Clustering Mixed Numerical and Uncertain Categorical Data with M-BILCOM: Significance Metrics on a Yeast Example

Bill Andreopoulos

Aijun An

Xiaogang Wang

Technical Report CS-2005-03

March 2005

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

Clustering Mixed Numerical and Uncertain Categorical Data with M-BILCOM: Significance Metrics on a Yeast Example

Bill Andreopoulos
Department of Computer Science, York University, Toronto, Canada, M3J1P3
billa@cs.yorku.ca

Aijun An
Department of Computer Science, York University, Toronto, Canada, M3J1P3
aan@cs.yorku.ca

Xiaogang Wang
Department of Mathematics and Statistics, York University, Toronto, Canada, M3J1P3
stevenw@mathstat.yorku.ca

ABSTRACT
We have designed the M-BILCOM clustering tool for mixed numerical and categorical data sets, where the categorical attribute values (CAs) are not certain to be correct and have associated confidence values (CVs) from 0.0 to 1.0 to represent their certainty of correctness. M-BILCOM performs bi-level clustering of mixed data sets that resembles a Bayesian process. We have applied M-BILCOM to yeast data sets where the CAs were perturbed randomly and CVs were assigned indicating the confidence of correctness of the CAs. On such mixed data sets M-BILCOM outperforms other clustering algorithms, such as AutoClass. We have applied M-BILCOM to real numerical data sets from gene expression studies on yeast, incorporating CAs representing Gene Ontology annotations on the genes and CVs representing Gene Ontology Evidence Codes on the CAs. We have applied novel significance metrics to the CAs in resulting clusters, to extract the most significant CAs based on their frequencies and their CVs in the cluster. For genomic data sets, we have used the most significant CAs in a cluster to predict gene function.

Software
http://www.cs.yorku.ca/~billa/MULIC/

Keywords
Clustering, categorical, numerical, low quality, confidence, significance metric, gene ontology, evidence code, function prediction.

1. INTRODUCTION
Clustering attempts to partition a set of data objects into groups, so that objects with similar characteristics are grouped together and different groups contain objects with different characteristics. A high quality clustering tool produces clusters with *high intra-class* similarity and *low inter-class* similarity. [11, 13, 15, 17].

We designed the M-BILCOM clustering tool for numerical data sets that incorporates in the clustering process CAs and CVs indicating the confidence that the CAs are correct. M-BILCOM was mainly inspired by numerical gene expression data sets from DNA microarray studies, where CAs and CVs can be derived from Gene Ontology annotations and Evidence Codes [4-10, 12, 14, 21-22]. One of the main advantages of our algorithm is that it offers the opportunity to apply novel significance metrics for spotting the most significant CAs in a cluster when analyzing the results [3]. In genomic data sets, our significance metrics allow significant CAs to be extracted from a cluster based on their CVs and their frequencies and to be used for predicting the functions of other genes in the cluster. This provides a different insight for predicting gene function by giving the 'full picture' of the data set, because the significant CAs are extracted from genes that may have been appended to the cluster on the basis of numerical or categorical similarity or both.

Our approach offers several advantages over other approaches:

- Our clustering algorithm may cluster data sets where all genes have numerical annotations but not all genes have CAs. Each CA has a CV associated - a real number between 0.0 and 1.0 - indicating our confidence about its correctness.
- During the clustering process, our method starts from CAs and CVs at the lower level and then moves to numerical clustering at a higher level. The CAs and CVs are actually used in the clustering process, instead of just annotating the clusters afterwards [1, 3]. The method of Wu et al. as applied previously to high-throughput biological data, starts from the numerical clustering, then adds CAs at a higher level and finally CVs are calculated (P-values) [25].
- During the clustering process, objects having CAs with high confidence to be correct, get clustered by putting more weight on the categorical similarity and less weight on the numerical similarity. On the other hand, objects having CAs with low confidence to be correct get clustered by putting more weight on numerical similarity [1].
- Our clustering algorithm allows us to define significance metrics indicating the significance of a CA in a cluster. Such metrics are calculated on the basis of how frequently a CA appears in a cluster as well as how strongly the CVs support the CA's correctness in a cluster [3].
- For genomic data sets CVs can be derived from GO evidence codes to point out the most reliable CAs to be used for gene functional prediction purposes [8, 12]. This is in contrast to previous methods, where CVs were calculated at the end to indicate the reliability of a CA's belonging to a cluster [25].

The remainder of this paper is organized as follows. In Section 2 we describe previous work on the k-Modes clustering algorithm. In Section 3 we describe our M-BILCOM clustering algorithm, which is a combination of MULICsoft and BILCOM. In Sections 4 and 5 we describe the results for applying M-BILCOM to highly noisy yeast data sets for which the correct result was known and we show that it was more successful than BILCOM and AutoClass in reproducing the correct cluster structure. In Section 6 we propose and test two significance metrics for the CAs in the

resulting clusters and we demonstrate their utility for gene functional prediction. In Section 7 we discuss implications of our study for biologists and gene functional prediction. Finally, we conclude the paper in Section 8.

2. BACKGROUND ON K-MODES

k-Modes is a clustering algorithm that deals with categorical data only [18,19]. The k-Modes clustering algorithm requires the user to specify from the beginning the number of clusters to be produced and the algorithm builds and refines the specified number of clusters. Each cluster has a mode associated with it. Assuming that the objects in the data set are described by m categorical attributes, the mode of a cluster is a vector $Q=\{q_1, q_2, \ldots, q_m\}$ where q_i is the most frequent value for the ith attribute in the cluster of objects.

Given a set of data and a number k, the k-Modes algorithm clusters the set of data as follows:

1. Select initial k modes for k clusters.
2. For each object X
   a. Calculate the similarity between object X and the modes of all clusters.
   b. Insert object X into the cluster c whose mode is the most similar to object X.
   c. Update the mode of cluster c
3. Retest the similarity of objects against the current modes. If an object is found to be closer to the mode of another cluster rather than its own cluster, reallocate the object to that cluster and update the modes of both clusters.
4. Repeat 3 until no or few objects change clusters after a full cycle test of all the objects.

A similarity metric is needed to choose the closest cluster to an object by computing the similarity between the cluster's mode and the object. Let $X=\{x_1, x_2, \ldots, x_m\}$ be an object, where x_i is the value for the ith attribute, and $Q=\{q_1, q_2, \ldots, q_m\}$ be the mode of a cluster. The similarity between X and Q can be defined as:

$$\text{similarity}(X, Q) = \sum_{i=1}^{m} \delta(x_i, q_i)$$

$$\text{where } \delta(x_i, q_i) = \begin{cases} 1 & (x_i = q_i); \\ 0 & (x_i \neq q_i). \end{cases}$$

Given the similarity measure and k, the clustering result produced by the k-Modes algorithm on a set of data depends on the initial modes and the ordering of the objects presented to k-Modes. In [18] two methods for selecting the initial modes are discussed and compared.

In the descriptions that follow we assume that C represents the total number of clusters and we use c to index the clusters. We assume that I represents the number of attributes in the data set and we use i to index the attributes.

3. THE M-BILCOM CLUSTERING ALGORITHM

M-BILCOM is a combination of MULICsoft [2] and BILCOM [1]. The basic idea of our algorithm is to do clustering at two levels, where the first level clustering imposes an underlying framework for the second level clustering, thus simulating a Bayesian prior as described in [1]. The categorical similarity is emphasized at the first level and the numerical similarity at the second level. *The level one clusters are given as input to level two and the level two clusters are the output of the clustering process.* The process looks as in Figure 1. As shown, both level one and level two involve the same number of clusters, four in this example. The level two clusters consist of subclusters. Data object A was assigned to different level one and level two clusters, because the numerical similarity at the second level was stronger than the categorical similarity at the first level. Thus, in the case of Figure 1 the following relationship holds for object A:

$$categorical\_similarity(A, cluster2) + numerical\_similarity(A, cluster2) >$$

$$categorical\_similarity(A, cluster3) + numerical\_similarity(A, cluster3)$$

On the other hand, data object B was assigned to the same clusters in both levels one and two, because both numerical and categorical similarity supported this classification. Thus, our algorithm considers for a data object both categorical and numerical similarity to the clusters to which it may be allocated.

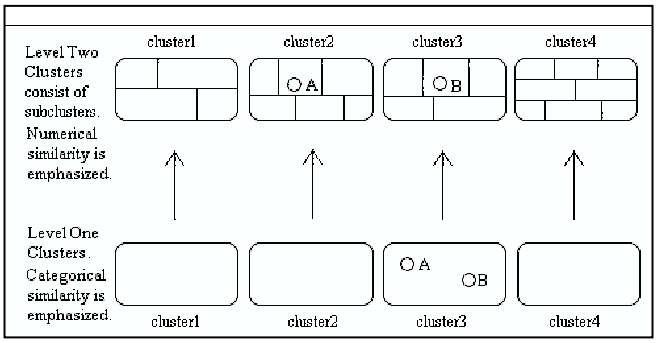

Figure 1. Overview of M-BILCOM clustering.

We emphasize that different types of data are used at levels one and two. At the first level the categorical data used represent something that has been observed to be true about the data set objects before the experiment takes place. For example the data at the first level might look as follows: Class:LIVE; SEX:male; STEROID:yes; FATIGUE:no; ANOREXIA:no . At the second level, on the other hand, the numerical data used represent the results of an experiment involving the data set objects. For example the data at the second level might look as follows: BILIRUBIN:0.39; ALBUMIN:2.1; PROTIME:10.

Our clustering method has the following requirements:

1. The coherence of each cluster should be maximized, considering both the numerical and categorical similarity of the objects.
2. Only the objects with highest categorical similarity to a cluster should form the basis for clustering at the first level.
3. The results of the first level clustering – which is the prior for the process - should not exert an overly strong effect on the second level, so that the second level clustering can escape a poor prior.
4. It should be possible to form a flexible number of clusters.
5. The similarity formula for comparing a mode to an object should increase an object's likelihood to be attached to a cluster as many CAs match the mode and as the CVs of those CAs increase.

We combined MULICsoft [2] and BILCOM [1] into an advanced clustering algorithm named M-BILCOM. This algorithm is similar to BILCOM [1], except that at the first level it also considers weights between 0.0 and 1.0 on the CAs - which we refer to as confidence values or CVs. This combined algorithm is especially useful in cases where the CAs have been perturbed and a CV between 0.0 and 1.0 has been assigned to each CA to indicate the certainty of correctness. Alternatively, M-BILCOM

could be used on biological yeast data sets – such as SGD - where the certainty of correctness that exists on current knowledge is expressed as GO evidence codes [8, 12, 20].

3.1 First Level Clustering: MULICsoft

At the first level, clustering is performed using MULICsoft, an extension of MULIC [2]. MULICsoft and MULIC are extensions of the k-Modes clustering algorithm for categorical data sets [18]. The number of clusters resulting from this level equals the final number of clusters desired, as illustrated by Figures 1 and 2.

The purpose of MULICsoft is to maximize the following similarity formula *at each iteration individually*, while ensuring that all objects may eventually be inserted in clusters:

$$\sum_{n=1}^{N} similarity(object_n, mode_n)$$

where $object_n$ is the nth object in the data set to be clustered and $mode_n$ is the mode of the cluster to which $object_n$ has been classified.

MULICsoft starts by reading all objects from the input file in order. Then, it continues iterating over all objects that have not been placed in clusters yet, to find the closest cluster. In all iterations, the closest cluster is determined for each unclassified object by comparing how many similar annotations exist between a mode and the object. The similarity between a mode and an object is determined using a special variation of the k-Modes similarity metric [2,18] that is described in Section 3.1.1.

The variable *num_CAs_can_differ* is maintained to indicate how strong the similarity has to be between an object and the closest cluster's mode for the object to be inserted in the cluster – initially *num_CAs_can_differ* equals 0, meaning that the similarity has to be very strong between an object and the closest cluster's mode. If the number of different annotations between the object and the closest cluster's mode are greater than *num_CAs_can_differ*, then, the object is inserted in a new cluster on its own. If the number of different annotations between the object and the closest cluster's mode are less than or equal to *num_CAs_can_differ*, then, the object is inserted in the closest cluster and the mode is updated.

At the end of each iteration all clusters with size one are removed, so their objects will be reconsidered at the next iteration. This ensures that the clusters that persist are only those containing at least two objects for which a substantial similarity can be found. Objects belonging to clusters with size greater than one are removed from the linked list of objects, so they are not reclustered.

At the end of each iteration if no objects have been placed in clusters of size greater than 1, then, the threshold *num_CAs_can_differ* is incremented to represent how many annotations are allowed to differ next time. Thus, at the next iteration the threshold will be more flexible, ensuring that more objects will be placed in clusters. Eventually, all objects will be given the opportunity to be placed, even if the closest cluster is a little similar. The iterative process may stop when all objects have been placed in clusters of size greater than 1, or when *num_CAs_can_differ* is greater than a threshold.

The MULICsoft algorithm gives the opportunity for all objects to be placed in clusters eventually, because *num_CAs_can_differ* may continue increasing until all objects are classified. Even if, in the extreme case, an object *o* with *h* annotations has only one or zero annotations similar to the mode of the closest cluster, it can be classified when *num_CAs_can_differ = I-1 or I.*

Figure 2 illustrates what the results of MULICsoft typically look like. Each cluster consists of many different "layers" of objects. The layer of an object represents how strong the object's similarity was to the mode of the cluster when the object was inserted. The cluster's layer in which an object is inserted depends on the value of *num_CAs_can_differ*. Thus, lower layers have a lower coherence and correspond to higher values of *num_CAs_can_differ* and to a more flexible similarity criterion for insertion. MULICsoft starts by inserting as many objects as possible at higher layers and then moves to lower layers, creating them as the need arises. Eventually, all objects may be placed in clusters; if little similarity exists between an object and its closest cluster, the object will be inserted in a lower layer.

If an unclassified object has equal similarity to the modes of the two (or more) closest clusters, then, the algorithm tries to resolve this 'tie' by comparing the object to the modes of the clusters' top layers; these modes were stored by MULICsoft when the clusters were created so they do not need to be recomputed. If the object has equal similarity to all top layers' modes, the object is assigned to the cluster with the highest bottom layer. If all clusters have the same bottom layer then the object is assigned to the first cluster, since there is insufficient data for selecting the best cluster.

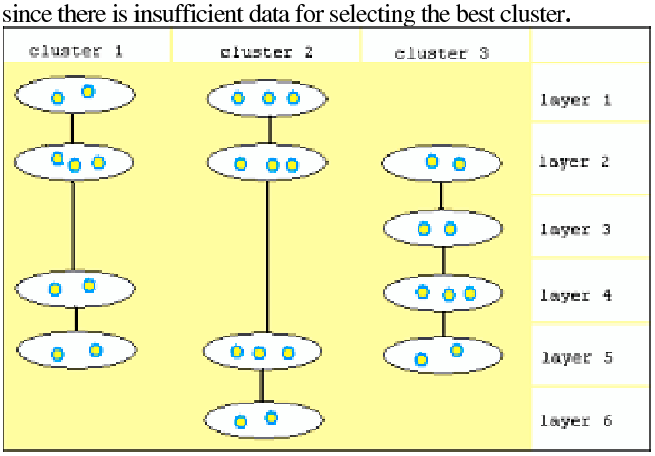

Figure 2. An illustration of typical MULICsoft results. As shown, each cluster might consist of different layers representing different coherences and similarities of the objects attached to the cluster.

The runtime complexity of MULICsoft is *O(IN)*, where *I* represents the number of annotations in an object and *N* represents the number of objects in the data set [2].

The question remains of how to select the data objects to be used at level one. The level one objects are those whose comparison to the mode of the closest cluster yields a result that is greater than or equal to a threshold *minimum_mode_similarity*. The rest of the objects are used at level two (described in Section 3.2). For this purpose, the user can specify a maximum value for *num_CAs_can_differ* - a value of *I-minimum_mode_similarity*, where *I* is the total number of annotations in an object. When *num_CAs_can_differ* passes this value any remaining objects are held to be considered in the second level instead. The reason for which we choose to insert in the first level clusters just the objects whose similarity to the closest mode yields a value higher than a threshold *minimum_mode_similarity* is because the objects that yield a low similarity to the closest mode are more likely to be inserted in the wrong cluster, as we show in [1,2]. Thus, the objects whose classification in clusters based on categorical similarity is not reliable enough, are clustered in the second level instead, where the numerical similarity of objects to second level

clusters is more influential, as described next. We discuss using different maximum values for *num_CAs_can_differ* in Section 4.

3.1.1 The MULICsoft Similarity Metric

All CAs in an object have "weights" or CVs in the range 0.0 to 1.0 associated with them. We represent the weights in an object as f_object_i. The similarity metric used in MULICsoft for computing the similarity between a mode and an object considers both the CAs and their weights. Our similarity metric attempts to amplify the object positions having high weights, at pairs of CAs between an object and a mode that have identical values.

$$f\_similarity = \sum_{i=0}^{I} \frac{6-(4\times f\_objecti)}{5-(4\times f\_objecti)} \times \delta(objecti, \text{mod } ei)$$

$$\text{where } \delta(object_i, \text{mod } e_i) = \begin{cases} 1 & (object_i = \text{mod } e_i = 1); \\ 0 & (object_i \neq \text{mod } e_i). \end{cases}$$

This similarity metric gives more importance to high weights (1.0) than low weights (0.1) at categorical attributes with identical values between the object and the mode.

Figure 3 shows that our similarity formula for comparing a mode to an object increases an object's likelihood to be attached to a cluster as many CAs match the cluster's mode and as the weights on those CAs increase.

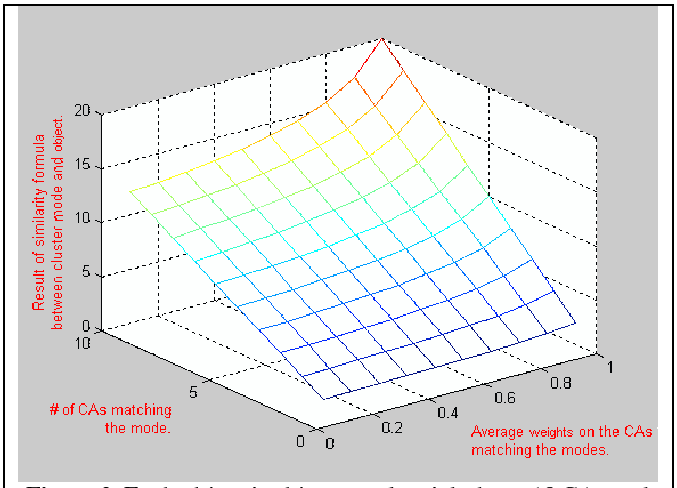

Figure 3. Each object in this example might have 10 CAs and "weights" (or CVs). This graph shows that an object will be much more likely to be assigned to a cluster if all CAs match the mode with high weights of 1.0, than if all CAs match the mode with medium weights of 0.5, than if all CAs match the mode with low weights of 0.1, than if 1 CA matches the mode with a high weight, than if 1 CA matches the mode with a low weight. This graph was produced with the values for the weights as described previously.

3.1.2 Dealing with Outliers

MULIC and MULICsoft can, eventually, put all the objects in clusters. When *num_values_can_differ* equals the number of attributes, an unclassified object can be inserted in the lowest layer *h* of any existing cluster. This is undesirable if the object is an outlier and has little similarity with any cluster. The user can disallow this situation from happening, by specifying a maximum value for *num_values_can_differ*, so, when this value is reached any remaining objects are not classified and are treated as outliers. As discussed in [2] for clustering of software systems, the overall quality of the results often improves by treating the lowest-layer objects as outliers.

3.2 Second Level Clustering: BILCOM

The second level performs clustering of numerical data sets, but bases the process on the first level results that impose an underlying framework using CAs on the objects. The second level uses both the objects clustered at the first level and all the rest of the objects that have NAs. The clusters resulting from the second clustering level are the output of the process [1].

Step 1: One object in each first level cluster is set as a *seed*, while all the rest of the objects in the cluster are set as *centers*. The object to be set as seed is determined by finding an object that is at the top layer of the cluster – ideally in layer 0. The reason why we choose for seed an object in the top layer is that the most influential objects at the second level should be those that are closest to the 'center' of a first level cluster. The MULIC paper [2] shows that objects at the highest layer have a higher average similarity to all other cluster objects than lower level objects do.

If the top layer of a cluster is layer 0, then, we have no difficulty in choosing the seed since all objects have the same CAs. If the top layer of a cluster is not layer 0 and it contains more than one object, then, we choose the seed by comparing all top layer objects to the cluster's mode to find the closest object. If this does not resolve the ambiguity, then, we compare all top layer objects to the cluster's top layer mode – which was stored by MULIC when the cluster was created - to find the closest object. If all top layer objects have the same similarities to modes, then, we assign the seed to be the first top layer object, since there is insufficient information for choosing the best seed.

Step 2: Each *seed* and *center* is allocated to a new *second level subcluster*. The output of this step are *subclusters*, that we refer to as *seed-containing* or *center-containing* subclusters, whose total number equals the number of objects clustered at the first level.

Step 3: Each object that did not participate at the first level is inserted into the second level subcluster containing the most numerically similar *seed* or *center* from the first level. Numerical similarity for Steps 3-5 is determined by the *Pearson correlation coefficient* or the *Shrinkage-based similarity metric* introduced by Cherepinsky et al [7].

Step 4: In this step each center-containing subcluster at the second level is merged with the most numerically similar seed-containing subcluster. The most similar seed-containing subcluster is determined by using our version of the ROCK goodness measure [16] that is evaluated between the center-containing subcluster in question and all seed-containing subclusters:

$$G(Ci, Cj) = \frac{link[Ci, Cj]}{size(Ci) \times size(Cj)}$$

where $link[C_i, C_j]$ stores the number of cross links between clusters C_i and C_j, by evaluating $\Sigma(p_q \in C_i,\ p_r \in C_j)\ link(p_q, p_r)$. $link(p_q, p_r)$ is a boolean value specifying whether a link exists between points p_q and p_r . It is computed by determining if the two points' NAs' similarity is higher than a threshold *minimum_numer_similarity* and if so, setting a link between them. The rationale for using a variation of ROCK's goodness metric for this step is that the link-based approach of ROCK adopts a global approach to the clustering problem, by capturing the global knowledge of neighboring data points between clusters. It has

been shown to be more robust than methods that adopt a local approach to clustering, like hierarchical clustering [16].

Step 5: In this step the loop below is performed that refines the second level clusters, increasing the chances that a center-containing subcluster will merge with the seed-containing subcluster such that the center and the seed were clustered together at the first level.

```
repeat {
for (each center-containing subcluster at level 2)
       if(numerical_similarity_of_center_subclust
er_to_level_1_seed_cluster ×
categorical_similarity_of_center_to_seed_of_level_
1_cluster >
numerical_similarity_of_center_subcluster_to_its_n
umerically_similar_level_2_cluster ×
categorical_similarity_of_center_to_seed_of_its_nu
merically_similar_level_2_cluster)   merge
center_subcluster to seed_subcluster from level 1;
} until (no center_subclusters change);
```

All variables take real values between 0.0 and 1.0. The variables:

```
numerical_similarity_of_center_subcluster_to_level
_1_seed_cluster
numerical_similarity_of_center_subcluster_to_its_n
umerically_similar_level_2_cluster
```

represent the numerical similarity between the center-containing subcluster and the level 1 seed-containing cluster or the most numerically similar level 2 cluster, respectively. This similarity includes the subclusters that were merged to the clusters in previous iterations.

The variables:

```
categorical_similarity_of_center_to_seed_of_level_
1_cluster
categorical_similarity_of_center_to_seed_of_its_nu
merically_similar_level_2_cluster
```

represent the categorical similarity between the center of a subcluster and the seed of its level one cluster, or the seed of its most numerically similar level two cluster, respectively. These are computed as follows:

$$\text{similarity}(\text{center},\text{mode}) = \frac{\sum_{i=1}^{I} \delta(\text{center}_i, \text{mode}_i)}{I}$$

where $\delta(center_i, mode_i) = 1$ *if* $center_i = mode_i$, *0 otherwise.*

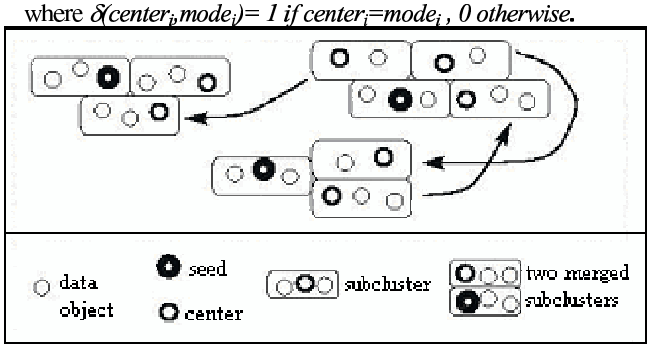

Figure 4. The results of second level clustering steps 4 and 5. A center-containing subcluster becomes attracted to a seed-containing subcluster if the center and seed were clustered together at the first level. The subclusters get merged if the numerical similarity between them permits.

According to this loop, every center-containing subcluster is attracted to the seed-containing subcluster such that the seed and center were clustered together at the first level. The attraction is stronger if there was high categorical similarity of seed and center, but it is lower if there was low categorical similarity. If a center was not categorically similar enough to the seed with which it was clustered together at the first level, then, the subcluster should be likely to remain merged with its current numerically most similar seed-containing subcluster, determined at step 4. The Bayesian rationale behind this loop is discussed in [1]. Figure 4 illustrates the results of steps 4 and 5.

The reason why the inequality comparison in step 5 considers the seeds of clusters, instead of the cluster modes, is that by considering similarity to seeds we are effectively giving the objects a *second chance* to reorganize and to escape their level 1 clustering, if the level 1 clustering was weak. Since the level 1 clustering was based on comparisons to modes that often yield wrong results and, therefore, objects may be attached to wrong clusters, the comparison in step 5 allows the similarities to be reconsidered. The MULIC paper [2] showed that seeds have a higher average similarity to all other cluster objects than lower level objects do. Then, an object can be attached either to the cluster suggested by level 1 or to its numerically similar level 2 subcluster, considering both the categorical similarity to the appropriate seed and the numerical similarity to the appropriate seed's cluster.

We have done tests to show that our algorithm is able to escape a poor prior – for instance because an object was attached to a level 1 cluster with weak similarity to the cluster mode, or because the similarity to the mode was erroneously high enough, or because the object had wrong CAs with low confidence to be correct. The level one categorical similarity may return medium or weak values when the prior is poor. In this case, a center-containing subcluster *a* will tend to stay merged to its most numerically similar second level seed-containing subcluster, instead of the subcluster suggested by the first level results. Thus, the prior can be escaped and the data can be clustered correctly. A center-containing subcluster *a* will not be merged to the seed-containing subcluster *b* - as suggested by the first level results such that seed and center were clustered together at the first level - unless their numerical similarity is very high.

On the other hand, we want to be sure that if a center-containing subcluster *a* gets merged to the subcluster *b* containing the seed with which the center was clustered together at the first level, then, *a* will still be numerically similar enough to *b*. This way we ensure that if we merge a subcluster to the subcluster that is recommended by the results of the first level clustering, the numerical similarity will not lose much of its value either.

4. EXPERIMENTS ON YEAST DATA

We have validated M-BILCOM on mixed numerical and categorical yeast data. We used the yeast data sets shown in Table 1, having mixed categorical and numerical attribute values [7, 9]. However, we perturbed the CAs randomly and assigned statistical confidence values based on the probability that an attribute was perturbed or not. Our tests and results are described below.

We represented CAs on a gene in terms of Gene Ontology (GO) and GOSlim annotations. GO is a dynamically controlled vocabulary that can be applied to many organisms, even as knowledge of gene and protein roles in cells is changing. GO is organized along the categories of *molecular function*, *biological process* and *cellular location* [12]. GOSlim are GO annotations that represent high level knowledge on genes and are also organized along the categories of function, process and location.

Most of the GO and GOSlim annotations on the yeast genes exist in the publicly accessible SGD database, along with GO evidence codes [8, 12, 20]. We created six pools of CAs for each gene and each pool contained GO annotations of a specific type. Three pools contained GO annotations for molecular function, biological process and cellular location of a gene. The other three pools contained GOSlim annotations for each GO annotation.

Many CAs were perturbed and the attribute values in the data set were assigned CVs between 0.1 and 1.0. For this purpose, for each CA we generated a *limit* in a range from 0.1 to 1.0 and, then, generated a random number ρ from 0.0 to 1.0. If ρ exceeded the *limit*, then we perturbed the CA by assigning it a value taken randomly from the set of possible values for that annotation. The CV for the CA was set equal to the *limit* regardless of whether it was actually perturbed or not. This simulates the uncertainty that exists on current knowledge and that is expressed in SGD as GO evidence codes [8, 12, 20]. All attribute values on all data objects were assigned a CV between 0.1 and 1.0 and objects whose CAs had lower CVs were more likely to have been perturbed than objects whose attribute values had higher CVs.

The yeast microorganism performs a constant cell-cycle. The yeast cell-cycle gene expression program is regulated by each of the nine known cell-cycle transcriptional activators; these control the flow from one stage of the cell-cycle to the next [7]. It was also found that cell-cycle transcriptional activators that function during one stage of the cell-cycle regulate transcriptional activators that function during the next stage. This serial regulation of transcriptional activators together with various functional properties suggests a simple way of partitioning some selected cell-cycle genes into nine clusters, each one characterized by a group of transcriptional activators working together and their functions [7]. Table 1 shows our hypothesis about how the genes should be correctly grouped by transcriptional activators and cell-cycle functions. For instance, group 2 is characterized by the activators Swi6 and Mbp1 and the function involving DNA replication and repair at the juncture of G1 and S stages.

Table 1. Genes in the data set of Cherepinsky et al. [7] grouped by functions. This is our hypothesis about the correct clustering results.

Group	Activators	Genes	Functions
1	Swi4, Swi6	Cln1, Cln2, Gic1, Msb2, Rsr1, Bud9, Mnn1, Och1, Exg1, Kre6, Cwp1	Budding
2	Swi6, Mbp1	Clb5, Clb6, Rnr1, Rad27, Cdc21, Dun1, Rad51, Cdc45, Mcm2	DNA replication and repair
3	Swi4, Swi6	Htb1, Htb2, Hta1, Hta2, Hta3, Hho1	Chromatin
4	Fkh1	Hhf1, Hht1, Tel2, Arp7	Chromatin
5	Fkh1	Tem1	Mitosis control
6	Ndd1, Fkh2, Mcm1	Clb2, Ace2, Swi5, Cdc20	Mitosis control
7	Ace2, Swi5	Cts1, Egt2	Cytokinesis
8	Mcm1	Mcm3, Mcm6, Cdc6, Cdc46	Prereplication complex formation
9	Mcm1	Ste2, Far1	Mating

Cherepinsky et al. [7] defined a notation to represent the resulting cluster sets and a scoring function to aid in their comparison. Each cluster set is written as:

$$\overline{\{x \rightarrow \{\{y_1, z_1\}, \{y_2, z_2\}, \ldots, \{y_{n_x}, z_{n_x}\}\}\}}_{x=1}^{\#\text{ of groups}},$$

where x denotes the group number as described in Table 1, n_x is the number of clusters group x appears in, and for each cluster $j \in \{1,\ldots, n_x\}$ there are y_j genes from group x and z_j genes from other groups in Tables 10 and 11. A value of * for z_j denotes that cluster j contains additional genes, although none of them are cell-cycle genes. The cluster set can then be scored as follows:

$$\mathrm{FP}(\gamma) = \frac{1}{2}\sum_{x}\sum_{j=1}^{n_x} y_j \cdot z_j$$

$$\mathrm{FN}(\gamma) = \sum_{x}\sum_{1 \le j < k \le n_x} y_j \cdot y_k$$

$$\text{Error score}(\gamma) = \mathrm{FP}(\gamma) + \mathrm{FN}(\gamma).$$

We have compared the error rates of M-BILCOM to those of AutoClass [24] and BILCOM [1] applied to the "perturbed" yeast data set. The error rates are shown in the table below. The M-BILCOM error rate is lower than BILCOM and AutoClass.

Clustering Tool	Cherepinsky Error rate
M-BILCOM	122
BILCOM	133
AutoClass	297

We have applied AutoClass [24] to the mixed categorical and numerical yeast data set.

Table 2. Clustering results of AutoClass.

Cluster	Genes
1	CLN1, CLN2, GIC1, GIC2, MSB2, RSR1, BUD9, MNN1, OCH1, EXG1, KRE6, CWP1, CLB5, CLB6, RAD51, CDC45, HTB1, HTA2, HHO1, TEL2
2	ARP7, TEM1, CLB2, ACE2, SWI5, CDC20, CTS1, EGT2, MCM3, MCM6, CDC6, CDC46, STE2
3	RNR1, RAD27, CDC21, DUN1, MCM2, HTB2, HTA1, HHF1, HHT1, FAR1

Given the hypothesis in Table 1 and the set of AutoClass results shown in Table 2, the resulting clusters with the error score can be written as follows, according to the error rate introduced by Cherepinsky et al. [7]:

{1->{{11,9}}, *2->{{4,16},{5,5}},* *3->{{3,17},{2,8}},* *4->{{1,19},{1,12},{2,8}},* *5->{{1,12}},* *6->{{4,9}},* *7->{{2,11}},* *8->{{4,9}},* *9->{{1,12},{1,9}} }.*	*FP = 265* *FN = 32* *Error = 297*

We have also applied BILCOM [1] to the mixed categorical and numerical yeast data set.

Table 3. Clustering results of BILCOM using as numerical similarity metric between objects the Pearson Correlation Coefficient [7] and a max value for *num_CAs_can_differ* of 7.

Cluster	Genes
1	RSR1, HHT1, ARP7, BUD9, CTS1
2	KRE6, CWP1
3	RNR1, CDC45, MCM3, CDC46, MCM2
4	EXG1, EGT2

5	MCM6, CDC6
6	HHF1, HTB2, HTA2
7	HTB1, HTA1, HHO1
8	GIC1, TEL2, GIC2, MSB2
9	FAR1, STE2, ACE2, SWI5, TEM1
10	RAD27, CDC21, DUN1
11	CLN2, RAD51, MNN1, CLN1, CLB6, OCH1, CLB5, CLB2, CDC20

Given the hypothesis in Table 1 and the set of BILCOM results shown in Table 3, the resulting clusters with the error score can be written as follows, according to the error rate introduced by Cherepinsky et al. [7]:

{1->{{4,4},{1,3},{1,1},{2,1},{2,0}}, *2->{{3,0},{3,2},{3,5}},* *3->{{3,0},{2,1}},* *4->{{2,3},{1,2},{1,4}},* *5->{{1,4}},* *6->{{2,3},{2,7}},* *7->{{1,4},{1,1}},* *8->{{2,0},{2,3}},* *9->{{2,3}} }.*	*FP = 49* *FN = 47+13+24* *Error = 133*

We have also applied M-BILCOM to the mixed categorical and numerical yeast data set.

Table 4. Clustering results of M-BILCOM using as numerical similarity metric between 2 objects the Pearson Correlation Coefficient [7] and a max value for *num_CAs_can_differ* of 7.

Cluster	Genes
1	RSR1, BUD9, CTS1
2	KRE6, ARP7, HHT1, CWP1
3	RNR1, CDC45, MCM3, STE2, CDC46, MCM2
4	EXG1, EGT2
5	MCM6, TEM1, CDC6
6	HHF1, HTB2, HTA2
7	HTB1, HHO1, HTA1
8	GIC1, TEL2, GIC2, MSB2
9	FAR1, ACE2, CDC20, SWI5
10	RAD27, CDC21, MNN1, DUN1, RAD51
11	CLN2, CLN1, CLB6, CLB5, OCH1, CLB2

Given the hypothesis in Table 1 and the set of M-BILCOM results shown in Table 4, the resulting clusters with the error score can be written as follows, according to the error rate introduced by Cherepinsky et al. [7]:

{1->{{3,3},{2,2},{2,1},{1,1},{2,2},{1,4}}, *2->{{4,1},{3,3}},* *3->{{3,0},{3,0}},* *4->{{2,2},{1,2},{1,3}},* *5->{{1,2}},* *6->{{3,1},{1,5}},* *7->{{1,2},{1,1}},* *8->{{2,4},{2,1}},* *9->{{1,5},{1,3}} }.*	*FP = 38* *FN = 35+49* *Error = 122*

5. EXPERIMENTS ON SIMULATED DATA

We generated artificial data sets that simulate the results by Spellman et al [23] who showed that in each cluster there is a consistent pattern of NAs that appear frequently and that different CAs are characteristic of different clusters. We used numerical data derived from gene expression studies on the yeast *Saccharomyces cerevisiae*. These data sets were produced at Stanford to study the yeast cell cycle across time and under various experimental conditions and are available from the SGD database [9, 23]. The data set contained 6,200 genes. The purpose of our simulation was to assign annotations to each gene based on the numerical gene expression data [9], in such a manner that the assignment of annotations simulates knowledge about the role of genes in the yeast cell cycle.

We assigned 6 CAs on each gene based on the numerical annotations (NAs), representing the genes' action during cell cycle. The assignment of the CAs followed a pattern that simulates existing knowledge on the role of genes in the yeast cell cycle. The assigned CAs split the objects into a number of well-defined groups, which we attempt to retrieve using clustering; thus, a different set of attribute values had to be used for each group. This simulates the results by Spellman et al, who showed that in each cluster there is a consistent pattern of NAs that appear frequently and that different CAs are characteristic of different clusters [23]. We assigned the 6 CAs using the following strategy: (1) The first annotation split the genes into cell cycle phases and has the values G1, S, G2, M, M/G1, or unknown. This annotation was set for each gene based on the experimental point at which the gene reaches its peak expression level, indicating what cell cycle phase it is likely to be involved in. This simulates the cell cycle phases as they are derived by Spellman et al and the genes that are likely to be involved in each phase. (2)(3) The second and third annotations were set for each gene based on whether the gene's expression level peaks at the phase of the cell cycle at which it is active. The second annotation, which can take 6 values (A,B,C,D,E,unknown), simulates an overall process like *DNA replication* or *transport of essential minerals and organic compounds across the cell membrane*. The third annotation, which can take 11 values (F,G,H,I,J,K,L,M,N,O,unknown), simulates a more specific function like *DNA polymerases* or *nucleotide synthesis* or *initiation of DNA synthesis*. The purpose of the second and third annotations was to further subdivide the groups created by the first annotation into subgroups. The first three annotations are influential enough to classify each gene in a cluster. Thus, the clusters that we will try to retrieve are based primarily on the first three annotations. (4) The fourth annotation was set for each gene based on whether the gene is observed to reach its peak expression level right before the G1 stage. This annotation simulates the *start of mitosis* and can take 3 values (high,low,unknown). (5) The fifth annotation was set for each gene based on whether the gene is observed to reach its peak expression level at the end of the cell cycle. This annotation simulates the *exit from mitosis* and can take 3 values (high,low,unknown). (6) Finally, the sixth annotation was set for each gene based on whether the gene is observed to reach its peak expression level right before the S stage. This annotation can take 3 values (high,low,unknown). The fourth to sixth annotations on their own may or may not classify each gene into a clear group.

Furthermore, we perturbed the CAs to simulate noise in the resulting data set. Our aim was to use M-BILCOM to retrieve the known underlying cluster structure effectively. A significant outcome of our experiments was to show that given the genes whose CAs were not perturbed in the simulation (most of which are likely to have high CVs) a fair number of genes were assigned to the correct clusters to which they were categorically similar and were not assigned to the incorrect clusters to which they may be numerically similar. The basis for this is that most of these genes had a high confidence overall. Another significant outcome of our experiments was to show that given the genes whose CAs were perturbed in the simulation (most of which are likely to have low

CVs) a fair number of genes were assigned to the correct clusters to which they were likely to be numerically similar and were not assigned to the incorrect clusters to which they were categorically similar. The basis for this is that most of these genes had a low confidence overall.

Many CAs were perturbed and the attribute values in the data set were assigned CVs between 0.1 and 1.0. For this purpose, for each CA we generated a *limit* in a range from *0.4* to 1.0 and, then, generated a random number ρ from 0.0 to 1.0. If ρ exceeded the limit, then we perturbed the CA by assigning it a value taken randomly from the set of possible values for that attribute. The CV for the CA was set equal to the *limit,* regardless of whether it was actually perturbed or not. This simulates the uncertainty that exists on current knowledge and that is expressed in SGD as GO evidence codes [8, 12, 20]. In the produced data set 2,024 data objects had their original attribute values modified out of 6100. All CAs on all data objects were assigned a CV between 0.4 and 1.0 and objects with lower CVs were more likely to have been modified than objects with higher CVs.

We clustered the simulated data set into 20 clusters. This number of clusters was derived from the number of combinations of values that the first three CAs of each object can take in our simulated data and because this number of clusters allowed the algorithm to converge in a reasonable amount of time. Table 5 shows the statistics for all 20 clusters, though we ignore the results for clusters whose size was too small.

Table 5. Results for clustering the data set into 20 clusters. We do not show results for clusters whose size was too small.
1 – Cluster #
2 - Number of objects in the cluster
3 - Most common values {A,B,C} on the objects' first 3 CAs
4 - Ratio X of objects in the cluster that had CAs modified during the simulation
5 - Ratio of X that had an original CA very close to {A,B,C}
6 - Ratio P of objects in the cluster that had an original CA very close to {A,B,C}
7 - Ratio of P that had its CAs modified during the simulation
8 - Number of merged second level subclusters

1	2	3	4	5	6	7	8
1	2032	{M,D, L}	616/ 2032	217/ 616	1047/ 2032	217/ 1047	537
2	1186	{MG1, E,N}	305/ 1186	202/ 305	1102/ 1186	202/ 1102	180
3	724	{G2,C, J}	177/ 724	111/ 177	672/ 724	111/ 672	121
4	317	{G1,A, F}	83/ 317	48/ 83	302/ 317	48/ 302	44
5	709	{S,B,H }	94/ 709	24/ 94	684/ 709	24/ 684	183
6	218	{M,D, M}	50/ 218	26/ 50	198/ 218	26/ 198	59
8	66	{MG1, E,N}	66/66	22/ 66	22/ 66	22/22	9
11	71	{MG1, E,N}	71/71	27/ 71	27/ 71	27/27	3
15	74	{MG1, E,N}	74/74	24/ 74	24/ 74	24/24	2
20	333	{S,B,H } and {S,B,I}	180/ 333	148/ 180	260/ 333	148/ 260	13

What is most noteworthy in this table are clusters 8, 11, 15, because all of their data objects had their annotations modified during our simulation (see column 4). As it can be seen, a vast majority of the data objects in these clusters had original annotations on their first 3 CAs consistent with the most representative annotations {A,B,C} for the cluster (see columns 5,6). Furthermore, all of the data objects whose first 3 CAs were equal to the most representative annotations {A,B,C} for the cluster, were objects whose annotations had been modified during the simulation to different values (see column 7).

Another interesting result is cluster 2, in which the most prominent genes are those with the CA sequence {MG1, E, N} in their first 3 CAs. 202/1102 objects had their CAs modified to a totally different annotation, but were nevertheless assigned to the correct cluster because they had low CVs (see column 7). This shows that our algorithm can overcome a poor prior that is likely to be incorrect and can still produce correct results by using numerical clustering instead. In this cluster, from all objects assigned to it that had their CAs modified (305/1186, as shown in column 4) 202/305 truly had an annotation of {MG1, E, N} or {MG1, E, O} (see column 5). The total cluster size was 1186 and consisted of 180 merged second level subclusters (see column 8). Four of the merged clusters contained a vast majority of objects with modified annotations, and all of these clusters had a substantial portion - or a majority - of objects with an original annotation of {MG1, E, N}.

6. TWO METHODS FOR ANALYZING THE SIGNIFICANCE OF A CATEGORICAL ATTRIBUTE VALUE IN A CLUSTER

In this section, we first describe the real yeast data on which we applied the M-BILCOM clustering algorithm. Then, we describe two significance metrics (SMs) for determining the significance of a CA in a cluster and for supporting gene functional prediction.

6.1 Description of Real Yeast Data Sets

Our algorithm is designed with the goal of applying it to numerical data sets for which some CAs exist and the confidence that the CAs are correct may vary. We used numerical data derived from gene expression studies on the yeast *Saccharomyces cerevisiae*. These data sets were produced at Stanford to study the yeast cell cycle across time and under various experimental conditions and are available from the SGD database [9, 23]. When clustering this data set, we consider each gene to be a 'data object'. The data set contained 6,200 objects.

We represented *CAs* on a gene in terms of Gene Ontology (GO) and GOSlim annotations. GO is a dynamically controlled vocabulary that can be applied to many organisms, even as knowledge of gene and protein roles in cells is changing. GO is organized along the categories of molecular function, biological process and cellular location [12]. GOSlim are GO annotations that represent high level knowledge on genes and are also organized along the categories of function, process and location. Most of the GO and GOSlim annotations on the yeast genes exist in the publicly accessible SGD database, along with GO evidence codes [8, 12, 20]. We created six pools of CAs for each gene and each pool contained GO annotations of a specific type. Three pools contained GO annotations for molecular function, biological process and cellular location of a gene. The other three pools contained GOSlim annotations for each GO annotation.

We attached CVs to the CAs to represent our confidence that the corresponding CA is correct. CVs are real numbers between 0.0

and 1.0, assigned to the CAs of a gene. Besides indicating the confidence that a CA is correct, the CVs on a gene also specify how strongly the gene's CAs should influence the clustering process. The CVs are also used in the significance metrics that we define later. We determined the CVs by using GO evidence codes. GO evidence codes symbolize the evidence that exists for any particular GO or GOSlim annotation [12].

GO evidence codes can be thought of in a loose hierarchy from strong evidence to weak evidence. For example, 'TAS' means 'Traceable Author Statement', while 'NAS' means 'Non-traceable Author Statement' [9]. We assigned a numerical CV to each of the GO evidence codes based on its location in the hierarchy, as shown in Figure 5. NR and ND are set to 0.0, because they are used for annotations to 'unknown', so the CAs should not have an effect on the clustering process. In certain DBs (Swiss-prot-human) only 3 of these evidence codes are commonly used and the most commonly used one is TAS, which is at the top of the hierarchy, meaning that strong evidence exists [20]. We combined the CAs, CVs and numerical gene expression data using a Perl script.

TAS - 1.0
IDA - 1.0
IMP - 0.8
IGI - 0.8
IPI - 0.8
ISS - 0.5
IEP - 0.5
NAS - 0.2
IEA - 0.1
ND - 0.0
NR - 0.0

Figure 5. GO Evidence Codes are mapped to CVs.

A CV primarily depends on whether the CA refers to something that has been *observed* to be true, as opposed to something that is just *believed* to be true. For example, a CA of a "cancerous tissue" refers to an observed phenomenon with a high CV, while a CA of a "non-cancerous tissue" refers to something that is just believed to be true, as the tissue might turn out later to be cancerous.

6.2 Assigning to a Cluster's CAs M-values that take P1-values and CVs into Consideration

Given a resulting cluster, we assigned a P1-value to each CA in the cluster; the term 'P1-value' was derived from the statistical 'P-value'. A P1-value measures whether a cluster contains a CA of a particular type more frequently than would be expected by chance [25]. A P1-value close to 0.0 indicates a frequent occurrence of the CA in the cluster, while a P1-value close to 1.0 its seldom occurrence. We multiplied the resulting P1-value with the reciprocal of the average of all CVs assigned to the CA in the cluster, 1/avg(CV), thus resulting in what we call an *M-value*. M-values allow us to take into consideration the probability that a particular CA occurs in the cluster more frequently than expected by chance, in addition to our confidence that the CA is correct in the cluster. For CAs that occur only once or twice in a cluster, a high P1-value results with an avg(CV) trivial to estimate.

6.3 Analyzing the Significance of a Second Level Subcluster's Classification in a Cluster

This significance metric was inspired by the loop of step 5 that refines the subclusters composing a larger second level cluster (see Section 3.2). Specifically, each subcluster was assigned a significance number by evaluating a formula that considers both categorical (*CAsimilarity)* and numerical (*NAsimilarity)* similarity of the subcluster to the larger second level cluster:

*(weight1*CAsimilarity) +(weight2*NAsimilarity)*

The CAsimilarity for a subcluster is computed by evaluating a categorical variation of ROCK's goodness measure [16] between the subcluster and its larger cluster and multiplying the result by the percentage of genes in the subcluster that were assigned to it on the basis of categorical similarity (see Step 3). The NAsimilarity for a subcluster is computed similarly, by evaluating a numerical variation of ROCK's goodness measure [16] between the subcluster and its larger cluster and multiplying the result by the percentage of genes in the subcluster that were assigned to it on the basis of numerical similarity (see Step 3). We set *weight2* in our trials to be higher than *weight1*, to ensure proper consideration of the numerical similarity of a subcluster.

The subclusters in an overall second level cluster for which the above metric yields the highest values are used for functional prediction by identifying and extracting the most significant genes' CAs in the cluster with highest avg(CV)s.

When a subcluster is placed in a larger second level cluster on the basis of high CA similarity (0.5-1.0) - whether it was assigned there at the beginning of the clustering process or joined it later - this is a factor that increases the significance. The NAsimilarity on the subcluster might be:

- high (0.5-1.0) in which case the significance of its membership is increased, because both CAs and NAs support the gene's classification in the cluster.
- low (0.1-0.4) in which case the significance of its membership is decreased, because CAs support the gene's classification in the cluster but NAs do not.

When a subcluster is placed in a larger second level cluster on the basis of low CA similarity (0.0-0.4) - whether it was assigned there at the beginning of the clustering process or joined it later - this is a factor that decreases the significance. The NAsimilarity on the subcluster is:

- always high (0.7-1.0), however since the CA similarity is low the significance of its membership is decreased because NAs support the gene's classification in the cluster but CAs do not.

6.4 Functional Prediction for Uncharacterized Genes

Both of the above significance metrics (SM) were used for functional prediction of genes.

The first SM was used for functional prediction by taking for each cluster the CAs with the lowest M-values for molecular function, biological process, cellular location and for the GOSlim terms. Then, we tried to apply these CAs to genes in the cluster having CAs labeled as 'Unknown'. Therefore, the CAs with the lowest M-values in a cluster were used to predict the cellular role of genes.

The second SM was used for functional prediction by identifying the subcluster with the highest significance in a larger second level cluster and identifying its genes' CAs with highest avg(CV)s in the cluster, as these were the most significant ones. Then, we tried to apply the extracted CAs to other genes in the cluster having CAs labeled as 'Unknown'. Therefore, the CAs belonging to the subcluster that had the highest significance were used to predict the cellular roles of genes.

6.5 Validation of Predictions Using the First Significance Metric

Our strategy for validating the accuracy of the functional predictions is to reclassify certain genes' CAs as 'Unknown' before the clustering process and attempt to predict the correct genes' cellular roles using the cluster CAs pointed out by our metric. The CAs to be set to 'Unknown' were chosen to have a high average(CV) over all their occurrences in the cluster, because

these are primarily the ones that we would like to be able to predict correctly. The process described next helps us to determine how likely genes are to be assigned their correct CAs.

We iterated over the genes in the cluster with CAs labeled as 'Unknown'. To assess the effectiveness of our technique, we verified that the original CAs of these genes correlated better to the cluster CAs with low M-values - that are pointed out by our SM - than those with high M-values. This correlation signified the likelihood that the genes' CAs labeled as 'Unknown' would be assigned their original annotations, by using CAs with low M-values that are pointed out by our SM. A relatively large number of genes' CAs labeled as 'Unknown' should be likely to be re-assigned their original annotations using CAs with low M-values in the cluster, because a low M-value indicates that a CA occurs frequently amongst the cluster's genes and that a CA is likely to be correct.

We initially clustered the yeast data into 5 clusters. Table 6 gives a description of some CAs that were pointed out in all 5 clusters by our SM, after the CAs with the highest average(CV) in each cluster were set to 'Unknown' and the set was clustered.

Table 6. CAs pointed out in 5 clusters as the most significant. The CAs pointed out in clusters 1-5 as having the lowest M-values - the most representative ones for the cluster - correlated with the CAs in the original cluster that were set to 'Unknown'.

Cluster	Some of the CAs pointed out in each cluster as having low M-values (meaning they occurred frequently and had high avg(CV)) after the CAs with the highest avg(CV) in each cluster were set to 'Unknown' and the set was clustered.
1	vacuolar membrane, ubiquitin-specific protease, small nuclear ribonucleoprotein complex, glycolysis, 3'-5' exoribonuclease, cytosolic small ribosomal subunit, lipid particle, cytosolic large ribosomal subunit, tricarboxylic acid cycle
2	rRNA modification, ATP dependent RNA helicase, nuclear pore, structural molecule, small nucleolar ribonucleoprotein complex, snoRNA binding, mediator complex
3	cytosol, proteasome endopeptidase, non-selective vesicle fusion, translation initiation factor
4	transcription initiation from Pol II promoter, general RNA polymerase II transcription factor, nucleus
5	endoplasmic reticulum membrane, component:endoplasmic reticulum

We have also performed these tests on the yeast data producing 35 and 71 clusters. We provide a concrete example of the utility of our technique for 35 clusters, by focusing on the second cluster having 224 genes. In the original clustering, the following CAs were pointed out as having the lowest M-values: function:transcription regulator, component:nucleus, process:transport, process:cell growth and/or maintenance, process:metabolism, function:transporter, nucleus(a specific, granular annotation). We focus on the 2 most significant (representative) CAs for the cluster:

1) annotation component:nucleus occurred in 160 genes in this cluster and had an average(CV) of 1.0 across all genes. Some genes containing this annotation were YOR064C, YBR247C, YDR205W, YDR206W, YFR023W, YKL117W, YPR196W, YOR141C, YOL116W, YOR294W, YDR076W, YFR037C, YNL148C, YDR510W, YLR074C, YPL049C, YDL064W, YML109W, YNL016W.
2) annotation nucleus(a specific, granular annotation) occurred in 82 genes in this cluster and had an average(CV) of 0.904878 across all genes. Some genes containing this annotation were YBR247C, YDR205W, YDR206W, YFR023W, YKL117W, YPR196W, YOL116W, YOR294W, YDR076W, YFR037C, YNL148C, YLR074C.

In different trials we set all CAs of many genes in which these two annotations occurred originally to 'Unknown' and then re-clustered the data set into 35 clusters. Using the results, we were able to predict correctly that these genes should be annotated as either component:nucleus or nucleus(a granular annotation) by extracting the CAs with lowest M-values. This means that our SM predicted these genes to have their original correct CAs, after setting them to 'Unknown', re-clustering the data set and extracting the CAs with lowest M-values. Figure 6 illustrates the results obtained.

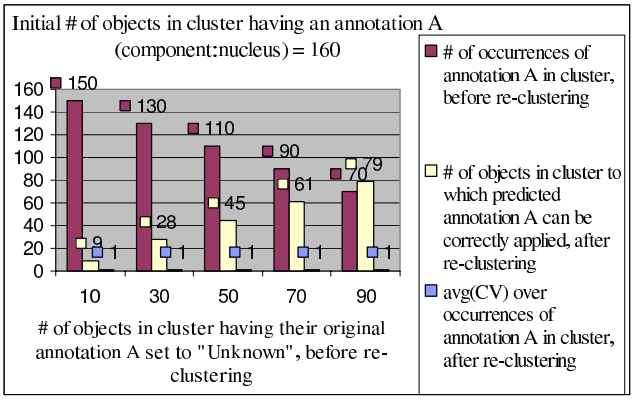

Figure 6. As an increasing number of CAs in the cluster were set to 'Unknown', the same annotation qualified as one of the most significant annotations in the cluster and also remained applicable to a relatively large number of objects (i.e. genes).

6.6 Validation of Predictions Using the Second Significance Metric

Our strategy for validating the accuracy of the functional predictions was to reclassify the CAs of certain genes as 'Unknown' before the clustering process and attempt to predict the correct genes' cellular roles using the cluster CAs pointed out by our metric. The CAs set to 'Unknown' were primarily ones with a high average(CV) over all their occurrences in the cluster, because these were primarily the ones that we would like to be able to predict correctly. The process described next helped us to determine how likely genes were to be assigned their correct CAs.

We iterated over the CAs in the cluster that were labeled as 'Unknown'. To assess the effectiveness of our technique, we verified that the original CAs of these genes correlated better to the cluster annotations pointed out as having the highest significance. CAs pointed out as highly significant were ones occurring frequently across the cluster's genes with high avg(CV). This correlation signified the likelihood that the genes' CAs labeled as 'Unknown' would be re-assigned their original annotations, by using CAs that were pointed out by our SM. A reasonable number of genes' CAs should be likely to be assigned their original annotations using the CAs pointed out by our SM.

We initially clustered the yeast data into 35 clusters, each of which contained a number of smaller subclusters. The second level subclusters pointed out by our SM as significant enough were those containing genes:

1) YHR053C (SM>1 ; 80% of genes not having CV high enough ; 23 genes total),
2) YDL179W (SM>1 ; 96% of genes not having CV high enough ; 104 genes total),
3) YKL182W (SM>0.58 ; 94% of genes not having CV high enough ; 210 genes total),
4) YKR075C (SM>0.44 ; 96% of genes not having CV high enough ; 27 genes total),
5) YLR342W (SM>0.10 ; 91% of genes not having CV high enough ; 22 genes total),
6) YMR246W (SM>0.88 ; 75% of genes not having CV high enough ; 61 genes total),
7) YJL079C (SM>0.06 ; 75% of genes not having CV high enough ; 4 genes total),
8) YCR005C (SM>0.58 ; 63% of genes not having CV high enough ; 470 genes total),
9) YMR186W (SM>0.06 ; 50% of genes not having CV high enough ; 8 genes total),
10) YBR029C (SM>1 ; 0% of genes not having CV high enough ; 1 gene total),

The reason other subclusters yielded low significance was because a majority of their genes had high average(CV) over their CAs, so most genes were assigned on the basis of categorical similarity rather than on the basis of numerical similarity. Thus the dominant factor in the significance metric was low and the overall result was low.

We next needed to identify the CAs in these clusters with the highest average(CV) throughout the entire cluster. We identified the following CAs, for each of the subclusters listed above:

1) copper binding, avg(CV) 0.5 ; cytosol, avg(CV) 1.0
2) cell cycle, avg(CV) 0.5
3) fatty-acid synthase complex, avg(CV) 1.0 ; fatty acid biosynthesis, avg(CV) 1.0 ; vacuole (sensu Fungi), avg(CV) 0.8 ; vacuole inheritance, avg(CV) 0.8 ; thiol-disulfide exchange intermediate, avg(CV) 0.5 ; plasma membrane, avg(CV) 1.0 ; tricarboxylic acid cycle, avg(CV) 1.0
4) cytoplasm, avg(CV) 1.0
5) 1,3-beta-glucan synthase, avg(CV) 0.55
6) long-chain-fatty-acid-CoA-ligase, avg(CV) 0.55 ; lipid metabolism, avg(CV) 0.75 ; lipid particle, avg(CV) 1.0
7) nuclear membrane, avg(CV) 1.0
8) glyoxylate cycle, avg(CV) 1.0 ; peroxisomal matrix, avg(CV) 0.95 ; folic acid and derivative biosynthesis, avg(CV) 0.95 ; pantothenate biosynthesis, avg(CV) 0.8 ; allantoin catabolism, avg(CV) 0.8 ; purine nucleotide biosynthesis, avg(CV) 0.95 ; helicase, avg(CV) 0.5 ; spore wall assembly, avg(CV) 0.8 ; RAB-protein geranylgeranyltransferase, avg(CV) 0.55 ; protein amino acid geranylgeranylation, avg(CV) 1.0 ; RAB-protein geranylgeranyltransferase complex, avg(CV) 1.0
9) response to stress, avg(CV) 0.75
10) phosphatidate cytidylyltransferase, avg(CV) 1.0 ; phosphatidylserine metabolism, avg(CV) 1.0 ; mitochondrion, avg(CV) 1.0

In different trials we set the CAs of many genes in which these annotations occurred originally to 'Unknown' in each of these clusters and re-clustered the entire data set. The same annotations were still pointed out by our significance metric as highly significant in the corresponding clusters. This encouraged us to re-assign the original annotations to the genes whose CAs were set to 'Unknown', which we interpret as a success of our approach.

6.7 Mapping the GO Evidence Codes to CVs

The GO Evidence Codes (GOECs) form a loose hierarchy from strong evidence to weak evidence. The top GOECs in the hierarchy represented by 'TAS' and 'IDA' are mapped to a CV of 1.0 while the bottom GOECs 'ND' and 'NR' to 0.0. We want to show that all other GOECs falling between these extremes in the hierarchy are assigned a real CV that will allow the objects to be partitioned the best in the clustering process. For this purpose, we used the simulated yeast data set from Section 5.

We did a trial using GOECs from the top 3 hierarchy scales: TAS/IDI, IMP/IGI/IPI, ISS/IEP. We set a randomly chosen 1/3 of the CVs in the data set to TAS/IDI, 1/3 to IMP/IGI/IPI and 1/3 to ISS/IEP. Then we set all CVs of objects falling in 5 classes A-E in the data set to the middle GOEC of IMP/IGI/IPI. By mapping the set of GOECs to corresponding CVs of 1.0, 0.8, 0.5, the objects belonging in classes A-E were more clearly partitioned from other objects than when mapping the GOECs to CVs of 1.0, 0.9, 0.6.

Then we repeated this trial by using GOECs from the bottom 3 hierarchy scales: ISS/IEP, NAS, IEA. By mapping the set of GOECs to corresponding CVs of 0.5, 0.2, 0.1, the objects belonging in classes A-E were more clearly partitioned from other objects than when mapping the GOECs to CVs of 0.5, 0.3, 0.2.

7. DISCUSSION ON UTILIZING THE TWO SIGNIFICANCE METRICS FOR DERIVING POTENTIAL GENE FUNCTIONS FOR EXPERIMENTAL VALIDATION

Biologists will find this method useful in wet lab work, for deriving hints about the potential functions of genes and proteins. The hints that are derived as to a gene's function can later be validated experimentally. This will save time and money from the biological experimentalists' side. In our experiments with the yeast cell cycle data set, the utility of the significance metrics is especially evident from the fact that the vast majority of genes in each cluster or subcluster analyzed had all CAs set to 'Unknown' meaning that no knowledge exists. For example, when analyzing the subcluster containing YHR053C using the second significance metric, only 6 out of 20 genes had some kind of CA, while the other 14 genes had CAs set to 'Unknown'. Our significance metric could point out the most representative CAs that are likely to be applicable to the other 14 genes and these functional hints can be tested experimentally.

The *first* significance metric is useful for identifying the most representative CAs in clusters with a plethora of CAs that have *high CVs*. The first metric allows one to identify the CAs in this pool that appear frequently (with a low P1-value) so as to try to apply them to other genes. In our experiments with the yeast cell cycle data set we realised that although 1185 second level subclusters had been produced in total, most of those (1175 = 1185-10 subclusters) had a majority of genes with an average CV over their CAs that was considered high. These genes were assigned to the clusters on the basis of categorical rather than numerical similarity, according to step 3. The first significance metric could be utilized on these 1175 subclusters; or it could be utilized on the overall clusters produced as an end result by our algorithm (the total number of clusters was 5, 35 and 71).

The *second* significance metric applies primarily for identifying the most representative CAs in clusters with a plethora of CAs that have *low CVs*. The second metric allows us to identify the

few CAs that have high CVs in these clusters, so as to try to apply them to other genes. In our experiments, only 10 second level subclusters out of 1185 in total, had a majority of genes with an average CV across their CAs that was considered low enough. These genes were assigned to the clusters on the basis of numerical rather than categorical similarity, according to step 3. The second significance metric could be utilized on these 10 subclusters.

Future work will include justifying on a theoretical basis and formalizing the mapping of GO Evidence Codes to CVs, as described in Section 6. We will also be applying our algorithm to more gene expression data sets for organisms on which low quality CAs exist. Furthermore, we will be developing more significance metrics for the M-BILCOM clustering results.

8. CONCLUSION

When clustering low quality data with uncertainties on the data's correctness, we need to develop our ability to integrate data from various sources, including numerical data and categorical data. Furthermore, we need to be able to claim that what we see in a clustering analysis is more reliable or less reliable and, therefore, may or may not be a strong basis for making decisions. In this paper we have described the novel M-BILCOM clustering algorithm for mixed numerical and uncertain categorical data sets that incorporates both CAs and CVs on the correctness of the annotations. This clustering algorithm inspired us to define two new significance metrics for extracting from each cluster the most reliable CAs that form a strong basis for deriving conclusions on the likely annotations of other objects in the cluster. We showed that these significance metrics can be successfully used for finding out which CAs are the most significant ones in a cluster. For genomic data sets we tried to apply the significant CAs to other genes in the cluster, as part of functional prediction. Furthermore, we experimented with our clustering tool on highly noisy simulated data sets for which the correct results were known; we showed that our clustering algorithm can reliably identify the cluster structure in such simulated data sets.

9. REFERENCES

[1] Andreopoulos, B., An, A. and Wang, X. (2005) BILCOM: Bi-level Clustering of Mixed Categorical and Numerical Biological Data. Technical report CS-2005-01. York University, Department of Computer Science.

[2] Andreopoulos, B., An, A. and Wang, X. (2004) MULIC: Multi-Layer Increasing Coherence Clustering of Categorical Data Sets. Technical report CS-2004-07. York University, Department of Computer Science.

[3] Andreopoulos, B., An, A. and Wang, X. (2003) Significance Metrics for Clusters of Mixed Numerical and Categorical Yeast Data. Technical report CS-2003-12. York University, Department of Computer Science.

[4] Adryan B. and Schuh R. (2004) Gene ontology-based clustering of gene expression data, Bioinformatics, Nov 2004; 20: 2851 - 2852.

[5] Ben-Dor A., Shamir R., Yakhini Z. (1999) Clustering Gene Expression Patterns. Journal of Computational Biology 6(3/4): 281-297.

[6] Brown M.P.S. Grundy W.N., Lin D., Cristianini N., Sugnet C.W., Furey T.S., Manuel Ares, and Haussler D. (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines. PNAS 97(1), 262-267.

[7] Cherepinsky V., Feng J., Rejali M. and Mishra B. (2003) Shrinkage-Based Similarity Metric for Cluster Analysis of Microarray Data. PNAS 100(17): 9668-9673.

[8] Dwight SS, Harris MA, Dolinski K, Ball CA, Binkley G, Christie KR, Fisk DG, Issel-Tarver L, Schroeder M, Sherlock G, Sethuraman A, Weng S, Botstein D, Cherry JM. (2002) Saccharomyces Genome Database provides secondary gene annotation using the Gene Ontology. *Nucleic Acids Research* 30: 69-72.

[9] Eisen, M.B. & Brown, P.O. (1999) DNA arrays for analysis of gene expression. *Methods Enzymol.* 303, 179-205.

[10] Eisen MB, Spellman PT, Brown PO, Botstein D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA. 1998 Dec 8;*95(25):14863-8.

[11] Fasulo D. (1999) An Analysis of Recent Work on Clustering Algorithms, Technical Report # 01-03-02, Department of Computer Science & Engineering, University of Washington.

[12] The Gene Ontology Consortium (2001). Creating the gene ontology resource: design and implementation. *Genome Research* 11: 1425-1433. http://www.geneontology.org/GO.evidence.html

[13] Goebel, M. & Gruenwald, Le (1999). A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explorations* 1, 20-33 .

[14] Golub, T. R. et al. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286, 531-537.

[15] Grambeier J., Rudolph A. (2002) Techniques of Cluster Algorithms in Data Mining. *Data Mining and Knowledge Discovery* 6: 303-360.

[16] Guha S., Rastogi R., Shim K. (2000). ROCK: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems* 25(5): 345-366.

[17] Hartigan, J. A. (1975) Clustering algorithms. (John Wiley and Sons, New York, 1975).

[18] Huang Z. (1998) Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery* 2(3): 283-304.

[19] Huang, Z. (1997) Clustering Large Data Sets with Mixed Numeric and Categorical Values. *Knowledge discovery and data mining: techniques and applications.* World Scientific.

[20] Lord P.W., Stevens R.D., Brass A. and Goble C.A. (2003). Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. *Bioinformatics* 19: 1275-83.

[21] Pasquier C., Girardot F., Jevardat de Fombelle K., and Christen R. (2004) THEA: Ontology driven analysis of microarray data. Bioinformatics, Nov 2004; 20: 2636 - 2643.

[22] Slonim D.K., Tamayo P., Mesirov J.P., Golub T.R., and Lander E.S.. (2000) Class prediction and discovery using gene expression data. *Proceedings of the Fourth Annual Conference on Computational Molecular Biology (RECOMB),* 263-272.

[23] Spellman, P.T. et al. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces Cerevisiae by microarray hybridization. Mol. Biol. Cell 9, 3273-3297 (1998).

[24] Stutz J. and Cheeseman P. (1995) Bayesian Classification(AutoClass): Theory and results. Advances in Knowledge Discovery and Data Mining, 153-180, Menlo Park, CA, AAAI Press.

[25] Wu L.F., Hughes T.R., Davierwala A.P., Robinson M.D., Stoughton R. and Altschuler S.J. (2002). Large-scale Prediction of Saccharomyces Cerevisiae Gene Function Using Overlapping Transcriptional Clusters. *Nature Genetics* 31:255-265.