A Framework for Clustering Categorical Data based on Empirical Distributions

Bill Andreopoulos

Aijun An

Xiaogang Wang

Technical Report CS-2006-01

January 2006

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

A Framework for Clustering Categorical Data based on Empirical Distributions

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

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

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

ABSTRACT

Density-based clustering algorithms often have a solid mathematical basis. A challenge involved in applying density-based clustering to categorical data sets is that the 'cube' of attribute values has no ordering defined. In this paper we propose the CEED framework for *clustering categorical data based on its empirical probability distribution*. CEED offers a basis for designing categorical clustering algorithms that balance the tradeoff of accuracy and speed. The advantages of CEED are: (*i*) it offers a probabilistic basis for clustering categorical data, (*ii*) it minimizes the user-specified input parameters, (*iii*) it is insensitive to the order of the input objects, (*iv*) it can discover clusters of arbitrary shapes and sizes. We present a faster approximation of CEED called the MULIC algorithm, which is designed for categorical data sets with a *multi-layered* structure. We evaluate CEED and MULIC on various data sets, including protein interaction data. CEED produces more accurate results than other algorithms on small-dimensional data sets. MULIC can find the multi-layered structure of special data sets such as protein interaction data better than other algorithms and has comparable runtimes.

Keywords

clustering, categorical, empirical, distribution, framework

1. INTRODUCTION

A growing number of clustering algorithms for categorical data have been proposed in recent years, along with interesting applications such as partitioning large software systems and protein interaction data [3, 7, 14]. In the past, polynomial time approximation algorithms have been designed for NP-hard partitioning algorithms [10]. Moreover, it has recently been shown that the "curse of dimensionality" involving efficient searches for approximate nearest neighbors in a metric space can be dealt with, if and only if, we assume a bounded dimensionality [13, 22]. Clearly, there are tradeoffs of efficiency and approximation involved in the design of categorical clustering algorithms. A framework for clustering categorical data would be a basis for algorithms that approximate the framework. This would allow designing and comparing categorical clustering algorithms on a more formal basis.

A categorical data set with m attributes is viewed as an m-dimensional 'cube', offering an empirical distribution basis for density-based clustering. A cell of the cube is mapped to the number of objects having values equal to its coordinates. Clusters in such a cube are regarded as *subspaces* in which the objects are dense and are separated by subspaces of low object density. Clustering the cube poses several challenges:

(*i*) Since there is no ordering of attributes or values, the cube cells have no ordering either. The search for dense subspaces could have to consider several orderings of the cube's dimensions to identify the best clustering.

(*ii*) The density of a subspace is often defined relative to a user-specified value such as a radius. However, different radii are preferable for different subspaces of the cube [4]. In dense subspaces where no information should be missed, the search is more accurately done 'cell by cell' with a low radius of 1. In sparse subspaces a higher radius may be preferable to aggregate information. The cube search could start from a low radius and gradually move to higher radii.

We present the *CEED framework* for clustering categorical data that addresses the above challenges. CEED clusters the m-dimensional *cube* representing the empirical distribution of a set of objects with m categorical attributes. To find its dense subspaces, CEED considers an object's neighbors to be all objects that are within a *radius* of *maximum dissimilarity*. Objects' neighborhoods are insensitive to attribute or value ordering. Clusters start from the most dense subspaces of the cube. Clusters expand outwards from a dense subspace, by connecting nearby dense subspaces. Figure 1 shows simple examples of creating and expanding clusters in a 3-dimensional data set. The radius specifies the maximum number of dimensions by which neighbors can differ. CEED can discover clusters of arbitrary shapes and sizes.

CEED offers a framework for categorical clustering, rather than an algorithmic solution. There is often a tradeoff between clustering accuracy and time efficiency. Many clustering problems are NP-complete [10, 22]. The time requirement forbids an exhaustive search of all possible clusters and

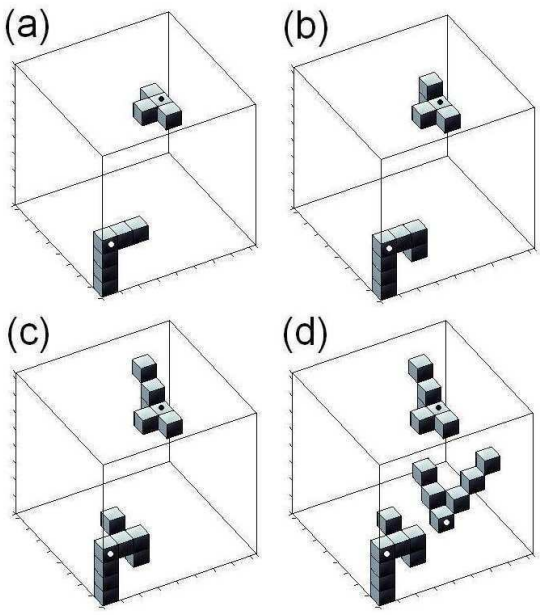

Figure 1: Clustering examples. The cube has no ordering. Cells in a cluster are shown near one another for illustration clarity. A cluster is a dense subspace with a 'central' cell marked with a dot. (a) radius=1, two new clusters. (b) radius=1, clusters expand. (c) radius=2, clusters expand. (d) radius=2, one new cluster.

assignments of objects to find the clustering that gives the best value for some quality metric. Clustering algorithms are often approximations to some ideal solution.

We present the *MULIC algorithm* as a faster approximation of CEED. MULIC is motivated by clustering of categorical data sets that have a *multi-layered* structure. For instance, in protein interaction data a cluster often has a center of very similar proteins surrounded by peripheries of less similar proteins [5, 8]. On such data, MULIC's accuracy outperforms other algorithms that create a flat clustering.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents the CEED framework. Section 4 describes the MULIC clustering algorithm and its relation to CEED. Sections 5 and 6 discuss the experiments and protein interaction data results. Section 7 discusses the runtimes. Section 8 concludes the paper.

2. RELATED WORK

Section 2.1 provides an overview of density-based clustering algorithms for categorical and numerical data. Sections 2.2-2.3 describe several more categorical clustering algorithms.

2.1 Density-based Clustering

Density-based clustering approaches use a local density criterion. Clusters are subspaces in which the objects are dense and are separated by subspaces of low density. Advantages of many of these algorithms include time efficiency and ability to find clusters of arbitrary shapes. Some of these algorithms take a large number of input parameters. Some can not always identify clusters of very different densities. In some of these algorithms the central subspace of a cluster can not always be distinguished from the rest of the cluster based on a higher density [7, 14, 32]. Our approach deals with these cases since a central subspace often has a higher density and the radius relaxes gradually. Density-based algorithms for *categorical* data include CACTUS [11], COOLCAT [6], CLICK [32], ROCK [15], CLOPE [31], STIRR [12].

CACTUS assumes the existence of a minimum size of the distinguishing attribute value sets that uniquely occur within one cluster. The distinguishing sets in CACTUS rely on the assumption that a cluster is uniquely identified by a core set of attribute values that seldomly occur in other clusters. While this assumption may hold true for many real world datasets, it may be unnatural for clustering [32]. CACTUS has difficulty finding clusters within clusters or subspace clusters [13]. The distinguishing sets are extended to candidate cluster projections, which are used to generate cluster candidates of higher dimensionality. Too many candidate cluster projections may result in too many clusters.

COOLCAT is an entropy-based algorithm for categorical clustering. COOLCAT first identifies a set of k initial maximally dissimilar points from the data set to create initial clusters. Clusters are created by "cooling" them down, *i.e.*, reducing their entropy. All remaining tuples of the data set are placed in one of the clusters such that, at each step, the increase in the entropy of the resulting clustering is minimized. Naturally, this approach is highly dependent on the order of selection [7, 14].

CLICK creates a graph representation of a data set. Vertices are categorical values and an edge is a co-occurrence of values in an object. A cluster is a k-partite clique such that most pairs of vertices are connected by an edge. CLICK may return too many clusters or too many outliers [7, 14].

ROCK is an adaptation of an agglomerative hierarchical clustering algorithm for categorical data. It does not require the user to specify the number of clusters. ROCK assumes a similarity measure between tuples and defines a "link" between two tuples whose similarity exceeds a threshold w. Initially, each tuple is assigned to a separate cluster and then clusters are merged repeatedly according to the closeness between clusters. The closeness between clusters is defined as the sum of the number of "links" between all pairs of tuples, where the number of "links" represents the number of common neighbors between two clusters. The algorithm exhibits cubic complexity in the number of objects, which makes it unsuitable for large data sets [7, 14, 32].

CLOPE is a clustering algorithm for categorical and transactional data. CLOPE uses a heuristic method of increasing the height-to-width ratio of the cluster histogram. Its advantages include fast performance and scalability to large data sets with high dimensions. Its main disadvantage is that the accuracy of the results often suffers [7, 14].

STIRR is an iterative algorithm. STIRR applies a linear dynamical system over multiple copies of a hypergraph of

weighted attribute values, until a fixed point is reached. STIRR is sensitive to the input ordering and lacks a definite convergence. STIRR looks for relationships between all attribute values in a cluster. It is not designed to distinguish the center of a cluster based on higher density or a subset of the attribute values. The notion of weights is non-intuitive and several operators are left to the user to define. The final detected clusters are often incomplete [32].

Density-based clustering algorithms for *numerical* data include DBSCAN [9], OPTICS [4], DENCLUE [17], CLIQUE [1], WaveCluster [28]. Some density-based approaches such as DENCLUE, CLIQUE, Wave-Cluster are also grid-based, meaning that a histogram is constructed by partitioning the data space into a number of non-overlapping regions. Regions with relatively many objects are cluster centers. The size of the regions often must be specified by the user in this approach and it affects the success of the result.

In DBSCAN for each point of a cluster the neighborhood of a given radius (ϵ) has to contain at least a minimum number of points ($MinPts$) where ϵ and $MinPts$ are input parameters. OPTICS finds an ordering of the data which is consistent with DBSCAN. OPTICS covers a spectrum of all different $\epsilon' \leq \epsilon$. Both DBSCAN and OPTICS require a number of parameters to be specified by the user which will affect the quality of the result. However, OPTICS considers that different parts of the data space could require different parameters. DBSCAN and OPTICS have difficulty identifying clusters within clusters [13].

DENCLUE differs from previous approaches in that it pins density to a point in the attribute space instead of an object. It manages information about regions that contain objects in a tree-based access structure. Although DENCLUE has a large number of input parameters, it is very efficient with a complexity of $O(N)$.

CLIQUE is grid-based and assumes that points lie in vector space. It partitions the space into non-overlapping rectangular units and identifies the dense units. Then, it intersects dense units to form a search space of higher dimensionality. CLIQUE detects subspaces of the highest dimensionality such that high-density clusters exist in those subspaces. Its input parameters are the grid size and a global density threshold for clusters. CLIQUE considers only hyper-rectangular clusters and projections parallel to the axes [13].

WaveCluster is a multiresolution clustering algorithm that uses the Wavelet transform to transform the original data and find dense regions in the transformed space. The Wavelet transform is useful because it has the ability to suppress weaker information, effectively being a good method for noise removal. This results in two main benefits: since we are dealing with less information, we can speed up the mining process and it is easier to detect clusters at varying levels of accuracy. WaveCluster is fast with a time complexity of $O(N)$. However, it is only applicable to low-dimensional data [7, 14, 13]. Its input parameters are the number of grid cells for each dimension, the wavelet to use and the number of applications of the Wavelet transform.

2.2 k-Modes Categorical Clustering

The k-Modes categorical clustering algorithm requires the user to specify the number of clusters as an input parameter. The algorithm builds and refines the specified number of clusters [18]. K-Modes assigns a mode to each cluster as a summary of the cluster's most frequent attribute values. The mode of cluster c is a vector $\mu_c = \{\mu_{c1}, \cdots, \mu_{cm}\}$ where μ_{ci} is the most frequent value for the ith attribute in c.

A dissimilarity metric is used to choose the nearest cluster to an object, by computing the dissimilarity between the cluster's mode and the object. Let $o = \{o_1, \cdots, o_m\}$ be an object where o_i, $i = 1 \cdots m$, is the ith attribute's value. The dissimilarity between o and μ_c is defined as:

$$dist(o, \mu_c) = \sum_{i=1}^{m} \delta(o_i, \mu_{ci}) \; where \; \delta(o_i, \mu_{ci}) = \begin{cases} 1, & \text{if } o_i \neq \mu_{ci} \\ 0, & \text{if } o_i = \mu_{ci} \end{cases}$$

2.3 AutoClass and LIMBO

AutoClass is a clustering algorithm for categorical and numerical data [29]. It does not require the user to specify the number of clusters. AutoClass uses a Bayesian method for determining the optimal classes. AutoClass takes a prior distribution for each attribute in each cluster, symbolizing the prior beliefs of the user. It changes the classifications of objects in clusters and changes the means and variances of the distributions, until the means and variances stabilize.

LIMBO is a hierarchical categorical clustering algorithm that builds on the Information Bottleneck (IB) framework for quantifying the relevant information preserved when clustering [3]. LIMBO uses the IB framework to define a distance measure for categorical tuples. LIMBO handles large data sets, using a memory bounded summary for the data.

3. THE CEED CLUSTERING FRAMEWORK

The CEED framework consists of a clustering process for categorical data, based on its empirical probability distribution. We begin with section 3.1 defining the basic concepts. We proceed with section 3.2 describing CEED. Section 3.3 discusses the advantages and drawbacks of CEED.

3.1 Basics

We are given a data set of objects S (which might contain duplicates) with m categorical attributes, $X_1, \cdots, X_m$. Each attribute X_i has a domain D_i with a finite number of d_i possible values. The empirical distribution S^m includes the collection of possibilities defined by the cross-product (or cartesian product) of the domains, $D_1 \times \cdots \times D_m$. This can also be viewed as an m-dimensional 'cube' with $\prod_{i=1}^{m} d_i$ cells (positions). A cell of the cube represents the unique logical intersection in a cube of one member from every dimension in the cube. The function λ maps a cell $\mathbf{x} = (x_1, \cdots, x_m) \in S^m$ to the nonnegative number of objects in S with all m attribute values equal to $(x_1, \cdots, x_m)$:

$$\lambda : \{(x_1, \cdots, x_m) \in S^m\} \rightarrow N.$$

We define the CEED *hyper-cube* $C(\mathbf{x_0}, r) \subset S^m$, centered at cell $\mathbf{x_0}$ with radius r, as follows:

$$C(\mathbf{x_0}, r) = \{\mathbf{x} : \mathbf{x} \in S^m \; and \; dist(\mathbf{x}, \mathbf{x_0}) \leq r \; and \; \lambda(\mathbf{x}) > 0\}.$$

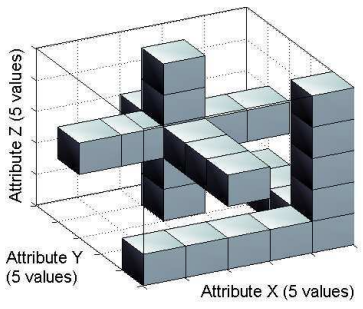

Figure 2: Two CEED 'hyper-cubes' in a 3D cube, for r=1.

The $dist(\cdot)$ is a distance function. The *Hamming* distance is defined as follows:

$$HD(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} \delta(x_i, y_i) \;\; where \;\; \delta(x_i, y_i) = \begin{cases} 1, & \text{if } x_i \neq y_i \\ 0, & \text{if } x_i = y_i \end{cases}$$

Figure 2 illustrates two CEED hyper-cubes in a 3-dimensional cube. Since r=1, the hyper-cubes are visualized like 'crosses' in 3D and are not shown as actually having a cubic shape. A hyper-cube excludes cells for which λ returns 0. A hyper-cube can not equal S^m, unless $r = m$ and $\forall \mathbf{x} \in S^m \; \lambda(\mathbf{x}) > 0$. Normally only a subset of S^m will belong in a hyper-cube.

The *density* of hyper-cube $C(\mathbf{x_0}, r) \subset S^m$ involves the sum of function λ evaluated over all of the hyper-cube's cells:

$$density(C(\mathbf{x_0}, r)) = \sum_{c \in C(\mathbf{x_0}, r)} \frac{\lambda(c)}{|S|}.$$

This probability density is the likelihood that the hyper-cube contains a random object from S. CEED seeks the most dense hyper-cube $C(\mathbf{x_0}, r) \subset S^m$. This is the hyper-cube centered at $\mathbf{x_0}$ which has the maximum likelihood of containing a random object from S. The cell $\mathbf{x_0}$ is a member of the set $\{\mathbf{x} \in S^m : Max(P(\Omega \in C(\mathbf{x}, r)))\}$, where Ω is a discrete random variable that assumes a value from set S.

The *distance* between two clusters G_i and G_j is the distance between the nearest pair of their objects, defined as:

$$Distance(G_i, G_j) = Min\{dist(\mathbf{x},\mathbf{y}) : \mathbf{x} \in G_i \; and \; \mathbf{y} \in G_j\}.$$

Clusters G_i and G_j are *directly connected relative to* r if $D(G_i, G_j) \leq r$. Clusters A and B are *connected relative to* r if: A and B are directly connected relative to r, or if: there is a chain of clusters $C_1, C_2, \cdots, C_n$, $A = C_1$ and $B = C_n$, such that C_i and C_{i+1} are directly connected relative to r for all $1 \leq i < n$.

3.2 The CEED Clustering Process

Figure 3 shows the CEED clustering process. The radius r's default initial value is 1. G_k represents the kth cluster formed. The remainder set, $R = \{\mathbf{x} : \mathbf{x} \in S^m \; and \; \mathbf{x} \notin G_i, i = 1, \cdots, k\}$, is the collection of unclustered cells after the formation of k clusters.

Input: empirical probability distribution S^m.
Output: a hierarchy of clusters.
Method:

$r = 1$. //radius of hyper-cubes
$R = S^m$. //set of unclustered cells
$k = 0$. //number of leaf clusters
$k_r = 0$. //number of clusters at level r
$G_k = null$. //kth cluster
$U = null$. //set of hyper-cube centers

Step 1: Find $\mathbf{x_0} \in R$ such that
$density(C(\mathbf{x_0}, r)) = \max_{\mathbf{x_w} \in R} density(C(\mathbf{x_w}, r))$.

If $density(C(\mathbf{x_0}, r)) \leq \frac{1}{|S|}$, then
(1) $r = r + 1$.
(2) If $k_{r-1} > 1$, then
(3) Merge clusters that are connected relative to r.
(4) $k_r = \#merged + \#unmerged\, clusters$.
(5) Repeat Step 1.

Step 2: Set $\mathbf{x_c} = \mathbf{x_0}$, $k = k + 1$, $G_k = C(\mathbf{x_c}, r)$, $R = R - C(\mathbf{x_c}, r)$ and $U = U \cup \{\mathbf{x_c}\}$.

Step 3: Examine $C(\mathbf{x_c}, r)$ and find a point $\mathbf{x}^*$ such that $\mathbf{x}^* \in C(\mathbf{x_c}, r)$ and $\mathbf{x}^* \notin U$ and
$density(C(\mathbf{x}^*, r)) = \max_{\mathbf{x_w} \in T(\mathbf{x_c}, r, R)} density(C(\mathbf{x_w}, r))$
where $T(\mathbf{x_c}, r, R) = \{\mathbf{x} : \mathbf{x} \in C(\mathbf{x_c}, r) \;\; and \;\; \mathbf{x} \notin U\}$.

Step 4: If $density(C(\mathbf{x}^*, r)) > \frac{1}{|S|}$, then
Update current cluster G_k: $G_k = G_k \cup C(\mathbf{x}^*, r)$.
Update R: $R = R - C(\mathbf{x}^*, r)$.
Update U: $U = U \cup \{\mathbf{x}^*\}$.
Re-set the new center: $\mathbf{x_c} = \mathbf{x}^*$.
Go to Step 3.
Otherwise, move to the next step.

Step 5: Set $k_r = k_r + 1$.
If $k_r > 1$, then execute lines (3) – (4).
If $r < m$ and $density(R) > 0$, then go to Step 1.

Step 6: While $r < m$, execute lines (1) – (4).

Figure 3: The CEED clustering framework.

Step 1 retrieves the most *dense* hyper-cube $C \subset S^m$ of radius r. Step 1 checks that the most dense hyper-cube represents more than one object ($density(C(\mathbf{x_0}, r)) > \frac{1}{|S|}$), since otherwise the cluster will not expand, ending up with one object. If the hyper-cube represents zero or one object, then r is incremented. *Step 2* creates a new cluster. Starting from an existing cluster, *step 3* tries to move to the most dense hyper-cube of radius r nearby. If a dense hyper-cube is found near the cluster, then in *step 4* the cluster expands by collecting the hyper-cube's cells. This is repeated for a cluster until no such connection can be made.

Objects are clustered until $r = m$, or R represents zero objects (*step 5*). For most data sets, most objects are likely to be clustered long before $r = m$.

Initially $r = 1$ by default, since most data sets contain subsets of similar objects. Such subsets are used to initially identify dense hyper-cubes. When r is incremented, a special process *merges* clusters that are connected relative to r.

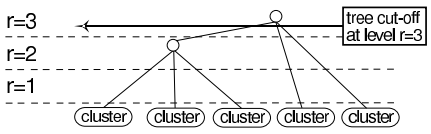

Figure 4: A link (circle) represents two or more merged clusters.

Although the initial $r = 1$ value may result in many clusters, similar clusters are merged gradually. The merging process considers previously merged clusters as one cluster. As Figure 4 shows, a merge is represented as a *link* at level r between two or more clusters or links. This process gradually constructs one or more cluster tree structures, resembling hierarchical clustering [20]. The user specifies a cut-off level (e.g. $r = 3$) to cut the tree(s); then, links at levels lower than the cut-off level are extracted as merged clusters. The cut-off level could be set based on Mojena's rule [26]. *Step 5* checks if a newly formed cluster is connected to another cluster relative to r and if so links them at level r. *Step 6* continues linking existing clusters into a tree, until $r = m$. By allowing r to reach m, a whole tree is built. At the top of the tree, there is a single cluster containing all the objects.

3.3 CEED Discussion

CEED can often distinguish the central hyper-cube of a cluster from the rest of the cluster because of its higher density. The radius relaxes progressively, implying that CEED can find clusters of different densities. An optional CEED feature is to identify as outliers any objects clustered after r exceeds some user-specified *threshold* in *steps 1-5*.

It is costly to cluster high-dimensional data sets with the CEED process. A computing of hyper-cubes is costly, although several effective indexation schemes exist. The CEED search of a cube in memory could rely on R^*-tree indexation or R^+-tree indexation, meaning $O(log(N))$ rather than $O(N)$ fetches per search [16, 23, 27]. Identifying the dense hyper-cubes in a cluster may require testing $\binom{m}{r}$ possibilities. Finally, manipulating a cube in memory is costly with the limited capacity of modern computers.

4. MULIC AS AN APPROXIMATION OF CEED

MULIC stands for *multiple layer clustering* of categorical data. MULIC is an approximation of CEED. MULIC balances clustering accuracy with time efficiency. The MULIC algorithm is motivated by data sets the cluster structure of which can be visualized as shown in Figure 5. In such data sets a cluster often has a center of objects that are very similar to one another, along with peripheral objects that are less similar to the central objects. Such data sets include protein interaction data, large software systems and others [5, 8, 21]. Users in these domains often value clustering accuracy more than speed.

MULIC does not store the cube in memory and makes simplifications to decrease the runtime. A MULIC cluster starts from a dense area and expands outwards via a radius represented by the ϕ variable. When MULIC expands a cluster it

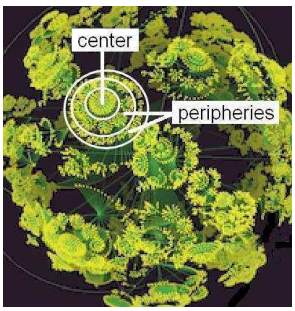

Figure 5: A cluster has a center surrounded by peripheral areas.

does not search all member objects as CEED does. Instead, it uses a mode that summarizes a cluster's content. MULIC produces layered clusters, does not require the user to specify the number of clusters and can identify outliers. Section 4.1 describes MULIC and 4.2 compares MULIC with CEED.

4.1 The MULIC Clustering Algorithm

Each MULIC cluster has a mode associated with it [18]. Assuming that the objects in the data set are described by m categorical attributes, the mode of cluster c is a vector $\mu_c = \{\mu_{c1}, \cdots, \mu_{cm}\}$ where μ_{ci} is the most frequent value for the ith attribute in the given cluster c. The MULIC clustering algorithm ensures that when an object o is clustered it is inserted into the cluster c with the least dissimilar mode μ_c. The default dissimilarity metric between o and μ_c is the Hamming distance presented in Section 3.1.

Figure 6 shows the main part of the MULIC clustering algorithm. The algorithm stores objects in S. An optional preprocessing step, described in [2], orders the objects by frequency of their attribute values such that the modes will likely contain the most frequent values. The first object is inserted into a new cluster, the object becomes the mode of the cluster and the object is removed from S. Then, it continues iterating over all objects that have not been assigned to clusters yet, to find the nearest cluster. In all iterations, the nearest cluster for each unclassified object is the cluster with the least dissimilarity between the cluster's mode and the object, as computed by the dissimilarity metric. A cluster's mode is updated through the process to reflect the most frequent values over all objects in the cluster. A mode can be viewed as moving to the 'center' of the cluster.

The variable ϕ is motivated by the r radius of CEED. It specifies that only objects within a dissimilarity distance of ϕ from the nearest cluster's mode are allowed to be inserted in the cluster. Thus, the variable ϕ in combination with the mode of a cluster are used for selecting the subspaces of the m-dimensional cube that will join a cluster. If the number of different values between the object and the nearest cluster's mode is greater than ϕ then the object is inserted in a new cluster on its own, since the object is outside distance ϕ of the mode; else, the object is inserted in the nearest cluster and the mode is updated. Initially ϕ equals 1, meaning that the dissimilarity has to be very small between an object and

Input: a set S of objects.
Parameters: (1) $\delta\phi$: the increment for ϕ.
(2) *threshold* for ϕ : the maximum number of values that can differ between an object and the mode of its cluster.
Default parameter values: (1) $\delta\phi = 1$.
(2) *threshold* = the number of categorical attributes m.
Output: a set of clusters.
Method:

1. Insert the first object into a new cluster, use the object as the mode of the cluster, and remove the object from S.
2. Initialize ϕ to 1.
3. Loop through the following until S is empty or ϕ is greater than the specified *threshold*
   a. For each object o in S
      i. Find o's nearest cluster c by using the dissimilarity metric to compare o with the modes of all existing cluster(s).
      ii. If the number of different values between o and c's mode is larger than ϕ, insert o into a new cluster
      iii. Otherwise, insert o into c and update c's mode.
      iv. Remove object o from S.
   b. For each cluster c, if there is only one object in c, remove c and put the object back in S.
   c. If in this iteration no objects were inserted in a cluster with $size > 1$, increment ϕ by $\delta\phi$.

Figure 6: The MULIC clustering algorithm.

the nearest cluster's mode and only neighbors of a mode are allowed to join its cluster.

Similar to the way CEED seeks dense hyper-cubes, MULIC seeks at least *two* objects that are within a dissimilarity distance of ϕ from each other. The clusters that persist through the process are only those containing at least two objects. Objects assigned to clusters of size greater than one are removed from the set S, therefore those objects will not be re-clustered. Clusters of size one are removed at the end of each iteration, therefore their objects will be re-clustered. This is motivated by CEED's hyper-cube density criterion.

At the end of each iteration, if no objects have been inserted in clusters of size greater than one, then the variable ϕ is incremented by $\delta\phi$. Thus, at the next iteration the criterion for inserting objects in clusters will be more flexible. This is motivated by CEED's expanding clusters and increasing r.

MULIC can create new clusters at *any* value of ϕ, just like CEED can create new clusters at any value of r. Figure 7 illustrates what the results of MULIC look like, showing that cluster 3 is created when $\phi = 2$. Each cluster consists of 'layers' of objects. The layer in which an object is inserted depends on the value of ϕ. The layer of an object reflects the object's dissimilarity to the mode when the object was assigned to the cluster. MULIC starts by inserting as many objects as possible in top layers - such as layer 1 - and then moves to lower layers, creating them as ϕ increases. Low layers correspond to high values of ϕ and their objects have a high dissimilarity to the cluster mode. This implies that objects in lower layer tend to be more dissimilar to the objects in top layers. This is *consistent with the multi-layered structure* of the data sets MULIC is designed for.

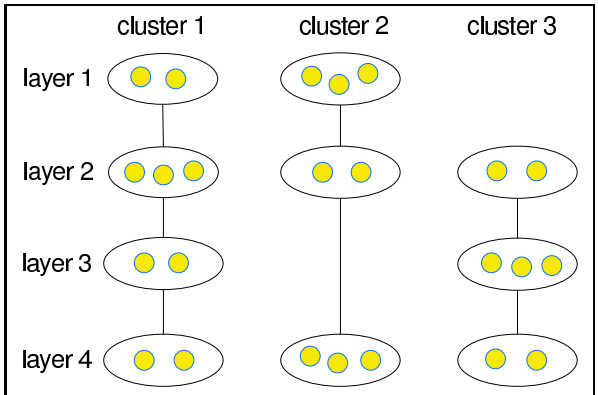

Figure 7: A MULIC cluster consists of one or more layers representing dissimilarities between the objects and mode. Ovals represent layers and circles represent objects.

MULIC can eventually classify all objects in clusters, since ϕ can continue increasing until it equals the number of attributes m. Alternatively, setting an upper *threshold* for ϕ allows outliers to be detected.

4.1.1 *Merging of clusters*

Sometimes the dissimilarity of the top layers of two clusters is less than the dissimilarity of the top and bottom layers of one of the two clusters. To avoid this, after the clustering process MULIC can merge pairs of clusters whose top layers' modes' dissimilarity is less than the maximum layer depth of the two clusters. For this purpose, MULIC preserves the modes of the top layers of all clusters. This process, described in Figure 8, reduces the total number of clusters and may improve the quality of the results.

for (c = first cluster to last cluster)
  for ($d = c+1$ to last cluster)
    if the dissimilarity between c's mode and d's mode is less than the maximum layer depth of c and d, merge c into d and break the inner loop;

The dissimilarity between two modes ($\mu_c = \{\mu_{c1}, \cdots, \mu_{cm}\}$ and $\mu_d = \{\mu_{d1}, \cdots, \mu_{dm}\}$) is defined as:

$$dissimilarity(\mu_c, \mu_d) = \sum_{i=1}^{m} \delta(\mu_{ci}, \mu_{di})$$

$$\text{where } \delta(\mu_{ci}, \mu_{di}) = \begin{cases} 1, & \text{if } \mu_{ci} \neq \mu_{di} \\ 0, & \text{if } \mu_{ci} = \mu_{di} \end{cases}$$

Figure 8: Merging clusters.

4.2 MULIC Discussion

MULIC is an approximation of CEED. The tradeoffs between accuracy and time efficiency are described as follows:

(*i*) When creating a cluster, CEED searches the cube to retrieve the most dense hyper-cube relative to r representing two or more objects, which is costly. MULIC creates a cluster if *two* or more objects are found within a dissimilarity

distance of ϕ from each other, likely indicating a dense subspace. Clusters of size one are filtered out. MULIC's ϕ variable is motivated by CEED's radius r. The initial objects clustered with MULIC affect the modes and the clustering. For this issue we proposed in [2] to order the objects by frequency of their attribute values in a preprocessing step.

(*ii*) When expanding a cluster CEED searches the member cells to find dense hyper-cubes relative to r, which is costly. MULIC instead uses a 'mode' as a summary of a cluster's content and only clusters objects within a distance of ϕ. MULIC increases ϕ by $\delta\phi$ when no new objects can be clustered, which is motivated by CEED's increasing r. MULIC can create new clusters at any value of ϕ, just like CEED can create new clusters at any value of r. Although MULIC can find clusters of arbitrary shapes by increasing ϕ, it loses some of CEED's ability in this realm.

(*iii*) MULIC's cluster merging is motivated by CEED's merging. Even though this process does not organize clusters in a tree like CEED does, it could be modified to do so. A tree could be constructed by merging pairs of clusters in increasing order of their dissimilarity. We chose not to construct a tree since we preferred the clusters to have a clear separation and not to have to specify a tree cut-off for MULIC applications, like the one discussed in section 6.

5. PERFORMANCE EVALUATION

In order to evaluate the applicability of CEED and MULIC to the clustering problem, first we use the categorical data sets described in Table 1 obtained from the UCI Repository of Machine Learning [24]. Objects have class labels defined based on some domain knowledge. We ignore class labels during clustering. We compare the CEED and MULIC results to those of k-Modes [18], ROCK [15] and AutoClass [29]. To evaluate the clustering quality we use *HA Indexes* [19] and an *Entropy-based metric*. HA Indexes is a class-label-based evaluation, which penalizes clustering results with more or fewer clusters than the defined number of classes. Since the class labels may or may not be consistent with the clustering structure and dissimilarity measure used, we also use the Entropy-based metric [14]. This penalizes non-uniformity of attribute values in a cluster. In this section we only discuss the results for *zoo* and *soybean-data*. We give the full set of results in [2], along with comparisons to more clustering algorithms such as CLOPE. We only present the best results in this paper because of space limitations. The MULIC value of $\delta\phi$ is 1, *threshold* equals the number of attributes m and clusters are *not* merged.

Data Set	Objects	Attribs	Classes
Soybean-small	47	35	4
Zoo	101	16	7
Soybean-data	307	35	19
Soybean-test	376	35	20
Congressional Vote	435	16	2
Balance Scale	625	4	3
Contraceptive Method	1,473	10	2
Mushroom	8,124	22	2

Table 1: The categorical test data sets.

For *zoo* we implement the CEED process described in Section 3. CEED clusters the *zoo* cube starting from $r = 1$, *i.e.*, a cell by cell cube search. For *zoo* we cut off CEED's tree of merged clusters at level $r = 1$, since *zoo* is a rather dense cube with many nonzero cells and we do not want to aggregate information in the cube. For *soybean-data* we cut off CEED's tree of merged clusters at level $r = 9$. *Soybean-data* is a sparse cube of mostly '0' cells, since the data set has 35 dimensions but only 307 objects. We do not store the entire *soybean-data* cube in memory, but just the 307 objects, since the cube is too large to fit in memory. Moreover, it would be too slow to search the *soybean-data* cube.

We apply each clustering algorithm on 10 random orderings of the objects of each data set and we report the average result. For the k-Modes and ROCK experiments, we set the number of clusters k to the number of classes in the data set, as well as to larger numbers, and we use the k value giving the best results. For example, for the *mushroom* data set we try k values between 2 and 1500. For k-Modes, we set the convergence *threshold* to 0 and we set the modes of the initial clusters equal to the first objects clustered. For ROCK, we set the θ parameter to values ranging from 0.05 to 0.9. For AutoClass, we do not specify the number of clusters as it considers varying numbers of clusters from a minimum of 2; we set the prior distribution to the single multinomial distribution, with no attributes ignored.

5.1 HA Indexes

Given a data set of objects S, suppose that:

- U represents the partition known or believed to be present in S.
- V is the clustering result by some algorithm.
- a is the number of pairs of objects that are placed in the *same class* in U and in the *same cluster* in V.
- b is the number of pairs of objects in the *same class* in U but *not* in the same cluster in V.
- c is the number of pairs of objects in the *same cluster* in V but *not* in the same class in U.
- d is the number of pairs of objects in *different classes* and *different clusters* in both U and V.

Then, Hubert and Arabie [19] define: $HA\ Index = \frac{a+d}{a+b+c+d}$.

	zoo			soybean-data		
Tool	*HAI.*	k	*sec*	*HAI.*	k	*sec*
CEED	95%	7	0.002	90.5%	22	0.05
MULIC	92.5%	11	0	88.1%	38	0.05
k-Modes	92.3%	10	0.005	84.6%	25	0.03
ROCK	85.5%	10	0.008	70.2%	25	0.04
AutoClass	89%	6	0.04	71.6%	7	0.13

Table 2: HA Indexes - higher is better.

5.2 Entropy-based Metric

Given N objects, m attributes and k clusters, n_i is the size of cluster c_i, $1 \leq i \leq k$, and D_j is the domain or set of possible values of the jth attribute, $1 \leq j \leq m$. Then, the entropy-based metric is defined as:

$$Entropy\ of\ the\ partition = \frac{\sum_{i=1}^{k}(n_i \times H(c_i))}{N}$$

$$where\ H(c_i) = -\sum_{j=1}^{m}\sum_{v \in D_j} P(c_{ij} = v) \times logP(c_{ij} = v).$$

	zoo			soybean-data		
Tool	*Entr.*	k	*sec*	*Entr.*	k	*sec*
CEED	2.7	7	0.002	11.6	22	0.05
MULIC	2.5	11	0	11.1	38	0.05
k-Modes	3.5	10	0.005	16.12	25	0.03
ROCK	3.8	10	0.008	19.5	25	0.04
AutoClass	3.6	6	0.04	18.6	7	0.13

Table 3: Entropy measures - lower is better.

Tables 2 and 3 show that CEED on small-dimensional data produces results of high quality. Even though all algorithms produce good results on *zoo*, CEED has better HA Indexes and Entropy. CEED's number of clusters on *zoo* and *soybean-data* is close to the class-label-based number of classes. MULIC without merging produces many clusters for these data sets. MULIC has good entropy measures and HA Indexes because the attribute values are quite uniform in clusters. It is interesting how MULIC finds subclusters of very similar animals. For example, the animals 'porpoise', 'dolphin', 'sealion' and 'seal' are clustered together in one MULIC cluster.

6. EVALUATION ON PROTEIN INTERACTION DATA

MULIC is useful for finding complexes in protein interaction data (PID). A complex is a group of proteins that interact at the same time. MULIC differs from traditional k-Cores partitioning of PIDs, which seeks high-degree subspaces [5]. MULIC instead creates *layered* clusters and seeks proteins with similar interaction sets. As a result, MULIC can find more complexes of varying sizes and shapes. We use the yeast *Saccharomyces cerevisiae* PID originating from [30], containing 988 proteins. We refer to this data set as Y^{2K}. The MULIC value of $\delta\phi$ is 3, *threshold* has its default value (988) and clusters are merged.

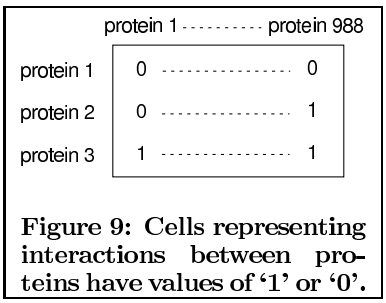

Figure 9: Cells representing interactions between proteins have values of '1' or '0'.

Figure 9 shows the formulation of Y^{2K} for our clustering approach. PID data on an organism is categorical. The objects (proteins) have categorical attribute values that are taken from the set of discrete values ('1','0'). These values have no specified ordering. We represent Y^{2K} as an $N \times N$ symmetric square matrix $A = (a_{ij})$, where N=988 is the number of proteins in Y^{2K}. The rows and columns represent proteins and $a_{ij} = 1$ if there is a known interaction between proteins i and j and $a_{ij} = 0$ otherwise.

The resulting MULIC clusters significantly overlap with many known protein complexes derived from the MIPS yeast database [25]. More than 50% of MULIC clusters match a known MIPS complex. The matching criteria are that a cluster should either have 60% of its proteins contained in a known MIPS complex of a similar size, or have 90% of its proteins contained in a known MIPS complex of a larger size. Table 4 shows a comparison of the MULIC results with the results of the RNSC clustering algorithm [21]. The RNSC results were evaluated using the same matching criteria as in our MULIC evaluation. Even with our strict matching criteria, our number of clusters that match a known MIPS complex is higher and our cluster size is often larger. With MULIC there is a cluster of 79 proteins matching the MIPS complex "550.1.149" of size 88 proteins. Their overlap is 44 proteins.

	Matching clusters	Largest size of a cluster that matches a MIPS complex
MULIC	45	MIPS complex "550.1.149" of size 88 matches MULIC cluster of size 79. Their overlap is 44 proteins.
RNSC	23	MIPS complex of size 29 matches RNSC cluster of size 17. Their overlap is 10 proteins.
k-Modes	18	MIPS complex of size 20 matches k-Modes cluster of size 15. Their overlap is 10 proteins.
AutoClass	10	MIPS complex of size 15 matches AutoClass cluster of size 14. Their overlap is 6 proteins.

Table 4: The number of Y^{2K} clusters matching a MIPS complex and the largest size of a cluster that matches a MIPS complex, for several algorithms.

As another comparison, Bader and Hogue use k-Cores to identify hubs of proteins of degree at least k. They generate a set of 209 protein complexes, of which 54 match the MIPS database in at least 20% of their proteins [5].

MULIC has characteristics specific to PID that allow it to find unknown protein complexes. In PID, there are many complexes of small sizes that have high internal connectivity, where the connectivity is the number of interactions divided by the number of proteins. For example, in the yeast proteome of 6,000 proteins most complexes have sizes of 4-40 proteins. MULIC does not require for the number of clusters to be specified - a new cluster is created when a set of proteins is discovered that have highly overlapping interaction sets. As the process continues MULIC relaxes its criterion for assigning proteins to clusters, forming cluster layers of lower connectivity. This is in accordance with a recent study [8] in which protein complexes were discovered to feature centers of highly co-expressed proteins which mostly display the same deletion phenotype.

The multiple layer structure of the MULIC clusters reveals several things about the structures of the predicted protein complexes that could not be identified with other algorithms. In all of the derived MULIC clusters the top-layer proteins (layer 0 and 1) have the highest connectivity to the other protein members of the cluster. For clusters that match known MIPS complexes, the proteins in the top layer are often 'central' points of connectivity for the matched complex and perhaps even the entire cell. In other words, interactions occur with top-layer proteins more frequently than other proteins in the complex. For example, the well-

studied FKS1p (YLR342W) and FKS2p (YGR032W) proteins have a high connectivity to the other proteins in their complex and were clustered in the top layers of MULIC clusters. The drug caspofungin binds to FKS1p and FKS2p to disturb the interactions of the glucan synthase complex. Thus, a biologist could start by testing a new drug on the proteins in top layers, instead of all proteins in the cluster.

Cluster layers	*Percentage* of proteins that are contained in the matched complex
1-4	75%
7-10	66%
13-19	40%

Table 5: Proteins in top layers of Y^{2K} clusters matching a known MIPS complex are more likely to be contained in the matched complex.

The multiple layer structure of the derived MULIC clusters can be useful in cases where few protein complexes are known for an organism, such as fruitfly and worm. Lab experiments can initially focus on the proteins clustered in top layers. Later, proteins in lower cluster layers can guide the lab experiments on finding protein complexes. Table 5 shows that for the Y^{2K} clusters that match a MIPS complex, the proteins in top layers (1-4) are likely to be contained in the matched complex. While the proteins in bottom layers (7-19) are less likely to be contained in the matched complex. This table was derived by averaging the percentage of the proteins in various layers of matching clusters that are contained in the matched complex.

7. RUNTIME EVALUATION

This Section discusses the MULIC runtimes. The experiments were performed on a Sun Ultra 60 with 256 MB of memory and a 300 MHz processor. MULIC, k-Modes, ROCK and AUTOCLASS are implemented in C/C++. The reason for the low MULIC runtimes presented in this section is that most objects are clustered in the initial iterations when the top layers (e.g. 1-5) are created. For instance, for the *mushroom* data set 5 iterations occurred. Thus, relatively few comparisons between objects and modes are done throughout the process. Section 7.1 compares the runtimes of MULIC to other algorithms on data sets of various sizes. Section 7.2 discusses the MULIC runtimes on larger data sets, ranging from the 8124×22 *mushroom* data set to a 5323×5323 protein interaction data set.

7.1 Comparison with Other Algorithms

Table 6 compares the runtimes of MULIC, k-Modes [18], ROCK [15] and AUTOCLASS [29]. The runtimes of MULIC with $\delta\phi$=1 are better than ROCK and AUTOCLASS, but comparable to k-Modes. Table 7 shows the runtimes of MULIC with $\delta\phi$=3 on data sets varying from 47 to 8,124 objects. MULIC with $\delta\phi$=3 runs slightly faster than k-Modes.

7.2 Larger Data Sets

7.2.1 Mushroom

Table 8 shows comparative runtimes for the 8124×22 *mushroom* data set. MULIC with $\delta\phi = 1$ took less time than other algorithms. The misclassification rate of MULIC on the *mushroom* data set was 0%, measured by averaging over all clusters the percentage of objects with a minority class label in the cluster. The misclassification rate of k-Modes on the *mushroom* data set decreased as the number of clusters increased. For k-Modes the best reported misclassification rate of 0.02% was produced by setting the number of clusters to 1500. However, many objects were placed separately in clusters of size one, which did not happen with MULIC. Initially MULIC produced 1,674 clusters for *mushroom*. After decreasing the number of clusters to 150 by merging, the misclassification rate remained the same.

Data set	**MULIC**	k**Modes**	**ROCK**	AutoClas
Soybean-smal	0	0.005	0.008	0.04
Zoo	0	0.005	0.008	0.04
Soybean-data	0.05	0.03	0.04	0.13
Soybean-test	0.06	0.04	0.055	0.12
Congr. Vote	0.01	0.03	0.1	0.27
Balance Scale	0	0.02	0.15	0.83
Contraceptive	0.14	0.16	2	1.97
Mushroom	1.37	3.1	5	1.23

Table 6: Runtimes in seconds of MULIC ($\delta\phi$=1), k-Modes, ROCK and AutoClass on various data sets.

Data set	**Objects**	**Attribs**	**Seconds**
Soybean-small	47	35	0
Zoo	101	16	0
Soybean-data	307	35	0.03
Soybean-test	376	35	0.03
Congressional Vote	435	16	0.01
Balance Scale	625	4	0
Contraceptive Method	1,473	10	0.12
Mushroom	8,124	22	1.36

Table 7: Runtimes in seconds of MULIC ($\delta\phi$=3).

Algorithm	**Runtime**
MULIC (with $\delta\phi$=1)	1.37 seconds
WEKA Expectation Maximization	$\sim$ 3 minutes
WEKA SimpleKMeans (k = 1500)	Did not terminate
CobWeb	Did not terminate
k-Modes (k = 2 to 1500)	3.1 seconds
ROCK (k = 2 to 1500)	5 seconds
AutoClass	1.23 seconds

Table 8: Runtimes on the *mushroom* data set.

7.2.2 Protein Interaction Data

Table 9 shows MULIC runtimes on PID sets of various sizes. The most costly test run was on Y^{78K} which took seven minutes. The runtimes of MULIC are comparable to those of other algorithms, but MULIC can find more complexes and the cluster structure is more interesting for analysis.

7.3 Computational Complexity

The best-case complexity of MULIC has a lower bound of $\Omega(mNk)$ and its worst-case complexity has an upper bound of $O(mN^2\frac{threshold}{\delta\phi})$. Often $m << N$, since in a categorical data set the number of attributes m is usually smaller than the number of objects N. The cost is related to the number of clusters k generated throughout the process. In most

PID	Proteins	Runtime
Y^{2K}	988	10 seconds
Y^{11K}	2617	30 seconds
Y^{78K}	5323	$\sim$ 7 minutes
F^{4K}	4603	$\sim$ 2 minutes
W^{5K}	3659	$\sim$ 1 minute

Table 9: Runtimes of MULIC on PID sets.

of the runtimes we have presented, the number of clusters k is smaller than the number of objects N and all objects are clustered in the initial iterations, thus N often dominates the cost. The worst-case runtime would occur for the rather unusual data set where all objects were extremely dissimilar to one another, such that the algorithm had to go through all m iterations and all N objects were clustered in the last iteration when $\phi = m$. In this case, at each iteration while $\phi < m$ the number of comparisons between objects and modes would be $\sum_{i=1}^{N} i = \frac{N(N+1)}{2} = O(N^2)$ implying a quadratic runtime. Decreasing the value of *threshold* or increasing the value of $\delta\phi$ often improves the runtime. Changing these parameters does not imply weakening the clustering quality. Decreasing the value of *threshold* is useful for detecting outliers. On some data sets, a value of $\delta\phi$ greater than 1 improves the clustering quality by allowing the modes to change from one iteration to another [2]. The MULIC complexity is comparable to that of k-Modes of $O(mNkt)$, where t is the number of iterations [18].

8. CONCLUSION

We have presented the CEED framework for categorical clustering. CEED produces good clustering quality on small-dimensional data sets. CEED offers a probabilistic basis for developing faster clustering algorithms. MULIC is an approximation of CEED, designed for categorical data sets that have a multi-layered structure. MULIC balances clustering accuracy with time efficiency. It performs better than traditional algorithms on special data sets and has comparable runtimes. Besides protein interaction data, MULIC is also applicable to clustering large software systems and other categorical data. MULIC provides a good solution for domains where clustering primarily supports long-term strategic planning and decision making. The tradeoffs involved in approximating CEED with MULIC point us to the challenge of designing clustering algorithms that are accurate, efficient and offer good solutions to the categorical clustering problem. One direction worth pursuing as future work is to design an approximation of CEED for fuzzy clustering of categorical data sets, such that there is no sharp boundary between clusters and objects have cluster membership degrees between zero and one. Another direction worth pursuing is to develop a parallel implementation of MULIC and CEED and apply it to very large data sets.

Acknowledgements

The authors gratefully acknowledge the financial support of Natural Science and Engineering Research Council (NSERC) and Ontario Graduate Scholarship (OGS).

9. REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In Proc. ACM-SIGMOD'98, Seattle, Washington. 1998.

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

[3] P. Andritsos, P. Tsaparas, R. J. Miller, K. C. Sevcik. LIMBO: Scalable Clustering of Categorical Data. In Proc. Ninth International Conference on Extending DataBase Technology (EDBT), March 2004.

[4] M. Ankerst, M. Breunig, H.P. Kriegel, J. Sander. OPTICS: Ordering Points to Identify the Clustering Structure. In Proc. ACM SIGMOD '99 Int. Conf. on Management of Data, Philadelphia PA, 1999.

[5] G. Bader, C. Hogue. An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics, 4 (2), 2003.

[6] D. Barbara, Y. Li, J. Couto. COOLCAT: an entropy-based algorithm for categorical clustering. In Proceedings of CIKM'02, pp. 582-589, 2002.

[7] P. Berkhin. Survey of Clustering Data Mining Techniques. Accrue Software, Inc. Technical report. San Jose, CA. 2002.

[8] Z. Dezso, Z.N. Oltvai, A.L. Barabasi. Bioinformatics analysis of experimentally determined protein complexes in the yeast Saccharomyces cerevisiae. Genome Res. 13, 2450-2454 (2003).

[9] M. Ester, H.P. Kriegel, J. Sander, X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proc. of KDD 1996.

[10] Guy Even, Joseph (Seffi) Naor, Satish Rao, Baruch Schieber. Fast Approximate Graph Partitioning Algorithms. SIAM Journal on Computing. Volume 28, Number 6. pp. 2187-2214, 1999.

[11] V. Ganti, J. Gehrke, R. Ramakrishnan. CACTUS-clustering categorical data using summaries. In Proceedings of KDD '99, pp. 73-83, 1999.

[12] D. Gibson, J. Kleiberg, P. Raghavan. Clustering Categorical Data: an Approach based on Dynamical Systems. In Proc. VLDB 1998.

[13] A. Gionis, A. Hinneburg, S. Papadimitriou, P. Tsaparas. Dimension Induced Clustering, In Proc. KDD 2005.

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

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

[16] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In Proc. ACM SIGMOD, pp. 47-57, June 1984.

[17] Hinneburg A., Keim D.A. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In Proc. KDD 1998.

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

[19] L. Hubert and P. Arabie, "Comparing partitions", Journal of Classification, 193-218, 1985.

[20] D. Jiang, J. Pei, A. Zhang. DHC: a density-based hierarchical clustering method for time series gene expression data. In Proc. 3rd IEEE Symposium on Bioinformatics and Bioengineering, 2003.

[21] King, A. D., Przulj, N., Jurisica, I. Protein Complex Prediction via Cost-based Clustering. Bioinformatics, 20 (3), 340-348, 2004.

[22] R. Krauthgamer, J. R. Lee. The black-box complexity of nearest neighbor search. In Proc. ICALP, pages 858-869, 2004.

[23] H. Kriegel, B. Seeger, R. Schneider, N. Beckmann. The R*-tree: an efficient access method for geographic information systems. In Proc. International Conference on Geographic Information Systems, 1990.

[24] C.J.Mertz, P.Merphy. UCI Repository of Machine Learning Databases, 1996. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[25] H.W. Mewes, D. Frishman, U. Guldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Munsterkotter, S. Rudd and B. Weil. Mips: a database for genornes and protein sequences. Nucleic Acids Research, 30 (1),31-34, 2002.

[26] R. Mojena, Hierarchical grouping methods and stopped rules: An evaluation. The Computer Journal, vol. 20, no. 4, pp. 359-363, 1977. http://www.cs.yorku.ca/~billa/pics/mojena.bmp

[27] T. Sellis, N. Roussopoulos, C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. The VLDB Journal 1987.

[28] G. Sheikholeslami, S. Chatterjee, A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In Proc. VLDB 1998.

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

[30] C. von Mering, R. Krause, B. Snel, M. Cornell, S.G. Oliver, S. Fields, P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. Nature, 417 (6887), 399-403, 2002.

[31] Y. Yang, S. Guan, J. You. CLOPE: a fast and effective clustering algorithm for transactional data. In Proc. KDD 2002.

[32] M. Zaki, M. Peters. CLICK: Clustering Categorical Data using K-partite Maximal Cliques. TR04-11, CS Dept., RPI, 2004.

[33] Yi Zhang, Ada Wai-chee Fu, Chun Hing Cai, Pheng-Ann Heng. Clustering Categorical Data. In Proc. ICDE 2000.