static	$\frac{\text{expected range of attribute values for } A_i}{\text{expected number of distinct intervals for attribute } A_i}$
dynamic	$\frac{2*\sigma_i}{\text{expected number of distinct intervals of attribute } A_i}$
adaptive	$\frac{\sqrt{2*\sigma_i*\text{expected range of attribute value of } A_i}}{\text{expected number of distinct intervals of attribute } A_i}$

Figure 3: Three methods implemented in ECOBWEB for defining the interval value

2.2 ECOBWEB

ECOBWEB is part of a larger system, Bridger[19, 20], that employs inductive learning techniques in designing cable-stayed bridges. At the core of the system is the clustering algorithm, ECOBWEB, an extension of the COBWEB system. ECOBWEB attempts to remedy some of the disadvantages inherent in the COBWEB/3 interpolation scheme:

- the normal distribution assumption for the *pdf* of numeric features, and
- the acuity value for bounding the CU contribution from numeric features.

In the ECOBWEB approach, the probability distribution for numeric features is approximated by the probability distribution about the mean for that feature.

For numeric features, the ECOBWEB approach captures the essence of the CU function by introducing an approximation:

$$\sum_j P(A_i = V_{ij} \mid C_k)^2 \approx P(A_i = \bar{V}_i \mid C_k)^2,$$

i.e., the expected score is estimated around the mean, $\bar{V}_i$, of the feature value distribution. This is computed by

$$P(A_i = \bar{V}_i \mid C_k)^2 = \sum_i (\int_{-I_{ik}}^{I_{ik}} p_{ik}(v)dv)^2,$$

where $\bar{V}_i$ is the mean value for feature i in cluster k, and I_i is the designated interval range around $\bar{V}_i$. For a set of N numeric features, the CU measure is defined as:

$$CU_k = P(C_k) \sum_i^N (\int_{-I_{ik}}^{I_{ik}} p_{ik}(v)dv)^2 - (\int_{-I_i}^{I_i} p_i(v)dv)^2$$

The choice of the interval size, I_i, has a significant effect on the CU computation. The simplest definition of the interval size I_i is a fixed one, where

$$2I_i = \frac{\text{expected range of attribute value of } A_i}{\text{expected number of distinct intervals of attribute } A_i.}$$

This fixed interval size is used through the entire clustering process. As the clustering hierarchy grows deeper, the range of the attribute values within each class may become narrower, therefore, these distributions become sharper and more peaked. If the interval value, $2I_i$, is chosen to be too large, the $P(A_i = V_{ij} \mid C_k)$ will always evaluate to 1, and this methodology fails to differentiate contributions of features with narrow distributions lower down in the hierarchy. On the other hand, if the value chosen results in a very narrow interval, the probability about the means of two distributions can be equal even though the actual distributions may differ dramatically. To address these problems, ECOBWEB defines two other methods for computing the interval values, a dynamic approach, where interval width is a function of the feature-value variance, and an adaptive approach, which uses the geometric mean of the static and dynamic approaches in computing interval size.

Method	static	dynamic	adaptive
static	1	0.187	0.333
dynamic		1	0.187
adaptive			1

(a) $n = 8$

n	2	8	16
2	1	0.707	0.613
8		1	0.667
16			1

(b) *Method* = statistic

Table 1: Comparisons of top level partitions generated by ECOBWEB on IRIS data with different combinations of n and *method* values.

Table 3 lists the three schemes that have been implemented in ECOBWEB to generate the $2I_i$ value, all of which rely on a user defined parameter, the "expected number of distinct intervals of property A_i", n. Since the choice of the $2I$ value is likely to have a very significant effect on the performance of the system, it is important to see how the two user controlled parameters: the *method* for calculating $2I_i$, and n, the expected distinct intervals of a property, affect the structure of the clustering hierarchy. A cross-comparative study is performed for this purpose where the system is run repeatedly on a common data set, with a different combination of the *method* and the n value for each run. The data set used in this study is the well known IRIS [1]data set. Top level partitions of the concept trees were extracted and their structural differences were compared using a measure we call the *Number of matched objects*. This measure is defined to be the total number of objects that group into corresponding clusters for two partitions created from two different runs. The *Number of matched objects* is normalized by the total number of objects in the data set to give a measure of similarity([0-1]) between the partitions generated in each run. The higher the similarity score, the more correspondence found between clusters of the two partitions, and the more similar are the pair of partitions. A cross tabulation between pairs of partitions is listed in Table 1. In Table 1(a), n was kept at a fixed value of 8 [2], and the three *methods* static, dynamic, and adaptive defined the runs. The pairwise similarity measures appear in the table. The numbers indicate a significant difference in the groupings for the three methods. In Table 1(b), the *method* was kept fixed(static), and the n value was varied. Results are shown for $n = 2, 8$, and 16. The higher similarity scores indicate that the partition structures are less sensitive to changes in n, though there are still significant structural differences in the concept hierarchies generated.

These results verify the above analysis given above that the clustering structures generated by ECOBWEB are very much dependent on the two user defined parameters. This may not be much of an issue when the hierarchical clustering tree is used for prediction purposes[19], but it is not likely to be a desirable feature objectives in knowledge discovery tasks. In the rest of the paper, $n = 8$, and *method*=static are used in experiments with ECOBWEB.

2.3 AUTOCLASS

AUTOCLASS[3] imposes a classical finite mixture distribution model on the data and uses a Bayesian method to derive the most probable class distribution for the data given prior information(PI). The finite mixture model contains two parts: (i) the class probability, $P(X_i \in C_j \mid PI)$, the probability that object X_i belongs to class C_j independent of information of the object, but based on prior information available about the allowed search space, the distribution assumptions, and the distribution parameters, and (ii) the intraclass *pdf*, $P(\vec{X}_i \mid X_i \in C_j, PI)$, the probability of observing the instance attribute values $\vec{X}_i$ given object $X_i \in C_j$. The intraclass mixture *pdf* is a product of individual or covariant attribute *pdf*'s, such as Bernoulli distributions for nominal attributes, Gaussian distributions for numeric attributes, and Poisson distributions for number counts.

[1]The Iris data set, chosen for the study contains 150 objects. Each object is described in terms of 4 numeric-valued attributes: sepal length, sepal width, petal length, and petal width. The objects are equally distributed in the three classes : setosa, versicolor and virginica. From previous studies, it is known that the setosa class is distinct, but versicolor and virginica are mixed.

[2]$n = 8$ is the default value in the system for the "expected number of intervals for a property". Experimental results in [19] indicated that *method*=static generated the best results for the ECOBWEB system.

Bayes rule can be applied to combine these two probabilities, i.e.,

$$P(\vec{X}_i, X_i \in C_j \mid PI) = P(X_i \in C_j \mid PI) \cdot P(\vec{X}_i \in C_j),$$

from which the probability of observing an instance X_i with attribute value $\vec{X}_i$, regardless of class, is derived by summing over all classes, and then the probability of observing the entire data set is derived by taking the product over all instances, i.e.,

$$(X \mid PI) = \prod_i (\sum_j P(X_i \in C_j \mid PI) \cdot P(\vec{X}_i \mid X_i \in C_j, PI).$$

This is converted to a Bayesian model that describes data and the distribution parameters, i.e.,

$$P(X, \vec{V}) \mid PI) = P(\vec{V} \mid PI) \cdot P(X \mid \vec{V}, PI).$$

The second term in the right hand side of this equation corresponds to the $P(X \mid PI)$ term above.

Assuming the number of classes for the data is known, the goal of AUTOCLASS is to maximize the posterior probability given the data. Two parameters help define the posterior probability: (a) the parameter values estimated for the chosen distributions, and (b) the *pdf* chosen for each feature.

With no information on class membership, direct optimization approaches for finding the best parameter values from mixture *pdf* is computationally intractable even for moderate size data sets. A variation of Dempster's EM algorithm[4] is used to approximate the solution where weighted instance assignments and weighted statistics, calculated from normalized class probability and data instances, are used to represent known assignment statistics. This allows locally optimal estimation of the parameter values. Weighted instance assignments and weighted statistics can be updated using the newly estimated parameter values which in turn allow for reestimation of locally optimal parameter values. Cycling between these two steps carries the current parameter values and weight estimates toward a mutually predictive and locally maximal stationary point.

Once multiple locally maximum parameter value sets are estimated for a given classification *pdf*, the posterior probability of the current classification *pdf* can be approximated using the local statistics derived for those parameter value sets. The posterior probability of the current classification *pdf* is recorded. Then the attribute *pdf* is selectively changed in each class. This initiates another round of the estimation process finding the parameter value sets and the posterior probability of this new classification *pdf*. The classification *pdf* that has the highest posterior probability is retained at the end of this iterative process.

In the description, it has been assumed that the number of classes presented in the data is known in the repetitive process of searching for the best $P(\vec{V} \mid PI)$ and best $P(T \mid PI)$, where T denotes the abstract mathematical form of the *pdf*. In fact, on top of this two-step process, another level of search for the approximate optimal number of classes for the data has to be imposed[2]. AUTOCLASS starts out with a number of classes J greater than the predicted true number of classes for the data. If the resulting classes all have significant probability, then the number of classes are increased progressively till there are classes in the resulting partition with negligible posterior probability. AUTOCLASS ignores these classes, and the populated classes represent an optimal classification of the data given the assumed class model function.

Despite the fact that a number of approximation and optimization methods have been implemented in the AUTOCLASS system, the computational complexity required by the three level search process is extremely high. Usually it takes days, sometimes weeks, on SPARC station architectures, for the system to reach meaningful results even for moderate sized data sets. It is a primary deficiency associated with this kind of exhaustive-search approach.

AUTOCLASS also suffers from the problem of over fitting associated with maximum likelihood optimization of probabilistic model. The Bayesian approach can adopt an "Occam Factor" scheme to limit the degree of over fitting, but it is hard to establish that the factor takes effect early enough in a classification process. Experimental results show AUTOCLASS tends to produce many more classes than the other unsupervised classification methods, especially when numeric features dominate the class structure.

3 The SBAC System

From earlier discussion, it is clear that two factors primarily govern the clustering process:

1. the *criterion function* for evaluating the goodness of a partition structure. Examples are the mean square error used in numeric partitioning schemes, the category utility measure used in COBWEB and related systems, and similarity measure based schemes used in agglomerative algorithms. Section 2 discussed two approaches for applying the CU measure to mixed data, and the AUTOCLASS scheme which is based on Bayesian classification schemes.

2. the *control algorithm* for generating the partition structure. Control algorithms can be: (a) *agglomerative*, where the partition structure is constructed bottom up through successive merging of atomic clusters into larger groups, and (b) *divisive*, where clusters are formed top down, from one large cluster and then successively subdivided into smaller groups. All three algorithms described in Section 2 employ divisive control algorithms. The COBWEB-based algorithms are incremental, so clusters are dependent on data orderings.

The SBAC methodology uses a similarity measure defined by Goodall[13], and adopts a hierarchical agglomerative approach to build partition structures. Given a pairwise similarity (or dissimilarity) matrix of the objects to be clustered, SBAC can use the single, complete link, and group-average methods[15] to define the aggregation process. In this work, we adopt the group average method, because empirical studies showed that it produced the best results.

3.1 The Similarity Measure

A number of different similarity measures have been used as measures of proximity. A group of similarity measures based on matching coefficients[15] have been used for nominal-valued data. The CU and correlation measures are best suited for binary-valued features, and result in loss of information and arbitrary weighting of features when applied to multi-valued nominal features because individual features are weighted in proportion to the number of values associated with them. There is even greater loss of information when they are applied to numeric features because the measures do not consider the magnitude of the difference between two feature values.

The Goodall similarity measure, inspired by biological and genetic taxonomy, defines a unified framework for handling nominal and numeric features. A pair of objects (i, j) is considered more similar than a second pair of objects (l, m) iff the objects i and j exhibit a greater match in feature values that are more uncommon in the population. This heuristic is built into the similarity measure by weighing features by the frequency of occurrence of their feature-values in the data set. Uncommon feature values make greater contributions to the overall similarity between the objects.

To explain the heuristic in a more concrete fashion, consider nominal feature k, for two pairs of objects (i, j) and (l, m). Objects i and j have identical value for this feature, i.e., $(V_i)_k = (V_j)_k$, and similarly, objects l and m have identical values for feature k, i.e., $(V_l)_k = (V_m)_k$. However, $(V_i)_k \neq (V_l)_k$, and the value $(V_i)_k$ is more or equally frequent in the population than value $(V_l)_k$, i.e.,

$$((p_i)_k = (p_j)_k) \geq ((p_l)_k = (p_m)_k),$$

where $(p_i)_k$, $(p_j)_k$, $(p_l)_k$, and $(p_m)_k$ define the probabilities of occurrence of the respective feature values in the population. In this situation, value $(V_i)_k$ is more common than value $(V_l)_k$, so, for object pair (i, j), the contribution to the similarity measure feature k, is less than or equal to the contribution to the similarity measure from the object pair (l, m). This can be expressed as:

$$\left.\begin{array}{l}((V_i)_k = (V_j)_k) \wedge ((V_l)_k = (V_m)_k) \\ ((p_i)_k = (p_j)_k) \geq ((p_l)_k = (p_m)_k)\end{array}\right\} \Longrightarrow (S_{ij})_k \leq (S_{lm})_k. \tag{2}$$

When comparing similarity for pairs of numeric feature values, the measure takes both the magnitude of the feature value difference and the uniqueness of the feature value pair into account. The magnitude of feature value difference is considered first. The smaller the magnitude of difference between two values $((V_i)_k, (V_j)_k)$, the less likely it is that a pair of values picked at random will fall in the segment defined by V_{ik} and V_{jk}, therefore, the more similar this pair of objects. For numeric feature k, given two pairs of objects (i, j) and (l, m):

$$(|(V_i)_k - (V_j)_k| > |(V_l)_k - (V_m)_k|) \Longrightarrow (S_{ij})_k < (S_{lm})_k, \tag{3}$$

Object Number	Feature 1 (nominal)	Feature 2 (numeric)
1	a	6
2	b	5.5
3	b	7.5
4	a	6
5	c	10.5
6	c	9
7	c	7.5
8	c	9
9	a	7.5
10	b	7.5

Table 2: Sample data set of 10 objects, each described in one nominal feature and one numeric feature

When the magnitude of difference for two pairs of values are equal, then the uniqueness of segment defined by the values weights the similarity index. The uniqueness of the segment is computed by summing up the frequency of occurrences of all values encompassed by the pair of values, i.e., $\sum_{t=(V_i)_k}^{(V_j)_k} p_t$ and $\sum_{t=(V_l)_k}^{(V_m)_k} p_t$ for object value pairs $((V_i)_k, (V_j)_k)$, and $((V_l)_k, (V_m)_k)$, respectively. For two pairs of values encompassing the same difference interval, the less the frequency of occurrence of values within a segment, the more unique the segment is, thus the more similar the pair of values that define the segment is, i.e.,

$$\left.\begin{array}{c} |(V_i)_k - (V_j)_k| = |(V_l)_k - (V_m)_k| \\ \sum_{t=(V_i)_k}^{(V_j)_k} p_t \geq \sum_{t=(V_i)_k}^{(V_m)_k} p_t \end{array}\right\} \Longrightarrow (S_{ij})_k \leq (S_{lm})_k. \tag{4}$$

3.1.1 Computing similarity for nominal features

To reiterate, the similarity index for a nominal-valued attribute k is defined as follows:

$$\begin{array}{lll} (V_i)_k \neq (V_j)_k & \Longrightarrow & (S_{ij})_k = 0, \\ (V_i)_k = (V_j)_k & \Longrightarrow & 0 < (S_{ij})_k < 1. \end{array}$$

As discussed above, when $(V_i)_k = (V_j)_k$, the degree of similarity is a function of the uncommonality of feature-values within the population. Given two objects i and j, having the same value for feature k (say $(V_i)_k$), to calculate the similarity contribution of this value, we first define its *More Similar Feature Value Set*, i.e., $MSFVS((V_i)_k)$. This is the set of all pairs of values for feature k that are more similar or equally similar to the pair $((V_i)_k, (V_i)_k)$. The value pairs are selected according to relation defined in equation 2. Note that a value pair is more similar if it has lower frequency of occurrence. The probability of picking a pair $((V_l)_k, (V_l)_k) \in MSFVS((V_i)_k)$ at random is $(p_l)^2{}_k = \frac{(f_l)_k \cdot ((f_l)_k - 1)}{n \cdot (n-1)}$, where $(f_l)_k$ is the frequency of occurrence of value $(V_l)_k$ in the population and n is the total number of objects in the population. Summation of the probabilities of all such pairs gives the dissimilarity score of the pair, $(D_{ii})_k$. Thus, the similarity of the pair $((V_i)_k, (V_i)_k)$ is computed as:

$$(S_{ii})_k = 1 - (D_{ii})_k = 1 - \sum_{l \in\ MSFVS((V_i)_k)} (p_l)_k^2. \tag{5}$$

A simple example illustrates this computation. Consider a population of 10 objects, each described by two features, one nominal and one numeric. The feature values of each object are shown in Table 2. As defined above, the similarity value for pairs of nominal feature values that differ is 0, i.e., $S_{(a,b)}$=$S_{(a,c)}$ = $S_{(b,c)}$=0. For pairs of identical values, i.e., $S_{(a,a)}$, $S_{(b,b)}$ and $S_{(c,c)}$, we first find the MSFVS for each pair. In this population, values a and b occur less frequently than value c: $f(a) = 3$, $f(b) = 3$, and $f(c) = 4$. The $MSFVS$ for feature value pair (c, c), $MSFVS(c, c) = \{(a, a), (b, b), (c, c)\}$. Once the MSFVS of the pair is

h

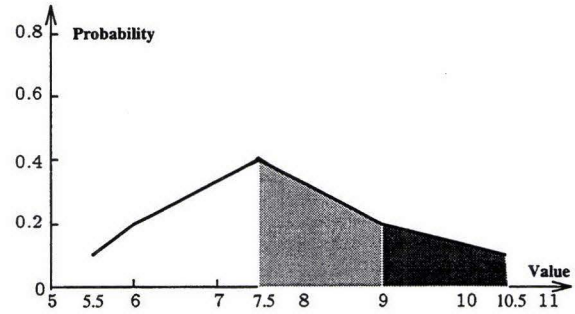

Figure 4: Numeric feature distribution for an example population

identified, the similarity value for the pair (c, c) is calculated according to equation 5:

$$\begin{aligned} D_{(c,c)} &= {p_a}^2 + {p_b}^2 + {p_c}^2 = \tfrac{3(3-1)}{10(10-1)} + \tfrac{3(3-1)}{10(10-1)} + \tfrac{4(4-1)}{10(10-1)} = 0.267, \\ S_{(c,c)} &= 1 - D_{(c,c)} = 0.733. \end{aligned}$$

When computing the similarity between objects 5 and 6, the contribution to the similarity measure from feature 1 is 0.733. On the other hand,

$$\begin{aligned} D_{(a,a)} &= {p_a}^2 + {p_b}^2 = \tfrac{3(3-1)}{10(10-1)} + \tfrac{3(3-1)}{10(10-1)} = 0.133, \\ S_{(a,a)} &= 1 - D_{(a,a)} = 0.867. \end{aligned}$$

This implies, for objects 1 and 4, the similarity contribution to the similarity measure from feature 1 is 0.867 which is greater than that for objects 5 and 8 though both sets have matched feature values.

The computational complexity for determining this value for a feature with m_d observed values is: $O(n + m_d log m_d)$. If $m_d \ll n$, the total number of objects, this reduces to $O(n)$.

3.1.2 Computing similarity for numeric attributes

The similarity score for a numeric feature value pair is calculated in a similar fashion as for a nominal-valued pair. For numeric feature k, to calculate the similarity for a pair of values $((V_i)_k, (V_j)_k)$, first find the *More Similarity Feature Segment Set*, $MSFSS((V_i)_k, (V_j)_k)$. This includes all pairs of values $((V_l)_k, (V_m)_k)$ satisfying equations 3 or 4. The probability of picking two objects from the population having values $(V_l)_k$ and $(V_m)_k$ for feature k, where $((V_l)_k, (V_m)_k) \in \text{MSFSS}((V_i)_k, (V_j)_k)$, is

$$\begin{cases} 2(p_l)_k(p_m)_k = \frac{2(f_l)_k(f_m)_k}{n(n-1)}, & (p_l)_k \neq (p_m)_k, \\ (p_l)_k(p_m)_k = \frac{(f_l)_k((f_l)_k-1)}{n(n-1)}, & (p_l)_k = (p_m)_k, \end{cases}$$

where f_l and f_m are the frequencies of occurrence of values $(V_l)_k$ and $(V_m)_k$, respectively, and n is the total number of objects in the population. Summing up the probabilities of all value pairs in $\text{MSFSS}((V_i)_k, (V_j)_k)$ gives the dissimilarity contribution of feature k, $(D_{ij})_k$. Thus the similarity of the pair $((V_i)_k, (V_i)_k)$ is computed as:

$$(S_{ij})_k = 1 - (D_{ij})_k = 1 - \sum_{l,m \in MSFSS\ ((V_i)_k,(V_j)_k)} 2(p_l)_k(p_m)_k.$$

Using the simple example given in the previous section, we demonstrate next how the similarity measures for numeric features are calculated. The probability distribution of the numeric feature of the sample population is plotted in Figure 4. Consider two pairs of feature values (7.5, 9) and (9, 10.5). Both are 1.5 units apart, but the population encompassed by the two intervals are different. In Figure 4, the population encompassed by the pair (7.5, 9) is lightly shaded, and the population encompassed by the pair (9, 10.5) has a dark shade. The lightly shaded area is larger than the area shaded dark. From our definition for similarity measures for numeric features, we know that the similarity contribution from feature 2 for the pair (9, 10.5)

should be greater than that of the pair (7.5, 9). To calculate the exact similarity measure for each pair, we first identify the MSFSS for each pair:

$$\begin{aligned} MSFSS(7.5,9) &= \{(5.5,5.5),(6,6),(7.5,7.5),(9,9),(10.5,10.5),(5.5,6),(6,7.5),(7.5,9),(9,10.5)\} \\ MSFSS(9,10.5) &= \{(5.5,5.5),(6,6),(7.5,7.5),(9,9),(10.5,10.5),(5.5,6),(9,10.5)\}, \end{aligned}$$

Then we calculate the dissimilarity score for each pair of values as:

$$\begin{aligned} D_{(7.5,9)} &= p_{5.5}p_{5.5} + p_6p_6 + p_{7.5}p_{7.5} + p_9p_9 + p_{10.5}p_{10.5} + 2p_{5.5}p_6 + 2p_6p_{7.5} + \\ &\quad 2p_{7.5}p_9 + 2p_9p_{10.5} = 0.51, \\ D_{(9,10.5)} &= p_{5.5}p_{5.5} + p_6p_6 + p_{7.5}p_{7.5} + p_9p_9 + p_{10.5}p_{10.5} + 2p_{5.5}p_6 + 2p_9p_{10.5} = 0.27. \end{aligned}$$

And finally, the similarity measure for the pair (9, 10.5) is $S_{(9,10.5)} = 1 - D_{(9,10.5)} = 0.73$, which is greater than that of the pair (7.5, 9), $S_{(7.5,9)} = 1 - D_{(7.5,9)} = 0.49$.

Note that the similarity contribution from numeric features is a function of both distance and density. As long as a change of scale or a transformation for a numeric feature preserves the order of relative distances between its feature values, the similarity measure for the feature value pairs is invariant. Also, the monotonic nature of the (dis)similarity measures is preserved. For example, $D(7.5,9) < D(9,11)$ because $|7.5-9| < |9-11|$ even though the range from 9 to 11 is very sparsely populated in the distribution in Figure 2. On the other hand, if two distances are equal, as in the example shown above, the values at the end points of the interval with the sparser population is considered more similar. Therefore, the metrical properties of the distance measure are retained for numeric features.

The direct implementation of the above calculation takes $O({m_c}^4)$ time complexity for a numeric feature with m_c unique values observed. To speed up the computation, we first find the difference of all pairs of values for a numeric feature and sort the differences in ascending order. An accumulator is then used to accumulate and assign similarity score to pairs of values in the sorted difference list. With this implementation, the time complexity for calculating the similarity score for a numeric feature is cut down to $O(m_c^2 log m_c^2)$. In the worst case, $m_c = n$, where n is the total number of objects, and the above complexity is bounded by $O(n^2 log n^2)$.

3.1.3 Combining Similarity Contributions from Individual Features

Up to this point, we have discussed similarity score computations for individual features. If similarity scores from individual features are looked upon as individual tests of similarity between two objects, and if we assume all such individual tests are independent, then we can combine these independent similarity test results into one aggregate similarity score that measures the similarity between two objects across the set of features that define the data.

A common statistical technique for combining probabilities from individual tests is to employ the additive properties of χ^2 distribution, assuming the individual results are expressed as the square of a standard normal deviate. Such combination can be simply achieved using Fisher's χ^2 transformation[9]: $\chi^2 = -2ln(P_i)$. This transformation works well with data from continuous populations and in discrete populations where a feature has a large number of distinct observations associated with it. However, for nominal features with a small number of possible observations, e.g., a binomial distribution with low index, empirical results show that the transformation causes the mean and standard deviation of the transformed population to deviate significantly from the theoretical mean and standard deviation, which diminishes the power of the tests[16]. To remedy this problem, Lancaster suggested a modified transformation called the *mean value* χ^2 transformation[16], χ^2_m. Instead of using the probability of the event actually observed, P, in the transformation, a value intermediate between the probability of the event actually observed, P, and the next smaller probability in the discrete set including it, P', are used. Through probability integral transformation, the *mean value* χ^2 transformation becomes:

$$\chi^2_m = \int_{P'}^{P} (-2lnP) d(\frac{P}{P-P'}) = 2 - \frac{2(PlnP - P'lnP')}{P-P'}.$$

Thus, we combine the similarity test scores from numeric features using Fisher's χ^2 transformation:

$$(\chi_c)^2_{ij} = -2\sum_{k=1}^{t_c} ln((D_{ij})_k), \tag{6}$$

Calculation Steps	Object Pairs	
	(Obj6, Obj9)	(Obj5, Obj8)
χ_c^2	1.35	2.62
χ_d^2	1.037	3.19
χ^2	2.385	5.81
D	0.66	0.21
S	0.34	0.79

Table 3: The step by step calculation of similarity measure between pairs of objects

where t_c is the number of numeric features in the data. χ_c also follows χ^2 distribution with t_c degrees of freedom. The similarity test scores from nominal features are combined using Lancaster's *mean value* χ^2 transformation:

$$(\chi_d)_{ij}^2 = 2\sum_{k=1}^{t_d}(1 - \frac{(D_{ij})_k ln(D_{ij})_k - (D_{ij})'_k ln(D_{ij})'_k}{(D_{ij})_k - (D_{ij})'_k}), \tag{7}$$

where t_d is the number of nominal attributes in the data, $(D_{ij})_k$ is the dissimilarity score for nominal attribute value pair $((V_i)_k, (V_j)_k)$, $(D_{ij})'_k$ is the next smaller dissimilarity score in the nominal set. χ_d^2 is χ^2 distributed with t_d degrees of freedom.

The addition of two χ^2 distribution is still χ^2 with degree of freedom equal to the sum of the two degrees of freedom, i.e., the probability distribution for combining the two types features is χ^2 distributed with $(t_c + t_d)$ degrees of freedom. The significance value of this χ^2 distribution can be looked up in standard tables or approximated from the expression:

$$D_{ij} = e^{-\frac{\chi_{ij}^2}{2}} \sum_{k=0}^{(t_d+tc-1)} \frac{(\frac{1}{2}\chi_{ij}^2)^k}{k!},$$

where $\chi_{ij}^2 = (\chi_c)_{ij}^2 + (\chi_d)_{ij}^2$. The overall similarity score representing the set of (t_c+t_d) independent similarity measures is $S_{ij} = 1 - D_{ij}$.

Continuing with the sample population of objects given in Table 2, we illustrate the computation for the similarity measure for pairs of objects. We have already shown, for the nominal feature, the dissimilarity measure for object pairs (obj6, obj9) and (obj5, obj8), are $D_{(a,c)} = 1$ and $D_{(c,c)} = 0.267$, respectively, and for the numeric feature, the similarity measure for the two pairs are $D_{(7.5,9)} = 0.51$ and $D_{(9,10.5)} = 0.27$, respectively. The similarity measures obtained from individual features are combined to give the overall similarity between pairs of objects. Since there is only one nominal feature and one numeric feature in this example, the summation of t_c and t_d terms in equations 6 and 7 are reduced to single term operations. Results of step by step calculations for χ_d, χ_c, dissimilarity measure and the similarity score for object pairs (obj6, obj9) and (obj5, obj8) are given in Table 3. The overall similarity between object 5 and object 8, 0.79, is higher than that between object 6 and object 9. This is in consistent with the fact that for each individual feature, values between object 5 and object 8 are more similar to each other than those between object 6 and object 9.

3.2 The Control Structure

SBAC's agglomerative, hierarchical clustering algorithm based on the Unweighted Pair Group Method with Arithmetic Average(UPGMA)[15], starts with a pairwise dissimilarity matrix D of the set of objects to be clustered. Dissimilarity between a pair of objects is the complement of their similarity score, $D_{ij} = 1 - S_{ij}$. At any step, the clusters that have the minimum pairwise dissimilarity value are merged into a single cluster. Dissimilarity between the new cluster and the other clusters is defined as the average dissimilarity between an old cluster and the component clusters of the new cluster. The end result is a dendrogram (or classification tree) whose leaf nodes are individual objects, and whose root defines a cluster or group that includes all objects. Fig. 5 describe the steps of the algorithm.

1. Begin with the disjoint clustering, i.e., each object i, $1 \leq i \leq n$ defines its own group. Assign sequence number $m = 0$, and level $L(m) = L(0) = 0$.
2. Find the least dissimilar pair of clusters in the current clustering, say the pair $((r),(s))$, where $D[(r),(s)] = \text{Min} \quad D[(i),(j)]$ for all pairs $((i),(j))$ in the current dissimilarity matrix.
3. Increment the sequence number: $m = m + 1$. Merge clusters $(r),(s)$ into a single cluster (p) to form the next clustering m. Set the level of this clustering to $L(m) = D[(r),(s)]$.
4. Update the dissimilarity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s), and adding a row and column corresponding to the newly formed cluster. The dissimilarity between the new cluster, (p) and an existing cluster (k) is defined as follows:

$$D[(p),(k)] = \frac{n_r}{n_r + n_s} D[(r),(k)] + \frac{n_s}{n_r + n_s} D[(s),(k)],$$

   where n_r and n_s are the number of objects in clusters (r) and (s), respectively.
5. Repeat Steps 2-4 until all objects are in a single cluster.
6. Extract the final partition from the dendrogram by selecting the "significant" cluster down each path of the dendrogram in a manner described in Section 3.3.

Figure 5: The control structure of SBAC system

In the classification tree, each node has an associated dissimilarity score which indicates the dissimilarity level at which its child nodes merge together. The decrease in dissimilarity score from a parent node to a child node indicates that the objects in the child node form a more cohesive group than the ones in the parent node. From the root of the tree to the leaves, the dissimilarity score along each path decreases monotonically. Although the dissimilarity scores of the leaves, i.e., singleton objects, have the lowest value, i.e., 0, the goal is to form clusters or groups that are cohesive but not too fragmented. We adopt this criterion as a measure of the desirable property of the partition structure in the data. This implies the classical tradeoff between cohesiveness and fragmentation. We are interested in clusters that are cohesive yet not fragmented. To achieve the proper balance between these two, we traverse the tree top-down in a depth first fashion, comparing the dissimilarity values between parent and child nodes. We cut off traversal at points where the difference in dissimilarity values is less than a certain percentage threshold, t. The set of nodes on the frontier of the traversal along the different paths define the clusters or groups of interest. In our experiment, we set the threshold value $t = 0.3 * D(root)$, where $D(root)$ is the dissimilarity score for the root of the tree.

To illustrate the scheme, an example of the depth-first traversal is shown on the classification tree in Figure 6. Here, $D(root) = 0.876$, so the threshold value is set at $t = 0.263$. From root of the tree to node 2 and from node 2 to node 4, the dissimilarity scores drop by 0.083 and 0.081, respectively, both of which are lower than the threshold value $t = 0.263$. A significant drop of similarity score, 0.585, is observed from node 4 to node 8, so the search is terminated below node 8. Similarly, following other paths from node 4 to its other children, node 9 and 10, similarity scores drop 0.37 and 0.31, respectively, both of which are higher than t, so the search is terminated below node 9 and 10. The search then continues from other children of node 2, in this case node 5. Another significant similarity score drop of 0.289 is observed in this. This makes node 5 part of the extraction frontier. From the root to node 3, the drop of similarity score is again higher than t. So node 3 becomes part of the extraction frontier and the search terminates. The end result is that nodes 8, 9, 10, 5 and 3 define the partition structure.

The overall computational complexity of the SBAC system can be broken down into two parts: (i) generating the dissimilarity matrix for all pairs of objects, and (ii) constructing the hierarchical dendrogram or concept tree. It is well established in literature[15] that the upper bound complexity for part (ii) is $O(n^2)$, where n is the number of objects in the dataset. Also, constructing the similarity matrix for pairs of objects, given the similarity index for individual attributes, is $O(n^2)$. Using the time complexities to compute the individual similarity indices reported earlier, the overall time complexity for generating partitions is

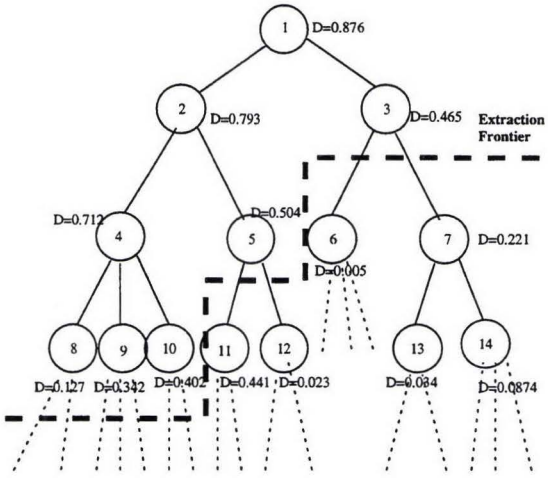

Figure 6: Illustration of partition extraction algorithm in SBAC

$O(n^2 + m_c^2 log m_c^2)$, where m_c is the highest possible number of distinct values observed for a numeric feature. For large data sets, this reduces to $O(n^2)$, where $n \gg m_c$, and for small data sets, it is $O(m_c^2 log m_c^2)$, where $n \ll m_c$.

4 Experimental Studies

Our goal for performing empirical studies with SBAC were two fold: (i) to gain a better understanding of the characteristics of the Goodall similarity measure, and (ii) to compare the performance of SBAC with existing unsupervised discovery systems, such as AUTOCLASS, COBWEB/3, and ECOBWEB.

4.1 Artificial Data

We describe first the method used for generating the artificial data, and then present a comparative analysis of the results.

4.1.1 Data Description

Each artificial data set had 180 data points, equally distributed into 3 classes: G_1, G_2, and G_3. Each data point was described using four features — two nominal and two numeric. The nominal feature values were predefined and assigned to each class in equal proportion. Nominal feature 1 has a unique symbolic value for each class, and nominal feature 2 had two distinct symbolic values assigned to each class. The numeric feature values were generated by sampling normal distributions with different means and standard deviations for each class, where the means for the three classes are three standard-deviations apart from each other. For the first class, the two numeric features are distributed as N($\mu = 4$, $\sigma = 1$) and N($\mu = 20$, $\sigma = 2$); for the second class, the distributions were N($\mu = 10$, $\sigma = 1$) and N($\mu = 32$, $\sigma = 2$); and for the third class, N($\mu = 16$, $\sigma = 1$) and N($\mu = 44$, $\sigma = 2$).

For the base data set, d0%c0%, none of the four feature values were corrupted. The remaining data sets were created by randomly corrupting the base data set using (a) non-Gaussian noise on both the nominal features and the numeric features, and (b) Gaussian noise, only on the numeric features. To avoid possible order effects in COBWEB/3 and ECOBWEB systems, objects for each data were randomized using the same randomization seed before feeding into any of the four systems.

In our experiments, we created data sets that had corrupted numeric features, or corrupted nominal features, but not both. Our primary goal here was to study how each one of the systems was biased toward numeric and nominal features. If both types of features were corrupted at the same time, it would be difficult to judge the performance of the systems on numeric versus nominal features, and compare the performance.

Data with Non-Gaussian Noise Six data sets were created as variations of the d0%c0% data set by successively mixing up 20%, 40%, and 60% of the feature values for one class with the other two classes. For example, the data set d20%c0% represents the data set where 20% of the nominal feature values picked randomly in each class were changed to values from other classes, but none of the numeric feature values were corrupted. On the other hand, d0%c40% implies that no nominal feature values were corrupted, but 40% of the numeric feature values in each class were corrupted by picking their values from the normal distributions defined for the other two classes. The corrupted datasets used in this study can be described as: d20%c0%, d40%c0%, d60%c0%, d0%c20%, d0%c40%, and d0%c60%.

Data with Gaussian noise Four data sets were created by adding successively higher degree of Gaussian noise to the numeric features. The Gaussian noise were distributed as $n(0, \frac{x}{2}\sigma)$, where x indicated the degree of noise introduced. For example, the data set $d0c(\frac{\sigma}{2})$ introduced into the base data Gaussian noise with mean=0, standard deviation=$\frac{1}{2}$ the standard deviations of the original features. Since the standard deviations of the two numeric features in the base data set were 1.0 and 2.0 respectively, $d0c\frac{\sigma}{2}$ represents data set with well separated nominal features and two numeric features corrupted with Gaussian noise $n(0, \frac{1}{2})$ and $n(0, 1)$ respectively. The four corrupted data sets created for this experiment were $d0c(\frac{\sigma}{2})$, $d0c(\sigma)$, $d0c(\frac{3\sigma}{2})$, and $d0c(2\sigma)$.

4.1.2 Results

The partitional structures generated by the four clustering systems are further characterized by a *misclassification count* measure assuming the partitional structure established in the d0%c0% data set as the true structure. The *misclassification count* computes the number of object misclassification in the C_i class assuming G_i is the true class. C_i and G_i correspond when a majority of the C_i objects have the G_i label, and no other G_k $(k \neq i)$ group has a larger number of objects in C_i. If more than three groups are formed, the additional smaller C groups are labeled as fragmented groups. The misclassification count is computed as follows:

1. if an object falls into a fragmented C group, where its type (G label) is a majority, it is assigned a value of 1,
2. if the object is a minority in any group, it is assigned a misclassification value of 2, otherwise
3. the misclassification value for an object is 0.

The sum of the misclassification values for all objects is the misclassification count for the partition. This is used as a measure of the goodness of the partition. The smaller this value, the closer this partition structure is to the true structure.

Results from Data with Non-Gaussian Noise The partitional structures generated by the four methods are listed in Table 4. The *misclassification counts* for the partitional structures are plotted in Fig. 7. The solid lines represent the effects of increasing noise added to the numeric attributes and the dotted lines represented the effects of adding noise to the nominal attributes. It was observed that the AUTOCLASS partitions deteriorated quite drastically as the numeric features were corrupted, and very little deterioration occurred when the nominal features were corrupted. COBWEB/3 seems to have the opposite effect. The differences are smaller for ECOBWEB, but almost equal for numeric and nominal degradation. It is interesting, however, that ECOBWEB is more sensitive to noisy numeric features whereas COBWEB/3 is more sensitive to noisy nominal features. Also, unlike the other systems, ECOBWEB did not perform very well on the base data set. SBAC's degradation is almost negligible compared to the other three for either type of feature degradation. This suggests that COBWEB/3, AUTOCLASS, and ECOBWEB criterion functions are biased toward numeric or nominal data.

Given the way these data sets are generated, SBAC assigns equal similarity scores to the same feature value pairs appearing for each individual feature in each of the seven data sets. When feature values of an object were corrupted (i.e, it took on a value from another class), the contribution of this feature to the similarity computation with other objects belonging to this class decreased significantly, but the other

Data Sets		SBAC	COBWEB3	AUTOCLASS	ECOBWEB
base data	d0%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_3:15) $C_2(G_2$:60) $C_3(G_3$:29) $C_4(G_3$:16)
noisy numeric features	d0%c20%	$C_1(G_1$:59) $C_2(G_2$:60, G_3:1) $C_3(G_3$:59, G_1:1)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:37) $C_2(G_2$:38) $C_3(G_3$:38) $C_4(G_3$:6) $C_5(G_1$:11, G_2:6)) $C_6(G_1$:1, G_2:6, G_3:5) $C_7(G_1$:5, G_3:6) $C_8(G_2$:5, G_3:6) $C_9(G_1$:6, G_2:5)	$C_1(G_1$:48, G_2 :2, G_3:5) $C_2(G_1$:12, G_2:11, G_3:55) $C_3(G_2$:47)
	d0%c40%	$C_1(G_1$:60, G_3:1) $C_2(G_2$:60) $C_3(G_3$:59)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:36, G_3:1) $C_2(G_2$:11, G_3:16) $C_3(G_2$:25, G_3:1) $C_4(G_1$:22) $C_5(G_3$:20) $C_6(G_1$:1, G_3:9) $C_7(G_2$:2, G_3:7) $C_8(G_2$:7, G_3:2) $C_9(G_2$:5, G_3:4) $C_{10}(G_1$:1, G_2:5) $C_{11}(G_2$:5)	$C_1(G_1$:36, G_2:12, G_3:9) $C_2(G_1$:11, G_2:36, G_3:12) $C_3(G_1$:13, G_2:12, G_3:39)
	d0%c60%	$C_1(G_1$:59, G_2:1, G_3:2) $C_3(G_2$:58, G_1:1) $C_3(G_3$:58, G_2:1)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:36) $C_2(G_1$:14) $C_3(G_2$:17, G_3:14) $C_4(G_2$:12, G_3:17) $C_5(G_2$:13, G_3:11) $C_6(G_2$:12, G_3:11) $C_7(G_1$:10, G_2:6, G_3:7)	$C_1(G_1$:60, G_2:2, G_3:57) $C_2(G_2$:58, G_3:3)
noisy nominal features	d20%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_2:7, G_3:1) $C_2(G_2$:53) $C_3(G_3$:59)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_3:12) $C_2(G_2$:60) $C_3(G_3$:27) $C_4(G_3$:21)
	d40%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:12, G_2:14, G_3:36) $C_2(G_1$:11, G_2:35, G_3:12) $C_3(G_1$:37, G_2:11, G_3:12)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_2:18, G_3:10) $C_2(G_2$:42) $C_3(G_3$:28) $C_4(G_3$:22)
	d60%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:17, G_2:15, G_3:26) $C_2(G_1$:16, G_2:25, G_3:18) $C_3(G_1$:27, G_2:20, G_3:16)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60) $C_5(G_3$:18) $C_2(G_2$:21) $C_6(G_3$:21) $C_3(G_2$:23) $C_7(G_3$:21) $C_4(G_2$:16)

Table 4: Partitional structures generated by clustering systems on artificial data sets with non-Gaussian noise

feature contributions remained high. Therefore, the effect of corruption was greatly mitigated. This effect is further illustrated in Table 5, and the average contribution from the nominal features and the numeric features to the overall similarity measure are compared for pairs of objects from the same class and pairs of objects from different classes. For the data set with well separated numeric features and corrupted nominal features, (see Table 5), when comparing the within class and between class object pairs, there is small difference in average similarity for nominal features. This indicates that the predictability of noisy nominal features is lowered. At the same time, there is a big difference in the within class and between class values for the numeric features. Moreover the contribution to similarity from the contribution features is about 2 times the contribution from the nominal features within a class and about $\frac{1}{2}$ between classes. This big difference indicates that in spite of the noise added, objects from the same class are still more similar to each other than objects from different classes and this is primarily because of the difference in contributions

Similarity Contribution	Object Pairs for Data d40%c0%		Object Pairs for Data d0%c40%	
	Same Class	Between Classes	Same Class	Between Classes
Average χ_c^2	8.44	1.82	4.66	3.68
Average χ_d^2	4.54	3.72	7.65	2.20
Average D	0.17	0.69	0.20	0.67
Average S	0.83	0.31	0.80	0.33

Table 5: Similarity contribution from nominal features and numeric features averaged over all pairs of objects from the same class and all pairs of objects from different classes

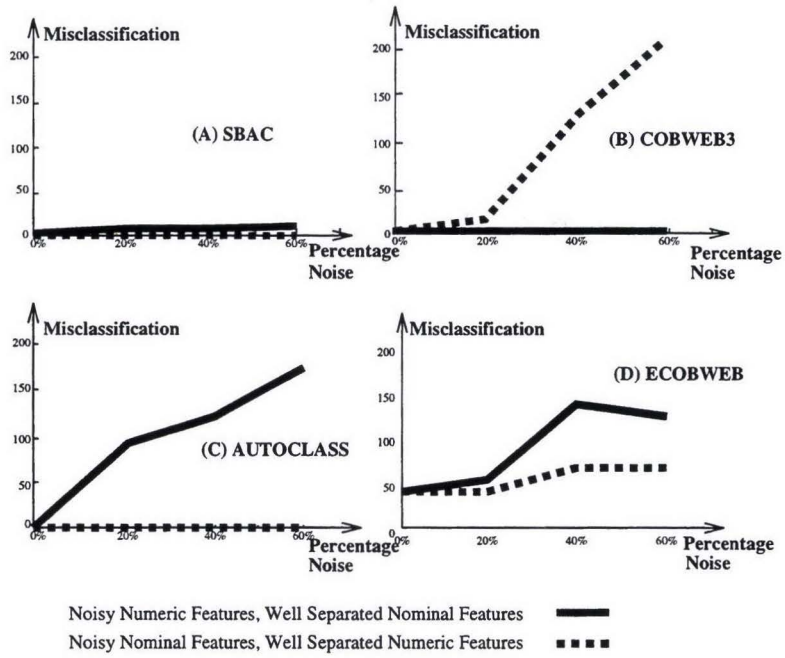

Figure 7: Comparison of partitional structures generated by clustering systems on artificial data sets with non-Gaussian noise

between the uncorrupted versus corrupted features. The overall results for the data set with well separated nominal features and noisy numeric features were just the same. Here noise reduced the difference between the within and between class similarity values for the numeric features, while maintaining a large difference for the nominal features. However, the differences in contribution were not as distinct, which explain the slightly larger misclassification percentages observed in Figure 7(A).

One possible explanation for the AUTOCLASS results may be that because the numeric data was Gaussian, it provided a closer fit to AUTOCLASS's assumption for the distribution of the prior probability. However, AUTOCLASS assumes a Bernoulli distribution with uniform Dirichlet conjugate prior for the nominal attributes, and this did not quite match the way we generated the nominal data for this experiment. Nevertheless, it is still a little surprising that AUTOCLASS seemed to disregard the distinctive nominal attributes completely in d0%c*% cases. Another possible reason for the deterioration of AUTOCLASS's performance to the gradual increase of noise in numeric features is that the noise model assumed by AUTOCLASS is Gaussian. Separate experiments conducted by adding Gaussian noise to the numeric features demonstrated better performance. This is discussed later. Also, when compared to cluster structures generated by the other algorithms, AUTOCLASS tended to construct more groups in the partition structure. This may be attributed to the fact that with the addition of noise, AUTOCLASS found better fit when using a larger number of component distributions. Cheesman and Stutz discuss this issue (*Occam's Razor*[3]) and implement schemes to help prevent AUTOCLASS from overfitting. However, the schemes do not seem to be activated early enough to prevent fragmentation in the partition structures created by AUTOCLASS.

The cluster structure generated by ECOBWEB was not dominated by either well separated nominal attributes or well separated numeric attributes. Despite the fact that ECOBWEB and COBWEB/3 share the same criterion function for nominal attributes, for d0%c⋆% data sets, cluster structures formed by COBWEB/3 are much better than those formed by ECOBWEB. The difference in the partitional structures can be explained by the differences in the implementations of the CU function for numeric features. For COBWEB/3, the $CU_{numeric}$ score depends solely on the standard deviation of the feature distribution for a class. When noise is introduced, the standard-deviations of the "model classes" increase, causing $CU_{numeric}$ to decrease. The higher the level of noise, the less the contribution of $CU_{numeric}$ to the overall CU score. Thus when numeric features were corrupted, the clustering process is mainly based on the $CU_{nominal}$ score which

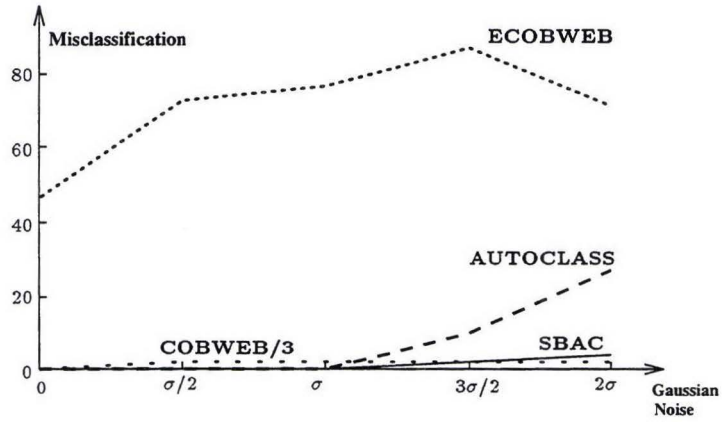

Figure 8: Comparison of partitional structures generated by clustering systems on artificial data sets with Gaussian noise

remains high for all d0%c⋆% cases. For ECOBWEB, its $CU_{numeric}$ score is calculated as the probability of a small interval around the mean value of the feature distribution in a class. When noise is introduced, the mean values of the features are shifted away from those of the "model classes". Depending on the size of the interval defined, the probability calculated around the mean, and thus the $CU_{numeric}$ contribution, can be large enough to make a difference in the overall CU score, which in turn slightly perturbed the class structures that would have been generated by using $CU_{nominal}$ scores alone. It is interesting to observe that the performance of ECOBWEB started to improve after a high level of noise was introduced to the numeric features(60%). A possible explanation for this is when an excessive amount of noise is present, the distribution of numeric features become close to uniform distributions over the entire range. No matter where the estimated mean value is for each of the three classes, the probability of the small interval around each mean is going to be very close. This effectively mitigates the effect of the $CU_{numeric}$ scores on the overall CU measure, causing the clustering system to group objects based mainly on the $CU_{nominal}$ scores.

One the other hand, for d⋆%c0% data, the performance of ECOBWEB is noticeably better than those from COBWEB/3. This is because nominal features have a greater impact on the CU computation than the numeric features for COBWEB/3. When nominal features are corrupted, the $CU_{numeric}$ scores may not be big enough to render the corrupted $CU_{nominal}$ scores negligible for the overall CU, especially when the standard deviations of feature distributions in a "true class' exceeds 1. In the case of ECOBWEB, its $CU_{numeric}$ measure is based on the approximation of the true probability of feature distributions in a class. When the approximation is close enough to the real probability, it not only does a better job in capturing the distinct differences in numeric feature values, but also renders the $CU_{nominal}$ scores derived from the corrupted nominal features negligible for the clustering process.

Results from Data with Gaussian Noise We again used the *misclassification count* on all partitional structures generated from the four clustering systems. The results are plotted in Figure 8. It is observed that the performance of SBAC does not change much from non-Gaussian to Gaussian noise, till the noise levels become very large(2σ). On the other hand, a significant improvement in performance is observed for AUTOCLASS. The performance of ECOBWEB with data having Gaussian or non-Gaussian noise follows almost exactly the same pattern: performance degrades as noise levels are increased, but improves slightly when noise levels become too high. Like SBAC and AUTOCLASS, there are only minor changes observed for the clusterings generated by COBWEB/3, when Gaussian noise is present.

In order to explain these results, we looked back to the criterion functions and assumptions employed by each system. For SBAC, the noise mitigation function takes effect the same way as it does for data with non-Gaussian noise. Since the nominal features of the data are well defined for each class, a big difference in the χ^2_d score is expected for pairs of objects from the same class versus pairs of objects from different classes. Only this time, the numeric features with low level of added Gaussian noises give a positive contribution to further distinguish apart objects of different class, i.e., when $\sigma_{noise} = \frac{1}{2}\sigma_{feature}$ and $\sigma_{noise} = \sigma_{feature}$. That

is because when the noise level is low, the relative distance of feature values does not change much, i.e., values from the same class are closer to each other than values from different classes. So the χ_c^2 score for objects from the same class is expected to be much higher than pairs of objects from different classes. Therefore, the overall ordering of the pairwise similarity values are not disturbed, and this ensured objects being assigned to classes that corresponded to the "model structure". When noise levels were further increased, a small number of feature values at the boundary of neighboring classes may mix together with values of a different class. This is when the contribution from the numeric features, χ_c^2 actually has an adverse effect on the overall similarity measure, which in turn causes misclassification of a small number of objects into its adjacent classes.

As was conjectured, the performance of AUTOCLASS improved greatly for data with Gaussian noise. When the noise level is low, i.e., $\sigma_{noise} = \frac{1}{2}\sigma_{feature}$ and $\sigma_{noise} = \sigma_{feature}$, the originally well distributed numeric features were again not affected greatly. The parameters of these slightly corrupted normal distributions can still be estimated by AUTOCLASS with sufficient confidence. So, even though the distributions of the nominal features do not satisfy AUTOCLASS's assumption completely, the class structures were preserved by the clustering process. When the noise level was increased for the numeric features, the originally well separated bell shaped distributions for the classes started overlapping, and the parameters estimated by AUTOCLASS for each distribution began deviating from their true values, causing a small number of fragmented groups, therefore, the increase in the misclassification value.

5 Conclusions

Limitations of earlier methodologies and criterion functions in dealing with data with mixed nominal and numeric features prompted us to look for a criterion function that would give better performance in clustering and discovery tasks. The similarity measure, initially proposed by Goodall [13], has been demonstrated to work well with mixed data sets. The measure assigns greater weight to feature-value matches that are uncommon in the population. For nominal-valued features, an uncommon feature value match is assigned a greater similarity value than a more common feature value match. For numeric-valued features, the uncommonality of feature value pairs is a function of the distances between pairs and the density of the population encompassed between the two values. The similarity value is inversely proportional to the distance between the values. However, for two pairs of values of equal distance, the one from the lower density distribution is assigned the higher similarity value. A common framework is defined for incorporating individual feature similarity values for both numeric and nominal features. Illustrative examples are presented to demonstrate the properties of the similarity measure. Efficient algorithms for computing the similarity values, especially for numeric features, has been developed.

The similarity measure is incorporated into an agglomerative clustering algorithm, SBAC, and its performance is studied on artificially generated data. Comparative studies on the artificial data reveal the limitations of AUTOCLASS, COBWEB/3, and ECOBWEB in working with mixed data. On the other hand, SBAC works well with mixed data and is robust to the introduction of different levels of noise in the data.

In previous work[1], we had discussed systematic discretization methods to generate uniform data description and avoid the limitations of the category utility measure in dealing with numeric features. This similarity measure goes a step further in that it retains the characteristics of probabilistic matching schemes, and includes useful information like the actual separation in values of the numeric features. By studying contributions of different features in the similarity contributions, one can perform better interpretation studies with SBAC and the Goodall measure. Our next step is to apply the SBAC system to real-world data where class structure is previously unknown, and to analytic discovery tasks like the one described in [17].

References

[1] G. Biswas, J. Weinberg, C. Li. "ITERATE: A Conceptual Clustering Method for Knowledge Discovery in Databases" in *Artificial Intelligence in the Petroleum Industry*, B.Braunschweig and R. Day, Editors. Editions Technip, 1995, chapter 3, pp. 111-139.

[2] P. Cheesman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, "Autoclass: A Bayesian Classification System", in *Proceedings of the Fifth International Conference on Machine Learning*, editor/program chair John Laird. Morgan Kaufmann Publisher, June 12-15, 1988.

[3] P. Cheesman and J. Stutz, "Bayesian Classification(AUTOCLASS): Theory and Results", in *Advances in Knowledge Discovery and Data Mining*, U.M. Fayyad, G.P. Shapiro, P. Smyth, and R. Uthurusamy, eds., AAAI/MIT Press, 1995, chapter 6.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm" in *Journal of the Royal Statistical Society Series B*, Vol 39(1),pp. 1-38.

[5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, New York, NY, 1973.

[6] D.H. Fisher. "Knowledge Acquisition via Incremental Conceptual Clustering," in *Machine Learning* Vol. 2, pp. 139-172. 1987.

[7] D.H. Fisher. "Iterative Optimization and Simplification of Hierarchical Clusterings", in *Journal of Artificial Intelligence Research*, Vol. 4, 1996, in press.

[8] D.H. Fisher and P.Langley. "Methods of Conceptual Clustering and Their Relation to Numeric Taxonomy", in *Artificial Intelligence and Statistics*, W. Gale, ed., Addison Weseley, Ready, MA, 1986.

[9] R.A. Fisher. *Statistical Methods for Research Workers*, 13th ed. Edinburgh and London: Oliver and Boyd, 1963.

[10] W.J. Frawley, G.P. Shapiro, and C.J. Matheus. "Knowledge Discovery in Databases: An Overview," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W.J. Frawley, eds., AAAI/MIT Press, Menlo Park, CA, pp. 1-27, 1991.

[11] J.H. Gennari, P. Langley, and D.H. Fisher. "Models of Incremental Concept Formation", in *Artificial Intelligence*, Vol. 40, pp. 11-61, 1989.

[12] M. Gluck and J. Corter. "Information, Uncertainty, and the Utility of Categories", in *Proceedings of the Seventh Annual Conference of the Cognitive Society*, pp. 283-287, Irvine, CA, 1985.

[13] D.W. Goodall. "A New Similarity Index Based On Probability" in *Biometrics*, Vol 22, pp. 882-907, 1966.

[14] S.J. Hanson and M. Bauer. "Conceptual Clustering, Categorization, and Polymorphy", in *Machine Learning*, Vol. 3, pp. 343-372, 1989.

[15] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*, Prentice Hall, Englewood Cliffs, 1988.

[16] H.O. Lancaster. "The Combining of Probabilities arising from data in discrete distributions" in *Biometrika*, Vol. 36, pp. 370-382, 1949.

[17] C. Li and G. Biswas, "Knowledge-based Scientific Discovery from Geological Databases" in *Proceedings of the First International Conference on Knowledge Discovery & Data Mining*, pp. 204-209, Montreal, Canada, August, 1995.

[18] K. McKusick and K. Thompson. "COBWEB/3: A Portable Implementation", Technical Report FIA-90-6-18-2, NASA Ames Research Center, June 20, 1990.

[19] Y. Reich. "Building and Improving Design Systems: A Machine Learning Approach". PhD thesis, Department of Civil Engineering, Carnegie Mellon University, 1991.

[20] Y. Reich and S.J. Fenves. "The Formation and Use of Abstract Concepts in Design", in *Concept Formation: Knowledge and Experience in Unsupervised Learning*, D.H. Fisher and M.J. Pazzani and P. Langley eds., Morgan Kaugmann, Los Altos, CA, 1991, pp. 323-353.