Hierarchical Density-Based Clustering of Categorical Data and a Simplification

Bill Andreopoulos, Aijun An, and Xiaogang Wang

York University, Dept. of Computer Science and Engineering,
Toronto Ontario, M3J 1P3, Canada
{billa, aan}@cse.yorku.ca, stevenw@mathstat.yorku.ca

Abstract. A challenge involved in applying density-based clustering to categorical datasets is that the 'cube' of attribute values has no ordering defined. We propose the HIERDENC algorithm for *hierarchical density-based clustering of categorical data*. HIERDENC offers a basis for designing simpler clustering algorithms that balance the tradeoff of accuracy and speed. The characteristics of HIERDENC include: *(i)* it builds a hierarchy representing the underlying cluster structure of the categorical dataset, *(ii)* it minimizes the user-specified input parameters, *(iii)* it is insensitive to the order of object input, *(iv)* it can handle outliers. We evaluate HIERDENC on small-dimensional standard categorical datasets, on which it produces more accurate results than other algorithms. We present a faster simplification of HIERDENC called the MULIC algorithm. MULIC performs better than subspace clustering algorithms in terms of finding the multi-layered structure of special datasets.

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 [6,13,29]. In the past, polynomial time approximation algorithms have been designed for NP-hard partitioning algorithms [9]. 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 [12,21]. Clearly, there are tradeoffs of efficiency and approximation involved in the design of categorical clustering algorithms. Ideally, a set of probabilistically justified goals for categorical clustering would serve as a framework for approximation algorithms [20,25]. This would allow designing and comparing categorical clustering algorithms on a more formal basis.

Our work is motivated by density-based clustering algorithms, such as CLIQUE [1], CLICKS [28], CACTUS [10], COOLCAT [5], DBSCAN [8], OPTICS [4], Chameleon [19], ROCK [14], DENCLUE [15], and others. Although most of these approaches are efficient and relatively accurate, we go beyond them and approach the problem from a different viewpoint. Many of these algorithms require the user to specify input parameters (with wrong parameter

Z.-H. Zhou, H. Li, and Q. Yang (Eds.): PAKDD 2007, LNAI 4426, pp. 11–22, 2007.

values resulting in a bad clustering), may return too many clusters or too many outliers, often have difficulty finding clusters within clusters or subspace clusters, or are sensitive to the order of object input [6,12,13,28]. We propose a categorical clustering algorithm that builds a hierarchy representing a dataset's entire underlying cluster structure, minimizes user-specified parameters, and is insensitive to object ordering. This offers to a user a dataset's cluster structure as a hierarchy, which is built independently of user-specified parameters or object ordering. A user can cut its branches and study the cluster structure at different levels of granularity, detect subclusters within clusters, and know the central densest area of each cluster. Although such an algorithm is slow, it inspires faster simplifications that are useful for finding the rich cluster structure of a dataset.

A categorical dataset with m attributes is viewed as an m-dimensional 'cube', offering a spatial density basis for 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* of high object density and are separated by subspaces of low object density. Clustering the cube poses several challenges:

(i) Since there is no ordering of attribute values, the cube cells have no ordering either. The search for dense subspaces could have to consider several orderings of each dimension of the cube to identify the best clustering (unless all attributes have binary values).

(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. Although the term 'radius' is borrowed from geometrical analogies that assume circular constructs, we use the term in a looser way and it is not a Euclidean distance.

We present the *HIERDENC* algorithm for hierarchical density-based clustering of categorical data, that addresses the above challenges. HIERDENC clusters the m-dimensional *cube* representing the spatial density of a set of objects with m categorical attributes. To find its dense subspaces, HIERDENC considers an object's neighbors to be all objects that are within a *radius* of *maximum dissimilarity*. Object neighborhoods are insensitive to attribute or value ordering. Clusters start from the densest subspaces of the cube. Clusters expand outwards from a dense subspace, by connecting nearby dense subspaces. Figure 1 shows examples of creating and expanding clusters in a 3-d dataset. The radius is the maximum number of dimensions by which neighbors can differ.

We present the *MULIC algorithm*, which is a faster simplification of HIERDENC. MULIC is motivated by clustering of categorical datasets that have a *multi-layered* structure. For instance, in protein interaction data a cluster often has a center of proteins with similar interaction sets surrounded by peripheries of

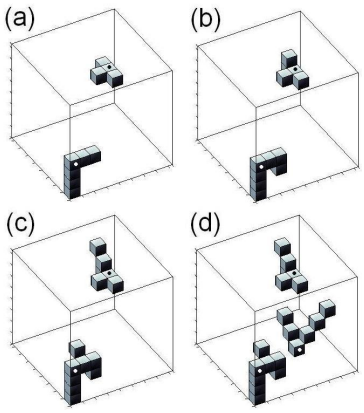

Fig. 1. A cluster is a dense subspace with a 'central' cell marked with a dot. (*a*) radius=1, two new clusters. (*b*) radius=1, clusters expand. (*c*) radius=2, clusters expand. (*d*) radius=2, one new cluster.

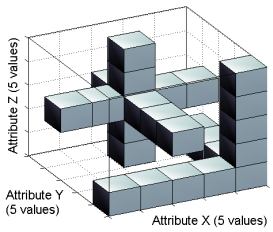

Fig. 2. Two HIERDENC 'hyper-cubes' in a 3D cube, for r=1

proteins with less similar interaction sets [7]. On such data, MULIC outperforms other algorithms that create a flat clustering.

This paper is organized as follows. Section 2 presents the HIERDENC algorithm. Section 3 describes the MULIC clustering algorithm and its relation to HIERDENC. Section 4 discusses the experiments. Section 5 concludes the paper.

2 HIERDENC Clustering

Basics. We are given a dataset 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 space 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 HIERDENC *hyper-cube* $C(\mathbf{x_0}, r) \subset S^m$, centered at cell $\mathbf{x_0}$ with radius r, as follows:

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

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

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

HD is viewed as the most natural way to represent distance in a categorical space. People have looked for other distance measures but HD has been widely accepted for categorical data and is commonly used in coding theory.

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

The *density* of a subspace $X \subset S^m$, where X could equal a hyper-cube $C(\mathbf{x_0}, r) \subset S^m$, involves the sum of λ evaluated over all cells of X:

$$density(X) = \sum_{\mathbf{c} \in X} \frac{\lambda(\mathbf{c})}{|S|}.$$

This density can also be viewed as the likelihood that a hyper-cube contains a random object from S, where $|S|$ is the size of S. HIERDENC seeks the densest hyper-cube $C(\mathbf{x_0}, r) \subset S^m$. This is the hyper-cube centered at $\mathbf{x_0}$ that 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:

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

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 $C_1, \cdots, C_n$, $A = C_1$ and $B = C_n$, such that C_i and C_{i+1} are directly connected relative to r for all i such that $1 \leq i < n$.

HIERDENC Algorithm and Discussion. Figure 3 shows the HIERDENC clustering algorithm. The default initial value of radius r 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.

Step 1 retrieves the *densest* hyper-cube $C \subset S^m$ of radius r. Step 1 checks that the densest 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 $leaf$ cluster at level $r \geq 1$. Starting from an existing leaf cluster, *step 3* tries to move to the densest 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

Input: space 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 $\max_{\mathbf{x_0}} density(C(\mathbf{x_0}, r))$.

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: Find $\mathbf{x}^* \in C(\mathbf{x_c}, r)$ such that $\mathbf{x}^* \notin U$ and $\max_{\mathbf{x}^*} density(C(\mathbf{x}^*, r))$.

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 $density(R) > 1\%$, then go to Step 1.

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

Fig. 3. The HIERDENC algorithm

can be made. New objects are clustered until $r = m$, or $density(R) \leq 1\%$ and the unclustered cells are identified as outliers (*step 5*). For many datasets, most objects are likely to be clustered long before $r = m$.

Initially $r = 1$ by default, since most datasets 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. Although the initial $r = 1$ value may result in many clusters, similar clusters are

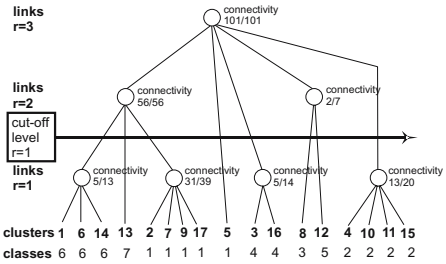

Fig. 4. The HIERDENC tree resulting from clustering the *zoo* dataset. A link (circle) represents two or more merged clusters.

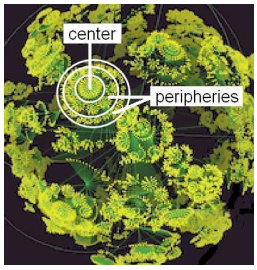

Fig. 5. A cluster has a center surrounded by peripheral areas (CAIDA)

merged gradually. As Figure 4 shows, a merge is represented as a *link* between two or more links or $leaf$ clusters, created at a level $r \geq 1$. A link represents a group of merged clusters. This process gradually constructs one or more cluster tree structures, resembling hierarchical clustering [18,24]. The user specifies a cut-off level (e.g. $r = 3$) to cut tree branches; links at the cut-off level are extracted as merged clusters. *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, an entire tree is built. At the top of the tree, there is a single cluster containing all objects of the dataset.

In [3] we propose and evaluate several methods for setting the HIERDENC tree cut-off level. One method involves cutting the HIERDENC tree at level r such that the average connectivity of the resulting merged clusters is minimized. The $connectivity_r$ of a merged cluster (a set of connected leaf clusters) relative to r is the fraction of its objects that have another object within distance r in a different leaf cluster in the same connected set. Another method useful for finding clusters within clusters is to set the cut-off(s) for a branch of links from leafs to root at the *level(s)* $r \geq 1$ such that the resulting merged cluster has $0.0 < connectivity_r < 1.0$. Another method is to balance the number of clusters with the entropy of the partition [22]. This involves setting the cut-off at level r such that the *Akaike's Information Criterion (AIC)* is minimized [2]. The AIC of a partition is $entropy + 2k$, where k is the number of clusters.

Although HIERDENC has similarities to CLIQUE [1], the two have significant differences. HIERDENC is intended for categorical data while CLIQUE for numerical data. HIERDENC minimizes input parameters, while CLIQUE takes as input parameters the grid size and a global density threshold for clusters. HIERDENC retrieves the densest hyper-cube relative to the radius. The radius relaxes gradually, implying that HIERDENC can find clusters of different densities. HIERDENC can often distinguish the central hyper-cube of a cluster from the rest of the cluster, because of its higher density. HIERDENC creates a tree representing the entire dataset structure, including subclusters within clusters.

3 MULIC as a Simplification of HIERDENC

MULIC stands for *multiple layer clustering* of categorical data. MULIC is a faster simplification of HIERDENC. MULIC balances clustering accuracy with time efficiency. The MULIC algorithm is motivated by datasets the cluster structure of which can be visualized as shown in Figure 5. In such datasets a cluster often has a center of objects that are similar to one another, along with peripheral objects that are less similar to the central objects. Such datasets include protein interaction data, large software systems and others [7].

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 does not search all member objects as HIERDENC does. Instead, it uses a mode that summarizes the content of a cluster. 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 [16]. 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 2.1, although any metric could be used. A MULIC cluster consists of *layers* formed gradually, by relaxing the maximum dissimilarity criterion ϕ for inserting objects into existing clusters. MULIC does not require the user to specify the number of clusters and can identify outliers. Figure 6 shows the main part of the MULIC clustering algorithm. An optional final step merges similar clusters to reduce the number of clusters and find more interesting structures.

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 layer 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. The default merging process, detailed in [3], merges clusters in a non-hierarchical manner such that clusters have a clear separation. However, a hierarchical cluster merging process is also proposed [3].

MULIC Discussion. MULIC is a simplification of HIERDENC. The tradeoffs between accuracy and time efficiency are as follows:

(i) When creating a cluster, HIERDENC searches the cube to retrieve the densest 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 HIERDENC's radius r. The initial objects clustered with MULIC affect the modes and the clustering. For this issue we propose in [3] an optional preprocessing step that orders the objects by decreasing aggregated frequency of their attribute values, such that objects with more frequent values are clustered first and the modes will likely

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. Order objects by decreasing aggregated frequency of their attribute values.
2. Insert the first object into a new cluster, use the object as the mode of the cluster, and remove the object from S.
3. Initialize ϕ to 1.
4. Loop through the following until S is empty or $\phi > threshold$
  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 $\delta\phi$.

Fig. 6. The MULIC clustering algorithm

contain the most frequent values. This object ordering process has been evaluated in [3], which showed that it is better than a random ordering of objects; we do not include the same results here.

(ii) When expanding a cluster HIERDENC 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 from the mode. MULIC increases ϕ by $\delta\phi$ when no new objects can be clustered, which is motivated by HIERDENC's increasing r. MULIC can create new clusters at any value of ϕ, just as HIERDENC can create new clusters at any value of r. Although MULIC can find clusters of arbitrary shapes by increasing ϕ, it loses some of HIERDENC's ability in this realm.

(iii) MULIC's cluster merging is motivated by HIERDENC's merging. The MULIC cluster merging process can organize clusters into a tree structure as HIERDENC does. For MULIC applications, such as the one on protein interaction data discussed in [3], we do not construct a tree since we prefer the clusters to have a clear separation and not to specify a cut-off.

MULIC has several differences from traditional hierarchical clustering, which stores all distances in an upper square matrix and updates the distances after

each merge [18,24]. MULIC clusters have a clear separation. MULIC does not require a cut-off to extract the clusters, as in hierarchical clustering; this is of benefit for some MULIC applications, such as the one on protein interaction data discussed in [3]. One of the drawbacks of hierarchical clustering is that the sequence of cluster mergings will affect the result and 'bad' mergings can not be undone later on in the process. Moreover, if several large clusters are merged then interesting local cluster structure is likely to be lost. MULIC, on the other hand, does not merge clusters during the object clustering. Instead, any cluster mergings that may be desirable for the particular application are done after object clustering has finished. MULIC aims not to lose cluster structure caused by several large clusters being merged during the clustering process.

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 $O(mN^2\frac{threshold}{\delta\phi})$. The cost is related to the number of clusters k and the number of objects N. Often $k \ll N$, $m \ll N$, and all objects are clustered in the initial iterations, thus N often dominates the cost. The worst-case runtime would occur for the rather uncommon dataset 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 $\phi = m$. The MULIC complexity is comparable to that of k-Modes of $O(mNkt)$, where t is the number of iterations [16].

4 Performance Evaluation

To evaluate the applicability of HIERDENC and MULIC to the clustering problem, we first use the *zoo* and *soybean-data* categorical datasets. These datasets were obtained from the UCI Repository [23]. Objects have class labels defined based on some domain knowledge. We ignore class labels during clustering. We compare the HIERDENC and MULIC results to those of several other density-based algorithms, ROCK [14], CLICKS [28], k-Modes [16], and AutoClass [26]. CLICKS was shown to outperform STIRR [11] and CACTUS [10]. To evaluate the clustering quality we use *HA Indexes* [17] and *Akaike's Information Criterion (AIC)* [2]. 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 estimate the AIC of each clustering. AIC penalizes non-uniformity of attribute values in each cluster and too many clusters. In [3] we discuss MULIC with non-hierarchical and hierarchical merging of clusters applied to protein interaction data and large software systems.

For MULIC we set $\delta\phi = 1$, $threshold = m$, and we order the objects as described in [3]. We applied the other algorithms (except HIERDENC) on more than 10 random orderings of the objects. For k-Modes and ROCK we set the number of clusters k to the number of classes, as well as larger numbers. AutoClass considers varying numbers of clusters from a minimum of 2.

Table 1. HA Indexes (higher is better), Entropy, and AIC measures (lower is better)

	zoo (7 classes)					**soybean-data (19 classes)**				
Tool	*HAI.*	*Entr.*	*AIC*	k	*sec*	*HAI.*	*Entr.*	*AIC*	k	*sec*
HIERDENC (leaf clusters)	85.5%	2.15	36.15	17	0.04	92.2%	4.4	182.4	89	2.1
HIERDENC (after tree cut)	94%	2.3	18.3	8	0.04	95%	7.5	47.5	20	2.1
MULIC (no merging)	84%	2.5	40.5	19	0	92%	4.6	182.6	89	0.05
MULIC (after merging)	91.5%	2.8	22.8	10	0.03	93%	11.5	61.5	25	0.08
k-Modes	90%	3.5	23.5	10	0.005	80%	16.12	66.12	25	0.03
ROCK	73%	3.7	23.7	10	0.008	69.2%	19.5	69.5	25	0.04
AutoClass	79.5%	4.8	16.8	6	0.04	77.6%	25	39	7	0.13
CLICKS	91.5%	2.5	20.5	9	0.01	70%	10	90	40	1
Chameleon (wCluto part.)	72%	3.8	23.8	10	0	79%	16.5	66.5	25	0.1

HIERDENC Results. Table 1 shows the HIERDENC results for these datasets before and after cutting the tree. After cutting the HIERDENC tree for *zoo*, its HA Indexes, Entropy, and AIC are slightly better than CLICKS. The HIERDENC results for *soybean-data* are significantly better than CLICKS. The Entropy is naturally lower (better) in results with many clusters; by comparing results of algorithms with similar numbers of clusters, the HIERDENC Entropy is often lower. The drawback we notice is that the HIERDENC runtime is significantly higher on *soybean-data* than on *zoo*.

Figure 4 illustrates the HIERDENC tree for *zoo*. There are 17 leaf clusters in total in the HIERDENC tree. Except for the last 3 created leaf clusters, all other leaf clusters are homogeneous with regards to the class labels of member objects. The last 3 leaf clusters were created for high r values of 7, 6, and 4. The rest of the leaf clusters were created for lower r values. For *zoo* we cut off the HIERDENC tree at level $r = 1$; *zoo* is a rather dense cube with many nonzero cells and we do not want to aggregate information in the cube. The $r = 1$ cut-off minimizes the connectivity relative to r of the resulting clusters. By cutting the HIERDENC *zoo* tree at $r = 1$, there are 8 resulting clusters. There are a few cases of incorrectly clustered objects by cutting at $r = 1$. However, the lower number of clusters results in improved HA Indexes.

For the *soybean-data* set, there are 89 leaf clusters in total in the HIERDENC tree. The leaf clusters created for $r \leq 9$ are homogeneous with regards to the class labels of member objects. For leaf clusters created for $r > 9$, the homogeneity of the class labels decreases. Only 23 objects are clustered for $r > 9$, so these could be labeled as outliers. For *soybean-data* we cut off the HIERDENC tree at $r = 4$; *soybean-data* is a sparse cube of mostly '0' cells, since the dataset has 35 dimensions but only 307 objects. The $r = 4$ cut-off minimizes the connectivity relative to r of the resulting clusters. By cutting the HIERDENC *soybean-data* tree at $r = 4$, there are 20 resulting merged clusters.

MULIC Results. Table 1 shows the MULIC results for these datasets with and without merging of clusters. 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 similar animals; for example, the animals 'porpoise', 'dolphin', 'sealion', and 'seal' are clustered together in one MULIC cluster. MULIC with non-hierarchical merging of clusters has as a result that the number of clusters decreases, which often improves the quality of the result according to the HA Indexes. After merging the MULIC clusters, the number of clusters for *zoo* and *soybean-data* is close to the class-label-based number of classes. After merging the MULIC clusters for *zoo*, the HA Indexes, Entropy, and AIC are as good as CLICKS. The MULIC results for *soybean-data* are better than CLICKS. The Entropy is naturally lower (better) in results with many clusters; by comparing results of algorithms with similar numbers of clusters, the MULIC Entropy is often lower. MULIC runtimes are lower than HIERDENC.

5 Conclusion

We have presented the HIERDENC algorithm for categorical clustering. In HIERDENC a central subspace often has a higher density and the radius relaxes gradually. HIERDENC produces good clustering quality on small-dimensional datasets. HIERDENC motivates developing faster clustering algorithms.

MULIC balances clustering accuracy with time efficiency. MULIC provides a good solution for domains where clustering primarily supports long-term strategic planning and decision making, such as analyzing protein-protein interaction networks or large software systems [3]. The tradeoffs involved in simplifying HIERDENC with MULIC point us to the challenge of designing categorical clustering algorithms that are accurate and efficient.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. SIGMOD 1998
2. H. Akaike. A new look at the statistical model identification. IEEE TAC, 19, 716-23, 1974
3. B. Andreopoulos. Clustering Algorithms for Categorical Data. PhD Thesis, Dept of Computer Science & Engineering, York University, Toronto, Canada, 2006
4. M. Ankerst, M. Breunig, H.P. Kriegel, J. Sander. OPTICS: Ordering Points to Identify the Clustering Structure. SIGMOD 1999
5. D. Barbara, Y. Li, J. Couto. COOLCAT:an entropy-based algorithm for categorical clustering. CIKM 2002
6. P. Berkhin. Survey of Clustering Data Mining Techniques. Accrue Software, Inc. TR, San Jose, USA, 2002
7. Z. Dezso, Z.N. Oltvai, A.L. Barabasi. Bioinformatics analysis of experimentally determined protein complexes in the yeast Saccharomyces cerevisiae. Genome Res. 13, 2450-4, 2003
8. M. Ester, H.P. Kriegel, J. Sander, X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD 1996
9. G. Even, J. Naor, S. Rao, B. Schieber. Fast Approximate Graph Partitioning Algorithms. SIAM Journal on Computing, 28(6):2187-2214, 1999

10. V. Ganti, J. Gehrke, R. Ramakrishnan. CACTUS-clustering categorical data using summaries. KDD 1999
11. D. Gibson, J. Kleiberg, P. Raghavan. Clustering Categorical Data: an Approach based on Dynamical Systems. VLDB 1998
12. A. Gionis, A. Hinneburg, S. Papadimitriou, P. Tsaparas. Dimension Induced Clustering. KDD 2005
13. J. Grambeier, A. Rudolph. Techniques of Cluster Algorithms in Data Mining. Data Mining and Knowledge Discovery 6: 303-360, 2002
14. S. Guha, R. Rastogi, K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Systems 25(5): 345-366, 2000
15. A. Hinneburg, D.A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD 1998
16. Z. Huang. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining & Knowledge Disc. 2(3): 283-304, 1998
17. L.Hubert & P.Arabie. Comparing partitions. J.Classification 193-218, 1985
18. D. Jiang, J. Pei, A. Zhang. DHC: a density-based hierarchical clustering method for time series gene expression data. IEEE Symp. on Bioinf. and Bioeng., 2003
19. G. Karypis, E.H. Han, V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. IEEE Computer 32(8): 68-75, 1999
20. J. Kleinberg, C. Papadimitriou, P. Raghavan. Segmentation Problems. STOC 1998
21. R. Krauthgamer, J.R. Lee. The black-box complexity of nearest neighbor search. ICALP 2004
22. T. Li, S. Ma, M. Ogihara. Entropy-Based Criterion in Categorical Clustering. ICML 2004
23. C.J. Mertz, P. Merphy. UCI Repository of Machine Learning Databases, 1996
24. R. Mojena, Hierarchical grouping methods and stopped rules: An evaluation. The Computer Journal, 20(4), 359-63, 1977
25. C.Papadimitriou. Algorithms, Games, and the Internet. STOC 2001
26. J. Stutz and P. Cheeseman. Bayesian Classification (AutoClass): Theory and results. Advances in Knowledge Discovery & Data Mining, 153-180, 1995
27. Y. Yang, S. Guan, J. You. CLOPE: a fast and effective clustering algorithm for transactional data. KDD 2002
28. M. Zaki, M. Peters. CLICK: Clustering Categorical Data using K-partite Maximal Cliques. TR04-11, Rensselaer Polytechnic Institute, 2004
29. Y. Zhang, A.W. Fu, C.H. Cai, P.A. Heng. Clustering Categorical Data. ICDE 2000