Designing Semantics-Preserving Cluster Representatives for Scientific Input Conditions

Aparna S. Varde[1,2], Elke A. Rundensteiner[1], Carolina Ruiz[1], David C. Brown[1,3], Mohammmed Maniruzzaman[2,3] and Richard D. Sisson Jr.[2,3]
1. Department of Computer Science
2. Center for Heat Treating Excellence
3. Department of Mechanical Engineering
Worcester Polytechnic Institute, Worcester, MA, USA
(aparna|rundenst|ruiz|dcb|maniruzz|sisson)@wpi.edu

ABSTRACT

In scientific domains, knowledge is often discovered from experiments by grouping or clustering them based on the similarity of their output. The causes of similarity are analyzed based on the input conditions characterizing a given type of output, i.e., a given cluster. This analysis helps in applications such as decision support in industry. Cluster representatives form at-a-glance depictions for such applications. Randomly selecting a set of conditions in a cluster as its representative is not sufficient since distinct combinations of inputs could lead to the same cluster. In this paper, an approach called DesCond is proposed to design semantics-preserving cluster representatives for scientific input conditions. We define a notion of distance for conditions to capture semantics based on the types of their attributes and their relative importance. Using this distance, methods of building candidate cluster representatives with different levels of detail are proposed. Candidates are compared using the DesCond Encoding proposed in this paper that assesses their complexity and information loss, given user interests. The candidate with the lowest encoding for each cluster is returned as its designed representative. DesCond is evaluated with real data from Materials Science. Evaluation with domain expert interviews and formal user surveys shows that designed representatives consistently outperform randomly selected ones and different candidates suit different users.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Performance, Design, Human Factors

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.
.

Keywords

Domain Knowledge, Visual Displays, Post-processing, Distance Metrics, Minimum Description Length, Decision Trees

1. INTRODUCTION

Clustering is often used to group data that involves a mixture of different types of attributes such a numbers and plain text. The data typically has semantics associated with it in the context of a given domain. Examples of such data include information on handheld PDAs (Personal Digital Assistants) [7], documents on web pages [15] and association rules derived from databases [11].

In this paper, we deal with such data coming from scientific experiments [1]. More specifically, we deal with the input conditions of experiments, where each condition gives the name and value of a process parameter and a set of such conditions forms the experimental input setup. These experiments are typically clustered based on their output and sets of input conditions leading to each cluster are identified to aid comparison of the corresponding processes [1]. Inferences drawn from clustering help in various applications such as parameter selection [14], computational estimation [18], simulation tools [12] and decision support systems [19].

Cluster representatives form concise depictions of each cluster in such applications. However, a randomly selected representative may not incorporate the necessary information in the cluster since distinct combinations of input conditions could lead to a single cluster. Moreover, different applications may need different levels of detail in the cluster. For example, presenting all the information in the cluster causes inefficiency in certain applications such as simulation tools [12] (elaborated in Section 2). In other applications such as visual displays for parameter selection [14] avoiding clutter is important. Hence it is advisable to design cluster representatives that preserve domain semantics in the context of specific applications.

In this paper, we propose an approach called DesCond to design semantics-preserving cluster representatives for scientific input conditions. A significant issue in the design is the notion of distance for the input conditions. These conditions have a mixture of different types of attributes such as categorical, numeric and ordinal. Each of them conveys a certain concept in the domain whose meaning needs to be captured. Moreover, the relative importance of the conditions in the domain also needs to be taken into account. In

this paper, we propose a distance function for input conditions that incorporates all these factors.

Using the given distance, candidate representatives are designed for each cluster showing increasing levels of detail. Three candidates are considered for each cluster, namely, a Single (set of) Conditions Representative closest to all other sets of conditions in the cluster, a Multiple (set of) Conditions Representative showing sub-clusters within the cluster, and an All (set of) Conditions Representative showing all information in the cluster abstracted suitably.

Candidate representatives are compared using an objective measure called the DesCond Encoding proposed in this paper. This encoding takes into account the complexity of each representative and the information loss due to it based on the interests of targeted users. The winning candidate based on the encoding is returned as the designed representative.

DesCond is evaluated using real data from the domain of Heat Treating of Materials [6] that inspired this research. Evaluations are conducted with domain expert interviews and with formal user surveys. Domain experts provide inputs to the DesCond Encoding reflecting interests of targeted users. The evaluation results show that the designed representatives are consistently better than randomly selected ones and that different candidates win in different applications. This is confirmed by the evaluations through user surveys where different categories of users prefer different designed representatives in applications such as parameter selection [14] decision support systems [19] and simulation tools [12]. Hence the output of DesCond is useful in designing the respective applications.

The rest of this paper is organized as follows. Section 2 gives a background of the Heat Treating domain with a motivating example for the given problem. Section 3 introduces the DesCond approach for designing representatives of input conditions while sections 4, 5 and 6 give its details. Section 7 summarizes the user evaluation. Section 8 overviews related work. Section 9 states the conclusions.

2. BACKGROUND AND MOTIVATION

We present a brief overview of the Heat Treating domain, since we will use examples from this domain throughout to explain the concepts in this paper. In Heat Treating, an important process is quenching, namely, the rapid cooling of a material in a liquid or gas medium to achieve desired mechanical and thermal properties [6]. Quenching is conducted with the following sets of input conditions.

- Quenchant Name: This refers to the cooling medium in the process, e.g., T7A, DurixolHR88A. Each cooling medium has properties such as viscosity and age that are characterized by the quenchant name.
- Part Material: This is the material of which the part being quenched is made e.g., ST4140, Inconel600. Each material has properties such as alloy content.
- Probe Type: A sample of the part called the probe is used for quenching. The probe has certain properties such as shape and dimension that are characterized by the probe type, e.g., CHTE probe, IVF probe.
- Oxide Layer: This states the presence and thickness of the oxide layer, whether absent, thin or thick.
- Agitation Level: This gives the extent to which the cooling medium is agitated during quenching, i.e., absent, low or high.
- Quenchant Temperature: This is the temperature at which the cooling medium is maintained during the quenching process. It is recorded in degrees Celsius.

Given this background, we present a motivating example. Consider the sets of conditions S_1 through S_9 in Example 1 showing a given cluster of experiments.

Example 1

- S_1: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (70-80), Probe Type = CHTE
- S_2: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (80-90), Probe Type = CHTE
- S_3: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = High, Oxide Layer = Any, Quenchant Temperature = (50-60), Probe Type = CHTE
- S_4: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = Low, Oxide Layer = None, Quenchant Temperature = (60-70), Probe Type = CHTE
- S_5: Quenchant Name = MarTemp355, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (20-30), Probe Type = CHTE
- S_6: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = Any, Oxide Layer = Thin, Quenchant Temperature = (60-70), Probe Type = CHTE
- S_7: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (60-70), Probe Type = CHTE
- S_8: Quenchant Name = MarTemp355, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (30-40), Probe Type = CHTE
- S_9: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (90-100), Probe Type = CHTE

All these sets of conditions in Example 1 lead to a similar experimental output, hence they have been assigned to the same cluster.

We now consider the application of simulation tools [12]. Users often run simulations of real experiments with a given set of input conditions. These simulations are typically as time-consuming as a real experiment (about 6 hours). They are preferred over a real experiment mainly because they save resources. Imagine that a randomly selected set of input conditions is displayed to the user as the output of

estimation. If the user runs a simulation using this representative, then ranges of information in the cluster are not captured, thus reducing the sample space of simulations. On the other hand if the user runs a simulation using a representative that conveys all the information in the cluster, it would take very long to run. Since each simulation takes approximately 6 hours with one set of input conditions, running it with 9 sets of conditions would take 54 hours, which is often not practical. Thus there is a need for a trade-off between the two extremes in such applications.

However, there are other applications where information loss is more critical while efficiency is not an issue, and vice versa. Thus there is a need to cater to various types of users. Hence it is necessary to design semantics-preserving cluster representatives in the context of targeted applications.

3. PROPOSED APPROACH: DESCOND

We propose an approach called DesCond to design a representative set of input conditions for each cluster. We first define the following terminology.

- Input Condition: This refers to an individual process parameter input to an experiment. Each condition is defined by an attribute value pair, e.g., *Part Material = ST4140*.
- Attribute: This gives name of each condition, e.g., *Part Material*.
- Value: This gives the content of each condition, e.g., *ST4140*.
- Set of Conditions: This refers to all the input conditions in a given experiment, e.g., *Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = low, Oxide Layer = none, Quenchant Temperature = (60-70)*.

The input to DesCond is clusters of experiments with all the sets of conditions characterizing each cluster. In our work [18], decision trees [16] are used to identify the combinations of conditions that characterize the clusters. All the decision tree paths leading to a given cluster of experiments are referred to as a cluster of conditions. Given this, the process of design is as follows.

The semantics of the domain is captured by defining a suitable a distance function for the set of conditions. Using this notion of distance, candidate representatives are designed for each cluster showing gradually increasing levels of detail. In the first level, the representative is selected from the original cluster as a single object (set of conditions) such that it forms the nearest neighbor for all other objects in the cluster. This candidate is called the *Single Conditions Representative (SCR)* . In the second level, a candidate known as the *Multiple Conditions Representative (MCR)* is constructed by forming sub-clusters within each original cluster using domain knowledge and the given notion of distance. In the third level, the candidate is constructed by combining all information in the cluster and abstracting it in a suitable form. This candidate is called the *All Conditions Representative (ACR)*.

The candidates are compared using a measure called the DesCond Encoding analogous to the Minimum Description Length principle [17]. This encoding takes into account the complexity of each representative measured as the number of data points stored for it and the information loss due to it measured as its distance from other objects in the cluster. The interests of targeted users based on the relative importance attached to the complexity and information loss are also taken into account in the encoding. The candidate giving the lowest value in the encoding is the winner and is returned as the designed representative. Note that there could be multiple winners based on the encoding, reflecting the corresponding user interests.

Thus, in our framework, the three main tasks in the design of domain-specific cluster representatives for conditions are as follows:

1. Defining a notion of distance for the set of conditions.
2. Obtaining candidate cluster representatives showing different levels of detail each capturing domain semantics.
3. Proposing an encoding to compare the candidates in order to find a suitable winner meeting specific application requirements.

These tasks are discussed in the following three sections.

4. NOTION OF DISTANCE

We consider three criteria in defining distance. The first one is the data type of each attribute as applicable to the domain. The second criterion is the distance between the individual attribute values defined in a domain-specific manner. The third one is the weight of each attribute based on its relative importance in the domain. These are explained as follows.

4.1 Data Types of the Attributes

The attributes describing the input conditions are of different types such as numeric, categorical and ordinal [10]. Categorical attributes are of the character or string type and store descriptive information. Numeric attributes represent data that is of the integer or real number type. Ordinal attributes are those whose values are also of the string and character type but store information where the order matters.

The types of attributes applicable to the Heat Treating datasets in our problem are listed below. For ordinal attributes, their possible values are also stated. Each attribute represents an individual input condition in Heat Treating.

- Quenchant Name (QN): categorical
- Part Material (PM): categorical
- Probe Type (PT): categorical
- Oxide Layer (OL): ordinal — none, thin, thick
- Agitation Level (AL): ordinal — absent, low, high
- Quenchant Temperature (QT): numeric

4.2 Distance between the Attribute Values

We use the sets of conditions shown in Example 1 in order to explain the calculation of distance for each type of attribute.

4.2.1 Categorical Attributes

For categorical attributes, the distance is defined as 0 if the attribute values are identical and 1 if they are not identical. Hence the distance is calculated as:

$D_{categorical}(S_i, S_j) = 0$ if $v_i = v_j$ and $D_{categorical}(S_i, S_j) = 1$ if $v_i <> v_j$ where S_i and S_j are the respective sets of conditions, while v_i and v_j are the respective values of the given categorical attribute.

Thus, considering the categorical attribute Part Material and referring to Example 1, we calculate distance between the Part Material values as $D_{PM}(S_1, S_3) = 1$, and $D_{PM}(S_1, S_2) = 0$, since Part Material values are not equal in the sets of conditions S_1 and S_3, while they are equal in S_1 and S_2.

4.2.2 Numeric Attributes

For numeric attributes, distance is calculated as the absolute difference of their attribute values. If the values are grouped into ranges as a data pre-processing step, then we consider the difference between the mean values of the respective ranges. Suitable scaling factors are applied if needed to maintain parity with other attributes. Thus, distance for numeric attributes is calculated as:

$D_{numeric}(S_i, S_j) = SF \times |v_i - v_j|$ where S_i and S_j are the respective sets of conditions, v_i and v_j are the values (or mean value of ranges) of the respective numeric attributes, and SF is a scaling factor based on domain knowledge.

Thus in Example 1, for the numeric attribute Quenchant Temperature with scaling factor $SF = 1/10$ (given in the domain) we get distances between Quenchant Temperature values as $D_{QT}(S_1, S_2) = 1$ and $D_{QT}(S_1, S_3) = 2$.

4.2.3 Ordinal Attributes

For ordinal attributes, the distance is calculated as the absolute difference between their values after the values are mapped to numeric based on their order. For example, Agitation values of *high*, *low* and *absent* are mapped to 3, 2 and 1 respectively. The mapping is a data preprocessing step. Distance for ordinal attributes is then given as:

$D_{ordinal}((S_i, S_j) = |v_i^{'} - v_j^{'}|$ where S_i and S_j are the respective sets of conditions, while $v_i^{'}$ and $v_j^{'}$ are numeric values to which the respective ordinal values are mapped.

In Example 1 therefore, for the ordinal attribute Agitation Level, distance is calculated as $D_{AL}(S_1, S_2) = 3-3 = 0$ and $D_{AL}(S_3, S_4) = 3 - 2 = 1$.

4.2.4 Distance for Set of Conditions

Given these distances for the attribute types, the distance function D_{cond} for the set of conditions is then defined in terms of the distances between individual attribute values and the weights of the respective attributes as follows:

$D_{cond} = \Sigma_{i=1}^{A} W_i \times D_i$ where each D_i is a distance function for the individual attributes, each W_i is a weight giving the relative importance of the corresponding attribute and A is the total number of attributes. The weights are obtained as explained in the next subsection.

4.3 Weights of the Attributes

As stated earlier, in our problem decision trees [16] are used to learn the relative importance of the conditions characterizing each cluster with respect to the domain. Hence the decision tree paths are used to derive the weights of the attributes depicting these conditions. The reasoning behind the method for deriving the weights is as follows.

1. An attribute is considered to have a higher weight than other attributes if it is at a higher level in the decision tree. This is because the root of the tree represents the most significant input condition while the lower levels represent less significant conditions. Also, attributes not identified in the decision tree represent insignificant conditions for the given data sample.

2. The shorter the path in which an attribute appears, the higher is the significance of that attribute. This is because a shorter path with fewer attributes is more definite in classifying the data than a longer path. An extreme of this would be one particular value of the root leading directly to a given cluster. For example, if all data pertaining to $QuenchantName = T7A$ belongs to *Cluster C*, irrespective of other attributes, then in this path Quenchant Name should get a higher weight than in a path having other attributes such as Part Material and Agitation Level.

3. The greater the number of experiments in the cluster corresponding to a path, the more important is that path and hence an attribute appearing in that path. This is because the given path then classifies a greater amount of data.

We draw an analogy with the decision tree induction algorithms such as ID3 and J4.8 [16] in this reasoning. It is not feasible to directly use the weights from these algorithms, because the weights are different in each epoch and we need one uniform set of weights for the attributes. Moreover, if we were to use their weights we would need to define a constant of proportionality which is not known apriori. Also, running the ID3/J4.8 epochs again on the same dataset is likely to be inefficient, given that the tree has already been constructed. Thus, we use the analogy behind the induction of decision trees.

Given these considerations and applying the reasoning above, a heuristic for the weights of the attributes in the decision tree is defined as below.

Decision Tree Weight Heuristic

$W_i = \frac{1}{P}\Sigma_{j=1}^{P}\frac{H_{i,j}}{H_j} \times G_j$
where, W_i = weight of each attribute,
P = total number of paths in the decision tree,
G_j = number of graphs in the cluster of path j,
$H_{i,j}$ = height of node for attribute i in path j and
H_j = height of path j
such that, "height" H is defined number of nodes away from the leaf.

Thus, in a given path the leaf has a height of 0, the node immediately above the leaf has a height of 1 and so forth. The height of a path is basically the height of its root node.

The use of the decision tree heuristic in calculating weights is explained in Example 2 using the partial decision tree shown in Figure 1.

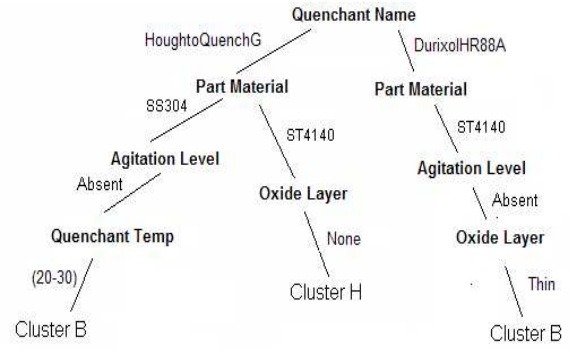

Figure 1: Partial Decision Tree

Example 2

For the given partial decision tree, assume that Cluster B has 10 experiments and Cluster H has 5 experiments. Then we get the following weights.
Quenchant Name: $W_{QN} = \frac{1}{3}(\frac{4}{4}\times 10+\frac{3}{3}\times 5+\frac{4}{4}\times 10) = 8.33$
Part Material: $W_{PM} = \frac{1}{3}(\frac{3}{4}\times 10+\frac{2}{3}\times 5+\frac{3}{4}\times 10) = 5.44$
Agitation Level: $W_{AL} = \frac{1}{3}(\frac{2}{4}\times 10+0+\frac{2}{4}\times 10) = 3.33$
Oxide Layer: $W_{OL} = \frac{1}{3}(0+\frac{1}{3}\times 5+\frac{1}{4}\times 10) = 1.39$
Quenchant Temperature: $W_{QT} = \frac{1}{3}(\frac{1}{4}\times 10+0+0) = 0.83$
Probe Type: $W_{QT} = \frac{1}{3}(0+0+0) = 0$

We will use the weights derived from the corresponding *complete* decision tree in this example in order to illustrate the design of the candidate cluster representatives. The weights of the attributes inferred from the *complete* tree (whose partial snapshot is shown in Figure 1) are as follows.

- Quenchant Name (QN): weight $W_{QN} = 8.12$
- Part Material (PM): weight $W_{PM} = 5.97$
- Agitation Level (AL) : weight $W_{AL} = 3.05$
- Oxide Layer (OL): weight $W_{OL} = 2.08$
- Quenchant Temperature (QT): weight $W_{QT} = 0.81$
- Probe Type (PT): weight $W_{PT} = 0$

Thus the distance function derived from the complete tree is: $D_{cond} = 8.12\times D_{QN}+5.97\times W_{PM}+3.05\times D_{AL}+2.08\times D_{OL}+0.81\times D_{QT}$
where the individual distances D_{QN}, D_{PM} and so forth are calculated based on the values and types of the individual attributes. Given the manner in which it is derived, this distance function incorporates domain semantics and can be used for the design of candidate cluster representatives.

5. LEVELS OF DETAIL

We consider the following levels of detail in designing the candidate representatives.

- *Level 1: SCR - Single Conditions Representative*, i.e., one set of conditions closest to all others in the cluster using the giving notion of distance
- *Level 2: MCR - Multiple Conditions Representative*, i.e., multiple sets of conditions summarizing cluster information with respect to the domain.

Quenchant Name	Part Material	Agitation Level	Oxide Layer	Quenchant Temp	Probe Type
DurixolV35	ST4140	High	Any	(50-60)	CHTE

Figure 2: SCR for Example 1

- *Level 3: ACR - All Conditions Representative*, i.e., all possible sets of conditions in the cluster abstracted using domain knowledge.

The process of designing each of these is explained below. In order to illustrate the concepts, we consider Example 1 showing all the sets of conditions, i.e., decision tree paths leading to a given cluster. These paths are obtained from the complete decision tree over the given data set. Using the distance function derived from the complete decision tree, candidate representatives are designed as follows.

5.1 Single Conditions Representative

The Single Conditions Representative, SCR, is one of the original set of conditions in the given cluster. Using the distance function for conditions that incorporates domain semantics, SCR is selected as the set of conditions closest to all others in the cluster. It other words SCR is such that the sum of its distances from all other sets of conditions in the cluster is the least. The SCR for the cluster in Example 1 is shown in Figure 2.
The Single Conditions Representative is designed in order to show the most important cluster information in a concise form. It is useful in applications where the user is interested in finding out the best possible set of input conditions that would give a desired nature of output.

5.2 Multiple Conditions Representative

This Multiple Conditions Representative, MCR, summarizes the information in the cluster and is constructed as follows. The set of conditions in each cluster are grouped into sub-clusters based on the similarity of the conditions. The notion of similarity for sub-clustering the conditions is the distance function D_{cond} defined earlier.
The number of sub-clusters for each cluster is determined based on domain knowledge. For example, in Heat Treating we have the following information.

- *Quenchant Name* is the root of the tree and gets a higher weight than other attributes in the distance function.
- One important purpose of conducting the quenching experiments in Heat Treating is to categorize the quenchants.
- *Quenchant Name* has more distinct values than the other attributes closer to the root.

Based on this knowledge, the number of sub-clusters is set equal to the number of distinct values of Quenchant Name. In other words, the sets of conditions in each cluster are grouped into sub-clusters based on the similarity of their Quenchant Names.
Sub-clustering is then done using any suitable clustering algorithm using D_{cond} as the notion of distance [10]. For each sub-cluster, a representative is selected as the set of conditions closest to all the others in the sub-cluster. Likewise, representatives are obtained for each sub-cluster. The

Quenchant Name	Part Material	Agitation Level	Oxide Layer	Quenchant Temp	Probe Type
DurixolV35	ST4140	High	Any	(50-60)	CHTE
DurixolW72	SS304	High	None	(60-100)	CHTE
MarTemp355	SS304	High	None	(20-40)	CHTE

Figure 3: MCR for Example 1

Quenchant Name	Part Material	Agitation Level	Oxide Layer	Quenchant Temp	Probe Type
DurixolV35	ST4140	High	Any	(50-60)	CHTE
DurixolV35	ST4140	Low	None	(60-70)	CHTE
DurixolV35	ST4140	Any	Thin	(60-70)	CHTE
DurixolW72	SS304	High	None	(60-100)	CHTE
MarTemp355	SS304	High	None	(20-40)	CHTE

Figure 4: ACR for Example 1

Multiple Conditions Representative is an aggregation of all sub-cluster representatives displayed in a tabular form. The MCR for Example 1 is shown in Figure 3.

The Multiple Conditions Representative is designed because it depicts a trade-off between the amount of detail displayed to the user and the amount of information captured within the cluster. It is useful in applications where the user wishes to find out, for example, distinct combinations of the most significant condition that would give a desired nature of output.

5.3 All Conditions Representative

The All Conditions Representative, ACR, is designed to capture all the data in the cluster with no information loss. It is built by retaining all the original sets of conditions and displaying them sorted in ascending order of the most significant attribute, followed by the next significant one and so forth. The significance of the attributes is determined based on the distance function D_{cond}. The values of each set of conditions are abstracted using domain knowledge wherever possible. For example, in Heat Treating, if three sets of conditions are identical except that the value of *Agitation Level* is *absent* for one, *low* for another and *high* for the third, then this is abstracted as *Agitation* = *any*, where *any* refers to any possible value of agitation applicable to the domain. Likewise, if two sets of conditions are identical except that *Quenchant Temperature* has two consecutive ranges $(110-120)$ and $(120-130)$, then these are abstracted into a single set of conditions with *Quenchant Temperature* $= (110-130)$. This is in order to avoid visual clutter, while still displaying all information in the cluster.

The All Conditions Representative is an aggregation of all the sets of conditions sorted in ascending order from the most to the least significant. The ACR for Example 1 is shown in Figure 4.

The All Conditions Representative is designed so as to convey all the information in the cluster in an organized manner. It is useful in applications where the user is interested in studying in detail all the possible inputs that would lead to a given nature of output.

Thus, three types of candidate representatives are designed for each cluster.

6. COMPARISON OF CANDIDATES

The candidate representatives are compared using an analogy with the Minimum Description Length (MDL) principle. The MDL principle proposed by Rissanen [17] aims to minimize the sum of encoding the theory and the examples using the theory. In the literature, when MDL is used to encode cluster information, it is essential to be able to recover the original cluster from the encoding. However, in the context of our problem, we do not need to retrieve the cluster. Instead, we need to compare the cluster representatives with each other in order to evaluate them. Hence we propose a measure for comparison that is analogous to the Minimum Description Length of the cluster.

Our proposed measure is called the *DesCond Encoding*. In our context, the theory (with respect to MDL) refers to the cluster representatives while the examples refer to all the other objects in the cluster. We take into account the complexity of each representative and the information loss due to it. Complexity refers to the ease of interpretation which is measured as the amount of data stored for the representative. Information loss refers to the capacity of the representative in capturing information within the cluster and is measured as the distance of the representative from all the objects in the cluster. The relative importance attached to the two terms of complexity and distance (information loss) is also taken into account in the encoding, based on the interests of targeted users. Given this, the DesCond Encoding is described below.

The DesCond Encoding

$En_c = UBC \times log_2(AV) + UBD \times log_2 \frac{1}{s} \Sigma_{i=1}^{s} D(R, S_i)$

where, En_c = encoding for conditions,

A = number of attributes in the representative,

V = number of values for each attribute in the representative,

R = cluster representative,

S_i = each set of conditions in cluster,

$D(R, S_i)$ = distance between representative and every set of conditions using the given distance function,

s = total number of sets of conditions in cluster,

UBC = percentage weight giving user bias for complexity,

UBD = percentage weight giving user bias for distance.

The first term in this encoding $log_2(AV)$ denotes the complexity of the representative. This is calculated as the number of attributes and values that need to be stored for that representative. The second term, i.e., the distance term $log_2 \frac{1}{s} \Sigma_{i=1}^{s} D(R, S_i)$ denotes the information loss due to the representative. It is calculated as the average distance of the representative from all the other sets of conditions in the cluster. The terms UBC and UBD are the percentage weights assigned to the complexity and distance terms respectively in order to give the user bias for those two terms. Unless otherwise specified, equal weights are assigned to complexity and distance, i.e., 50% each.

Candidate cluster representatives are evaluated using the DesCond Encoding. The representative with the lowest value of the encoding for the given cluster is considered the best and is returned as its designed representative.

7. USER EVALUATION

DesCond is implemented in Java and evaluated using real data from the Heat Treating domain [6]. Evaluation is conducted with domain expert interviews and with formal user surveys. Each of these is described below.

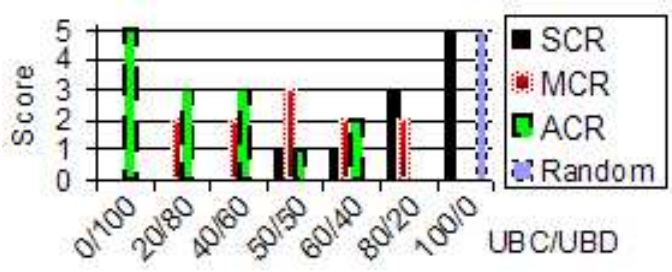

Figure 5: Results for Small Data Set

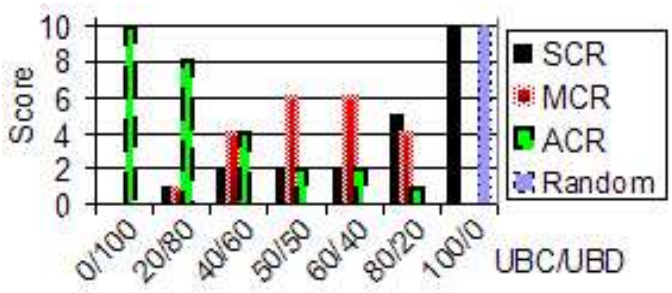

Figure 6: Results for Medium Data Set

7.1 Evaluation with Domain Expert Interviews

7.1.1 Method of Evaluation

In this evaluation, domain experts provide different user bias weights in the DesCond Encoding based on their notions of targeted user interests. Using these weights candidate representatives are evaluated. Different datasets consisting of Heat Treating experiments placed into clusters are sent as input to DesCond. Parameters altered in DesCond besides the user bias weights are dataset size and number of clusters. Any suitable algorithm such as k-means [13] is used to generate the clusters over the datasets. In addition to altering the values of k, i.e., number of clusters, the clustering seeds are also altered to provide randomization. Given these clusters as input, the output of DesCond is the winning candidate for each cluster.

For comparison, a random representative is considered per cluster in the evaluation process. Scores are then assigned to each representative as the number of clusters in the given dataset in which it is the winner. For example, in a dataset of 25 experiments placed in 5 clusters with (50/50) weights, if the winner is SCR for one cluster and ACR for four, then the scores are, SCR:1, MCR:0, ACR:4 and Random:0. The results are reported accordingly.

We show the evaluation results with a small dataset of 25 Heat Treating experiments placed in 5 clusters, a medium dataset of 150 experiments in 10 clusters and a large dataset of 400 experiments in 20 clusters. We consider 7 different user bias weights in the DesCond Encoding spreading over various possible applications as identified by experts. Results are reported as scores for representatives in Figures 5, 6 and 7 respectively.

7.1.2 Observations and Discussion

- For (20/80) weights, the All Conditions Representatives generally win. Such weights are likely to occur in applications such as intelligent tutoring systems [4]. In such systems it is important to study all information in the cluster to analyze process behavior in detail. Complexity of the representative does not matter as much. Thus ACRs would be useful here.

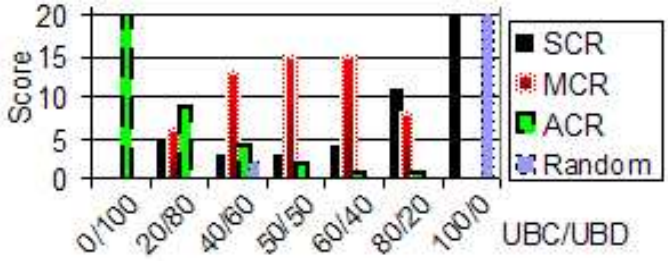

Figure 7: Results for Large Data Set

- For (50/50) weights, the Multiple Conditions Representatives win for most data sets. These would probably be useful in simulation applications [12] where a trade-off between complexity and information loss is needed.

- For (80/20) weights, the Single Conditions Representatives are often winners. These would most likely be useful in applications such as parameter selection [14]. Here a representative is used to analyze the behavior of a cluster to compare processes for selecting process parameters in industry. Thus a simple representative is good and hence SCRs are useful especially for large data sets.

- For the (40/60) and (60/40) weights, All Conditions Representatives win for the small dataset while Multiple Conditions Representatives win with or without a tie for the medium and large datasets. These would likely also be useful in various simulation applications where the user bias could tilt more or less in favor of complexity and distance, still requiring a trade-off.

- Random representatives lose in most cases. This indicates that designed representatives consistently outperform random ones in our targeted applications.

7.2 Evaluation with Formal User Surveys

DesCond is developed in the context of our larger project, the AutoDomainMine system [18] that performs computational estimation. In this system the designed representatives are used to estimate the results of experiments given their input conditions and to estimate the input conditions that would obtain a given result

Hence the effectiveness of the designed representatives in estimation is assessed through formal surveys conducted by the prospective users of this system. Users execute tests comparing the estimation of AutoDomainMine with real laboratory data not used for training. For every test, if the estimation provided by AutoDomainMine matches the real data, then the users report the test as accurate, else inaccurate. Accuracy of the system is then computed as the percentage of accurate tests over all the tests conducted.

In each test executed by users, the designed representatives are compared with each other in terms of how effective they are in displaying information in the applications

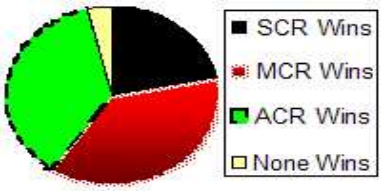

Figure 8: Winners in Computational Estimation

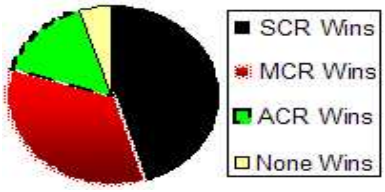

Figure 9: Winners in Parameter Selection

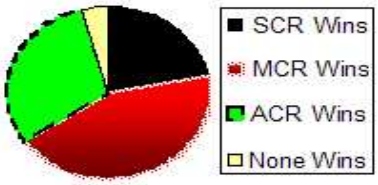

Figure 10: Winners in Simulation Tools

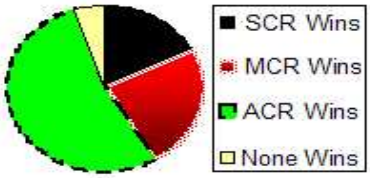

Figure 11: Winners in Tutoring Systems

of AutoDomainMine. The applications are parameter selection [14], simulation tools [12], intelligent tutoring systems [4] and decision support systems [19]. In order to perform this evaluation, the estimated output of AutoDomainMine is displayed to the users in three different levels of detail, as the Single Conditions Representative, the Multiple Conditions Representative and the All Conditions Representative respectively. Different categories of users are asked to choose which display (designed representative) best meets their needs with respect to the given application. We summarize the results of the surveys with respect to different applications.

7.2.1 Computational Estimation

The survey results in this category are for the AutoDomainMine system as a whole indicating the effectiveness of the designed representatives in computational estimation [18]. The users conducted 100 tests in this category. Figure 8 shows a pie chart giving the distribution of winners among the candidate representatives. In this pie chart the region corresponding to *None Wins* shows the inaccurate estimations. The estimation accuracy is observed to be 94%.

From Figure 8, it is seen that for computational estimation, All Conditions Representatives and Multiple Conditions Representatives are winners in most tests executed by users, with Single Conditions Representatives trailing closely behind. Since computational estimation has a broad range of users, different types of representatives are found to win.

7.2.2 Parameter Selection

In these applications, the output of DesCond is used to select process parameters in industry [14]. The users conducted 53 tests in this category. The winners in these applications are shown in the pie chart in Figure 9. The *None Wins* region in this chart (and charts in the following applications) indicates the tests where none of the candidate representatives were found suitable by the users.

As observed in Figure 9, Single Conditions Representatives are the winners for most tests. The reason for this likely would be that in process parameter selection, typically most users want one right answer.

7.2.3 Simulation Tools

In simulation tools, the users need the cluster representatives to run computer simulations of a real laboratory experiment [12]. The simulation users conducted 62 tests with DesCond. Figure 10 shows the winning candidates in these applications.

From Figure 10, it is seen that Multiple Conditions Representatives are the winners in most tests. This is probably because simulation tool users generally want to use ranges of information in order to increase the sample space of the simulations, but they also care about complexity since simulations are time-consuming. Hence, we find that they prefer the MCRs.

7.2.4 Intelligent Tutoring Systems

Intelligent tutoring systems are used to study in detail the behavior of processes analogous to classroom study on the given topic [4]. Totally 37 tests were conducted by users in this category. Figure 11 shows what type of representatives suited the users of these applications.

From Figure 11, it is clear that in most cases All Conditions Representatives are the winners. This is most likely due to the fact that in most intelligent tutoring applications, users are interested in learning more details about the system and do not care much about complexity. Hence, more detail is appreciated.

7.2.5 Decision Support Systems

Decision support systems [19] are used for various purposes. In high level business decision support, at-a-glance retrieval of information is important without much emphasis on detail. Some decision support users however, focus on process optimization and need to scrutinize information in more detail. We had 44 tests conducted by decision support system users. The distribution of winning candidates in decision support systems in shown in Figure 12.

From Figure 12, it is found that there is a fairly good mix of winners in these applications. This is because different decision support users are interested in different levels of detail. Hence it would be desirable to retain all the representatives in designing such applications, and to display information in increasing levels of detail.

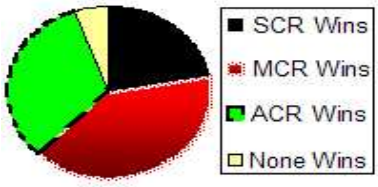

Figure 12: Winners in Decision Support Systems

7.2.6 Discussion on User Surveys

The following conclusions can be drawn from the results of the user surveys.

- The use of the designed representatives enhance the estimation accuracy to 94% in AutoDomainMine [18]. This is higher than the earlier version of the system that used randomly selected representatives for estimation giving an accuracy of 87%.
- The results of the formal user surveys agree with the results of the evaluation conducted with domain expert interviews using the DesCond Encoding. For example, Multiple Conditions Representatives win with (50/50) weights in the encoding. These representatives are also the winners in simulation applications [12] which require a trade-off between complexity and information loss.
- All the designed representatives are useful (more or less) in computational estimation [18] and decision support applications [19]. Hence in designing these applications all of them would be retained, displaying the information in three different levels of detail.
- Single Conditions Representatives are most useful in parameter selection [14], Multiple Conditions Representatives in simulation tools [12] and All Conditions Representatives in intelligent tutoring systems [4]. Hence in designing the systems for the corresponding applications these representatives would be used respectively.

8. RELATED WORK

In recent years there has been much interest in abstracting information that involves plain text, semi-structured text and so forth. Personal Digital Assistants (PDAs) often have displays in levels of detail. In [7] an approach is described for building representatives consisting of Semantic Textual Units (STUs) with paragraphs, sentences etc. Each STU is revealed gradually in terms of keywords, a single sentence, the first paragraph and the whole STU. In [15] An approach for text summarization over the web is proposed based on constructing representatives by exploiting diversity concepts in text. They take into account probability of occurrence of words, the type of grammatical constructs (nouns, verbs etc.) and the number of documents. However, neither [7] nor [15] propose objective evaluation measures to compare representatives based on user interests.

There is also work on similarity measures over character data. In [8] they consider similarity between categorical attributes not only based on the values of the given attributes but also based on the values of other attributes that are inter-dependent. In [9] they present the Iterated Contextual Distances algorithm which learns distances between attributes taking into account such inter-dependencies. However, the kind of inter-dependencies that they define do not exist in our datasets. Learnable similarity measures for strings are presented in [5] based on support vector machines and expectation maximization and applied for duplicate detection. However, they deal with natural language text strings and the involved semantics, while our data is different. We work with domain-specific input conditions that involve a mixture of attributes such as numeric, categorical and ordinal. We do not deal with strings of text whose meaning has to be interpreted in a broader natural language context. Moreover, in our context domain knowledge has already been derived from decision trees and can directly be applied to define a distance function for the conditions without further learning.

The Minimum Description Length (MDL) principle has been used in the context of clustering. In our earlier work [20] we propose an MDL-based effectiveness measure for cluster representatives of graphs. However, those graphs are images storing two-dimensional plots of numbers while in this paper we focus on input conditions involving different types of attributes. Hence the semantic issues are different. In the literature, clustering of association rules has been proposed in [11] and an MDL encoding is proposed to evaluate the clusters. In [3] they propose an MDL encoding as an objective evaluation criterion for clustering. However, these encodings are not used to evaluate different types of cluster representatives. Also, in their work they need to retrieve the original cluster from the encoding which is not a requirement in our context.

In [2] they discover knowledge from simulators. They identify regions in the input space that lead to a certain type of output behavior. Since the cost of simulations is high, they develop automated methods for for efficient knowledge discovery. They focus on which simulations to run next by using Support Vector Machines. However, their data is numeric while ours has different types of attributes such as categorical, numeric and ordinal. Also, they do not build and evaluate different types of representatives. Their focus is on knoweledge discovery while we focus on the display of information as well. Hence we develop methods to suit our problem.

9. CONCLUSIONS

In this paper an approach called DesCond is proposed to design semantics-preserving cluster representatives over input conditions of scientific experiments. Using a domain-specific notion of distance for conditions, candidate representatives are designed for each cluster showing increasing levels of detail. Candidates are compared using the DesCond Encoding analogous to the Minimum Description Length principle. The winning candidate for each cluster is returned as its designed representative. DesCond is evaluated with real data from Heat Treating. In evaluations conducted with domain expert interviews using the DesCond Encoding, designed representatives are observed to be consistently better than random ones and different designed representatives win in different applications. In the formal user evaluation surveys, it is found that different categories of users like different designed representatives. The output of DesCond is useful in developing the corresponding applications. User

surveys also indicate that DesCond improves the estimation accuracy of the AutoDomainMine computational estimation system that motivated its development.

10. ACKNOWLEDGMENTS

This work is supported by the Center for Heat Treating Excellence (CHTE) and its member companies and by the Department of Energy - Industrial Technology Program (DOE-ITP) Award Number DE-FC-07-01ID14197.

The authors thank all the Heat Treating users who spent their precious time in completing the user evaluation surveys. We also thank the organizers of the MPI (Metal Processing Institute) Spring 2006 Seminar that gave us the opportunity for system demonstration to set the stage for the user surveys.

The feedback of the Quenching Research Group in the Department of Materials Science and of the Database Systems Research Group (DSRG), the Artificial Intelligence Research Group (AIRG) and the Knowledge Discovery and Data Mining Research Group (KDDRG) in the Department of Computer Science at WPI is gratefully acknowledged.

11. REFERENCES

[1] D. Askeland and P. Phule. *Essentials of Materials Science and Engineering*. Thomson Publishers, Toronto, Canada, 2004.

[2] M. Burl, D. DeCoste, B. Enke, D. Mazzoni, W. Merline, L. Scharenbroich. Automated Knowledge Discovery from Simulators. In *SDM*, pages 82–93. SIAM, April 2006.

[3] A. Banerjee and J. Langford. An Objective Evaluation Criterion for Clustering. In *KDD*, pages 515–520. ACM, August 2004.

[4] D. Bierman and P. Kamsteeg. *Elicitation of Knowledge for Intelligent Tutoring Systems*. Technical report, University of Amsterdam, Netherlands, 1988.

[5] M. Bilenko and R. Mooney. Adaptive Duplicate Detection using Learnable String Similarity Measures. In *KDD*, pages 39–48. ACM, August 2003.

[6] H. Boyer and P. Cary. *Quenching and Control of Distortion*. ASM International, Metals Park, OH, 1989.

[7] O. Buyukkokten, H. Garcia-Molina and A. Paepcke. Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. In *WWW*, pages 652–662. May 2001.

[8] G. Das, H. Mannila and P. Ronkainen. Similarity of Attributes by External Probes. In *KDD*, pages 23–29. ACM, August 1998.

[9] G. Das and H. Mannila. Context-Based Similarity Measures for Categorical Databases. In *PKDD*, pages 201–210. Springer, September 1998.

[10] J. Han and M. Kamber. *Data Mining Concepts and Techniques*. McGraw-Hill Companies, 2001.

[11] B. Lent, A. Swami and J. Widom. Clustering Association Rules. In *ICDE*, pages 220–231. IEEE, April 1997.

[12] Q. Lu., R. Vader, J. Kang and Y. Rong. Development of a Computer-Aided Heat Treatment Planning System. *Heat Treatment of Metals*, 3:65–70, 2002.

[13] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Mathematical Statistics and Probability*, 1:281–297, 1967.

[14] M. Maniruzzaman, J. Chaves, C. McGee, S. Ma and R. Sisson Jr. The CHTE Quench Probe System: A New Quench Characterization System. In *ICFDM*, pages 13–17. July 2002.

[15] T. Nomoto and Y. Matsumoto. A New Approach to Unsupervised Text Summarization. In *SIGIR*, pages 26–34. ACM, September 2001.

[16] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[17] J. Rissanen. Stochastic Complexity and the MDL Principle. *Econometric Reviews*, 6:85–102, 1987.

[18] A. S. Varde, E. A. Rundensteiner, C. Ruiz, D. C. Brown, M. Maniruzzaman and R. D. Sisson Jr. Integrating Clustering and Classification for Estimating Process Variables in Materials Science. In *AAAI Extended Abstract and Poster Track*, AAAI, July 2006.

[19] A. S. Varde, M. Takahshi, E. A. Rundensteiner, M. Ward, M. Maniruzzaman and R. D. Sisson Jr. QuenchMiner^{TM}: Decision Support for Optimization of Heat Treating Processes. In *IICAI*, 993–1003. IEEE, December 2003.

[20] A. S. Varde. *Graphical Data Mining for Computational Estimation in Materials Science Applications*. PhD thesis, Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA, 2006.