Efficient Layered Density-based Clustering of Categorical Data

Bill Andreopoulos[ab] and Aijun An[b] and Xiaogang Wang[c] and Dirk Labudde[a]

[a] *Biotechnological Centre, Technische Universität Dresden, Germany*
[b] *Dept. of Computer Science and Engineering, York University, Toronto, Canada*
[c] *Dept. of Mathematics and Statistics, York University, Toronto, Canada*

Preprint submitted to Elsevier *2 December 2008*

Abstract

A challenge involved in applying density-based clustering to categorical biomedical data is that the "cube" of attribute values has no ordering defined, making the search for dense subspaces slow. We propose the HIERDENC algorithm for *hierarchical density-based clustering of categorical data*, and a complementary index for searching for dense subspaces efficiently. The HIERDENC index is updated when new objects are introduced, such that clustering does not need to be repeated on all objects. The updating and cluster retrieval are efficient. Comparisons with several other clustering algorithms showed that on large datasets HIERDENC achieved better runtime scalability on the number of objects, as well as cluster quality. By fast collapsing the bicliques in large networks we achieved an edge reduction of as much as 86.5%. HIERDENC is suitable for large and quickly growing datasets, since it is independent of object ordering, does not require re-clustering when new data emerges, and requires no user-specified input parameters.

Key words:
Clustering, Bioinformatics, Categorical, Network, Index Scalable, Layered

Email address: williama@biotec.tu-dresden.de (Bill Andreopoulos[ab] and Aijun An[b] and Xiaogang Wang[c] and Dirk Labudde[a]).

1 Introduction

Categorical datasets are frequently clustered in biomedical informatics. Applications range from health information records to protein-protein interaction and sequence similarity networks. Layered categorical clustering, where a cluster consists of a center of similar objects and outer layers of less similar objects, has acquired prominence. Layered clusters are useful in bioinformatics for finding protein modules, complexes, and for visualization purposes (1; 2; 3; 4; 5). However, often, the focus is on the quality of the clusters, with a secondary priority placed on the speed of the method, scalability to large datasets, and its usability.

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 (6). Clusters in such a cube are regarded as *subspaces* of high object density and are separated by subspaces of low object density (7). *Density-based* clustering algorithms, such as DBSCAN (8) or OPTICS (9), search for dense subspaces. A dense subspace is defined by a *radius* of maximum distance from a central point, and it has to contain many objects according to a threshold criterion (10). With the radius gradually increasing to allow more objects in clusters, *layered clusters* result. Our goal is to tackle some of the general challenges of existing clustering approaches:

(i) The density of a subspace is often defined relative to a user-specified *radius* [1]. However, different radii are preferable for different subspaces of the cube (9). 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.

(ii) The time requirement is often a problem in density-based clustering, since it may be too slow to find the densest subspace in a high-dimensional dataset, and the dataset may change often. In particular, 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 (11; 12; 13).

(iii) Other challenges include: re-clustering needed when new objects are introduced, difficulty finding clusters within clusters, sensitivity to order of object input, or user-specified input parameters required with wrong values affecting the end result (14; 15; 16; 17).

We present the *HIERDENC* algorithm for "hierarchical density-based cluster-

[1] 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.

ing of categorical data", which 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. Figure 1 shows that the radius is the maximum number of dimensions by which neighbors can differ. The cube search starts from a low radius and gradually moves to higher radii. With the clustering radius gradually increasing, *layered clusters* result, as Figure 2 shows. Figure 3 shows examples of creating and expanding clusters in a 3-dimensional dataset.

For scalability to large categorical datasets, we propose the *HIERDENC index*, which supports efficient retrieval of dense subspaces relative to a radius. When new objects are introduced, HIERDENC is updated efficiently. The neighborhood of an object is insensitive to attribute or value ordering. A user can study the layered cluster structure at different levels of granularity, detect subclusters within clusters, and know the central densest area of each cluster.

Applications of HIERDENC to biomedical informatics abound. One application is clustering networks to find bicliques. A network's adjacency matrix is a boolean-valued categorical dataset, where rows and columns represent objects and '1' is a connection; a biclique is a network whose objects can be divided into two disjoint sets, such that every object of the first set is connected to every object of the second set. Finding bicliques in a network has several applications to biological problems; in protein-protein interaction networks the bicliques can be visualized, or correlated with structural knowledge to find the structures that induce observed interactions (1; 5). Furthermore, HIERDENC is applicable to biomedical images and literature; we demonstrate a fast image retrieval system and a PubMed document clustering that we built. We also applied HIERDENC to clustering of Force-Distance curves from high-throughput proteomic studies, by aligning curves on the basis of their detected peaks to one another.

This paper is organized as follows. Section 2 gives an overview of related work on dissimilarity metrics and density-based clustering for categorical data. Sections 3 and 4 present the HIERDENC clustering algorithm and index. Section 5 discusses our performance evaluations. Applications to categorical datasets show runtime scalability and clustering quality. In section 6 we discuss biomedical applications. We apply HIERDENC to large networks, such that bicliques are collapsed, comfirming the runtime scalability. Section 7 concludes that our method amends some of the weaknesses of previous categorical density-based clustering approaches, and has promising utilities in biomedical informatics.

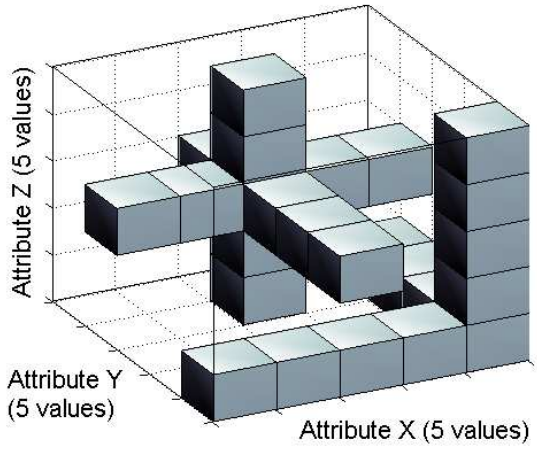


Fig. 1. Two HIERDENC 'hyper-cubes' for radius r=1, in a 3D cube. All neighbors of the central object for each hyper-cube differ from it in one dimension.

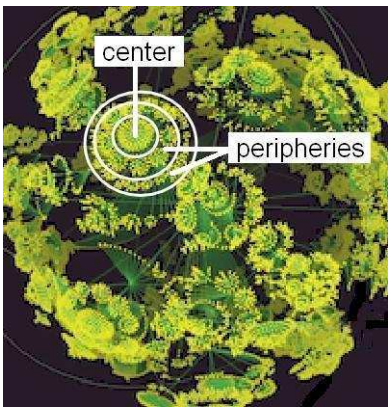


Fig. 2. A layered network cluster has a center surrounded by outer layers.

2 Background and Related Work

Section 2.1 describes the Hamming Distance, and section 2.2 provides an overview of density-based clustering algorithms for categorical data.

2.1 The Hamming Distance in Categorical and Binary Data

For a fixed length m, the Hamming distance is a metric on the vector space of the words of that length. Figure 4 shows an example of HDs in the *zoo* dataset (18). The serpent *tuatara* is within a relatively small HD from the other serpents; the maximum distance is $HD(tuatara \leftrightarrow seasnake) = 5$. On the other hand, $HD(tuatara \leftrightarrow gorilla) = 8$, and gorilla is unlikely to belong to the class of serpents. For binary strings a and b the HD is equivalent to the number of ones in a *xor* b. The metric space of length-m binary strings, with

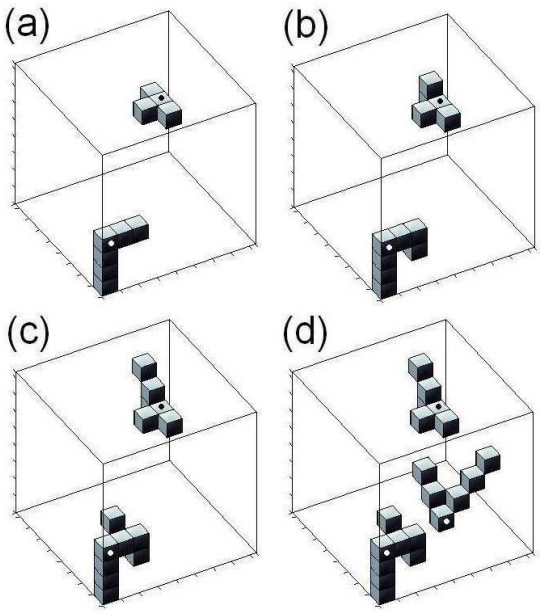


Fig. 3. A cluster is a dense subspace with a 'central' cell marked with a dot. The radius starts from 1 and changes when neither a cluster can be expanded, nor a new cluster can be formed. (a) radius=1, two new clusters. (b) radius=1, clusters expand. The radius did not change since a, but fewer objects are found within a radius of 1 than in a, implying a less dense subspace. (c) radius=2, clusters expand. (d) radius=2, one new cluster.

the HD, is known as the Hamming cube.

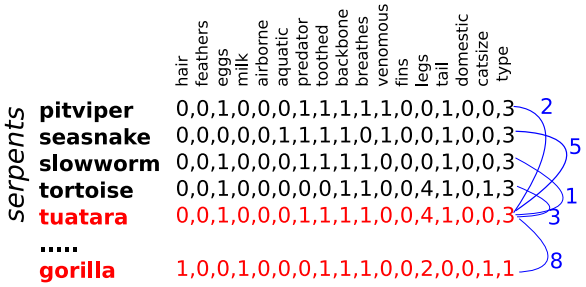


Fig. 4. Example of Hamming distances on the *zoo* categorical dataset.

2.2 Density-based Clustering

General desirable features of clustering algorithms include: a. Linear scalability on the size of the dataset, b. Insensitivity to object ordering, c. No re-clustering needed when new objects are introduced, and d. No user-specified input parameters required. Several density-based clustering algorithms for categorical data have been proposed.

Projected (or subspace) clustering is motivated by high-dimensional feature

spaces, in which many algorithms tend to break down since clusters often exist only in specific attribute subsets of the original space (19). Projected clustering methods are density-based algorithms that form clusters in subspaces in high-dimensional datasets, by finding the clusters' relevant attributes (20). For each cluster, projected clustering determines a set of "relevant attributes". The rationale of projected clustering is that only a subset of attributes is relevant to each cluster and each cluster can have a different set of relevant attributes. An attribute is relevant to a cluster if it helps identify its member objects. This means the values at the relevant attributes are distributed around some specific values in the cluster, while the objects of other clusters are less likely to have such values. The drawback is that clustering depends on some user parameters for determining the relevant attributes of each cluster; such parameters are the number of clusters or the average number of dimensions for each cluster (14; 16; 21; 22). Projected clustering may distinguish the center of a cluster based on higher density or the relevant attributes (23). PROCLUS is a well-known projected clustering algorithm (24).

CACTUS uses a minimum size for the relevant attribute sets, and assumes that a cluster is identified by a unique set of attribute values that seldom occur in other clusters. The relevant attributes are extended to candidate cluster projections (25). Many real world datasets might not have one minimum size of relevant attribute sets applicable for all clusters (17).

STIRR looks for relationships between all attribute values in a cluster (26). Two sets of attribute values, one with positive and another with negative weights, define two clusters. STIRR is sensitive to the input ordering and lacks a definite convergence. The notion of weights is non-intuitive and several operators are left to the user to define. The final detected clusters are often incomplete (17; 25).

CLICKS creates a graph representation; vertices are categorical values and an edge is a co-occurrence of values in an object. A cluster is a k-partite maximal clique such that most pairs of vertices are connected by an edge (17). A merging process is proposed to reduce the number of clusters or outliers.

CLOPE for categorical and transactional data uses a heuristic method of increasing the height-to-width ratio of the cluster histogram (27). CLOPE's advantages include fast performance and scalability to large data sets with high dimensions.

ROCK is an adaptation of a hierarchical clustering algorithm for categorical data. It does not require the user to specify the number of clusters. Initially, each tuple is assigned to a separate cluster and then clusters are merged repeatedly according to the closeness between clusters. The algorithm exhibits cubic complexity in the number of objects, which makes it unsuitable for large

data sets (11; 14; 16; 17).

DBSCAN regards clusters as dense regions of objects in space that are separated by regions of low density. 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. Every object not contained in any cluster is considered noise (28). The computational complexity of DBSCAN is $O(NlogN)$. The main advantage of DBSCAN is that it can discover clusters of arbitrary shape. DBSCAN is resistant to noise and provides a means of filtering for noise if desired. The main drawback of DBSCAN is that the user needs to specify parameter values, such as radius, that will affect the result (21) DBSCAN is not suitable for high-dimensional data; as dimensionality increases, so does the relative distance between points making it harder to perform density analysis (20).

OPTICS considers that different parts of space could require different parameters. OPTICS covers a spectrum of all different $\epsilon' \leq \epsilon$. OPTICS has the same complexity as DBSCAN, $O(NlogN)$. OPTICS finds an ordering of data that is consistent with DBSCAN (9). For sequence clustering, OPTICS was extended into SEQOPTICS, to support users choosing parameters (29). DBSCAN and OPTICS have difficulty identifying clusters within clusters (15; 16).

3 Categorical Data Clustering with HIERDENC

Section 3.1 presents the basic concepts. Section 3.2 describes the HIERDENC clustering algorithm. Clusters start from the densest subspaces of the cube, and expand by connecting nearby dense subspaces.

3.1 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 1 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 $1 \leq i < n$.

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. 5. The HIERDENC clustering algorithm.

3.2 HIERDENC Clustering Algorithm and Discussion

Figure 5 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; to achieve this fast, we use the index that is described in the next Section. 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 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, an *optional* special process *merges* clusters that are connected relative to r [2]. Although the initial $r = 1$ value may result in many clusters, similar clusters can be merged gradually. As Figure 6 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 (30; 31). 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 (2) we propose and evaluate several methods for setting the HIERDENC tree cut-off level (32). One method involves cutting the HIERDENC tree at level r that minimises the average connectivity of the resulting merged clusters. The $connectivity_r$ of a merged cluster (a set of connected leaf clusters) relative

[2] For network clustering no tree structure is built. Also in the biomedical image, PubMed clustering and curve alignment applications no parameters nor tree is used. Clusters are produced without a tree, but the hierarchical organization of clusters into a tree is still available as an option

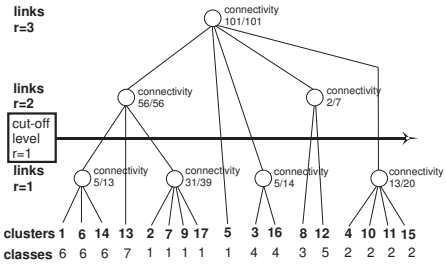


Fig. 6. The HIERDENC tree resulting from clustering the *zoo* dataset. A link (circle) represents two or more merged clusters.

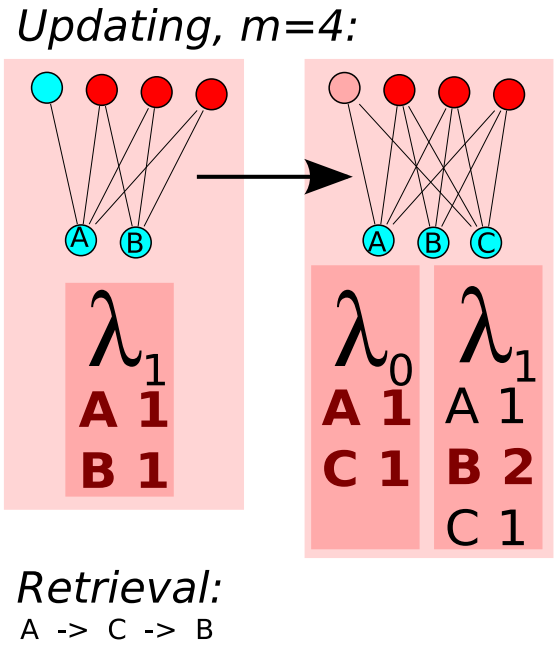


Fig. 7. HIERDENC updating and retrieval of the densest subspaces. As objects A, B, C are introduced, the lists λ_0 and λ_1 are updated accordingly. During retrieval, the bold entries in λ_0 and λ_1 are traversed in order.

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 is to balance the number of clusters with the entropy of the partition (33). This involves setting the cut-off at level r that minimises the *Akaike's Information Criterion (AIC)* (34). The AIC of a partition is $entropy + 2k$, where k is the number of clusters. Another method useful for finding clusters within clusters, or nested clusters, is to cut-off a tree's branches at various levels (31).

4 Efficient retrieval and updating of dense categorical subspaces

In this section we describe the index used in Step 1 of the HIERDENC algorithm for efficient retrieval of the *densest* subspace $C(\mathbf{x_0}, r) \subset S^m$ of radius r. Figure 7 shows the updating and retrieval steps involved.

We conceptualize a categorical dataset as a graph consisting of $m+1$ sets of nodes, $\{S_0 \cdots S_m\}$. S_0 represents the N objects in the dataset. Sets $\{S_1 \cdots S_m\}$ represent the m categorical attributes and their members are the attribute values. An object $\mathbf{x}$ is connected via an edge to a node in each of the sets $S_1 \cdots S_m$, representing $\mathbf{x}$'s value for the corresponding attribute. The dissimilarity of two objects $\mathbf{x}_\alpha, \mathbf{x}_\beta \in S_0$, $\mathbf{x}_\alpha \neq \mathbf{x}_\beta$, is $0 \leq HD(\mathbf{x}_\alpha, \mathbf{x}_\beta) \leq m$, and represents the number of sets $\{S_1 \cdots S_m\}$ in which $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ have dissimilar attribute values.

For object $\mathbf{x_c}$ and radius r, if we can retrieve in constant time all objects $\{\mathbf{x}|HD(\mathbf{x}, \mathbf{x_c}) = r\}$, then we can retrieve the densest subspace relative to r in time that is linear to the number of objects N. This could naively be done by iterating through all objects in the dataset and keeping the object $\mathbf{x_0}$ that maximizes $|\{\mathbf{x}|HD(\mathbf{x}, \mathbf{x_0}) = r\}|$. If we maintain for each r a list of objects ordered by the sizes of their r-neighborhoods, then the runtime becomes faster at the expense of updating the list.

Given this representation of categorical data as graphs, we proceed to describe the HIERDENC index. Our goal is fast retrieval of the object $\in S_0$ that has the most other objects $\in S_0$ with a dissimilarity of r. Given an object $\mathbf{x} \in S_0$, $\psi_{\mathbf{x}r}$ is the number of other objects that have dissimilarity to $\mathbf{x}$ of r. We call $\psi_{\mathbf{x}r}$ the *density* of subspace $C(\mathbf{x}, r) \subset S^m$ that is centered at $\mathbf{x}$ relative to r.

We maintain, for each $0 \leq r < m$, a list λ_r of ranked objects $\in S_0$ from highest to lowest according to their subspace density $\psi_{\mathbf{x}r}$. Let $\lambda_r(\mathbf{x})$ denote the rank of object $\mathbf{x}$ in the list λ_r. For two objects $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$, if $\psi_{\mathbf{x}_\alpha r} > \psi_{\mathbf{x}_\beta r} > 0$ then $\lambda_r(\mathbf{x}_\alpha) > \lambda_r(\mathbf{x}_\beta)$.

Updating the HIERDENC index

Updating the index when a new edge is added between $\mathbf{x}_\alpha \in S_0$ and $\alpha \in \{S_1 \cdots S_m\}$ involves the following steps:

(1) S_t = objects $\in S_0$ that are connected to α.
(2) For $\{\mathbf{x} \in S_t | \mathbf{x} \neq \mathbf{x}_\alpha\}$:
(3) Decrement $\delta = HD(\mathbf{x}, \mathbf{x}_\alpha)$.
(4) Increment $\lambda_\delta(\mathbf{x})$ and $\lambda_\delta(\mathbf{x}_\alpha)$.
(5) Decrement $\lambda_{\delta+1}(\mathbf{x})$ and $\lambda_{\delta+1}(\mathbf{x}_\alpha)$.

Retrieving the densest subspaces in order

Retrieval involves the following steps:

(1) S_t = objects in S_0 that have been returned; initially null.
(2) For $r \in range(0, \cdots, m-1)$:
(3) For $\mathbf{x_0} \in \lambda_r$ ordered by $\lambda_r(\mathbf{x_0})$, if $\mathbf{x_0} \notin S_t$:
(4) Return $C = \{\mathbf{x} \in S_0 | HD(\mathbf{x}, \mathbf{x_0}) = r\}$.
(5) $S_t = S_t \cup C$.

4.1 Time and Space Complexity

The first time the HIERDENC index is populated with a dataset of N objects, the average runtime is $O(Nm)$, where m is the number of categorical attributes (usually $m << N$). For each of the N new objects the dissimilarities involving m connected nodes are updated. When n new objects are introduced, the updating of the index has a runtime of $O(nm)$.

For the densest subspaces to be retrieved, the worst-case runtime is $O(N)$; the retrieval iterates until the subspaces centered at a maximum of N objects have been retrieved. At a newly introduced object, some of the lists λ_r, $0 \leq r < m$, are updated.

4.2 Potential issues

While the runtime scales well on dataset size, the worst-case space complexity is $O(N^2)$, which involves maintaining objects' dissimilarities; an implementation may store information about all pairs of objects that exhibit the minimum dissimilarity or maximum similarity. However, $O(N^2)$ space would be needed only for a dataset where all objects were equally dissimilar to all objects. Unless it is an exceptional dataset of a special case, not all objects have the same dissimilarity (are minimally dissimilar) to all other objects; in a typical real world dataset, most object pairs are highly dissimilar, and for each object there is a subset of objects with the minimal dissimilarity. For large datasets, most objects are dissimilar enough, such that their dissimilarities are not stored and significantly less space is needed. Furthermore, a user must be careful when implementing a HIERDENC index not to store information on highly dissimilar object pairs, which is likely to denote outliers, since storing this information may hurt unnecessarily the HIERDENC index's updating performance.

Categorical	*objects*	*attributes*	*classes*	*Network*	*objects*	*edges*
zoo	101	16	7	Gavin06 (yeast)	2551	93881
soybean	376	35	19	Krogan06 (yeast)	3670	14292
car	1728	6	4	Stelzl05 (human)	1529	2668
mushroom	8124	22	2	Rual05 (human)	1874	3618
nursery	12960	8	5	Internet topology	19938	59582
				Pubmed gene co-occs	10704	53319

Table 1
The UCI datasets (categorical) and the networks (boolean).

5 Performance Evaluation

To evaluate the applicability of HIERDENC to the problem of efficient clustering, we used large networks and categorical datasets obtained from the UCI Machine Learning Repository (18). Table 1 shows details for all datasets. Objects have class labels defined based on some domain knowledge that we ignored during clustering.

5.1 Runtimes

Figures 8*a* and 9*a* show the runtime scalability across all of the datasets. Updating scales linearly with the dataset size. This highlights the utility of HIERDENC for a fastly growing database like PubMed. Retrieval of the dense subspaces also scales linearly with dataset size.

Figure 8*b* shows that the cumulative updating and retrieval runtimes for the largest categorical dataset scale with the number of objects. Figure 9*b* shows the same for the largest network. The runtimes may decrease slightly later in the process, because the retrieved subspaces are less dense and have fewer objects. The runtimes for the other datasets looked similar and are not shown here to avoid redundancy.

5.2 Performance on UCI categorical datasets

Table 2 compares the HIERDENC results and runtimes to several classical algorithms for which we possessed the source code: k-Modes (35), ROCK (11), AutoClass (36), CLOPE (27), CLICKS (17). To evaluate the clustering quality we used F1-measure, a class-label-based evaluation that penalizes more or

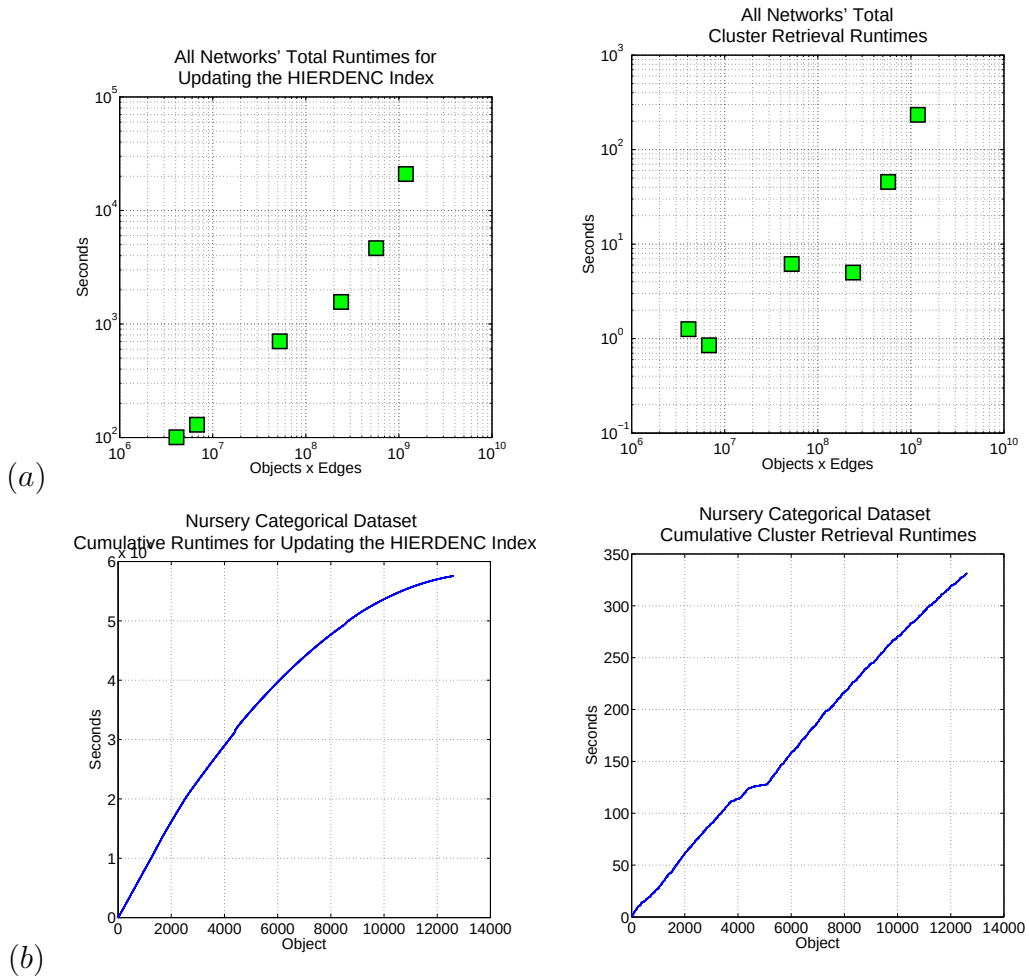


Fig. 8. Runtimes. (a) All categorical UCI datasets: total runtimes (y-axis) for updating and retrieval by dataset size (x-axis). (b) Nursery categorical dataset: cumulative runtimes (y-axis) for updating as new objects are introduced (x-axis), and retrieving the densest subspace centered at each object (x-axis).

fewer clusters than the number of classes. For k-Modes, we set the convergence *threshold* to 1 and we set the modes of the initial clusters equal to the first objects clustered. For ROCK, we set $\theta = 0.5$. For k-Modes and ROCK we set the number of clusters k to the number of classes, as well as larger numbers, and we report the best result. AutoClass considers various numbers of clusters starting from 2. For CLICKS we set $\alpha = 0.1$ and $minsup = 0.1$.

On small datasets HIERDENC took slightly more time than the other algorithms, but on the larger datasets the HIERDENC runtime was lower, highlighting its scalability. In comparison, we notice how CLICKS took significantly more time on the larger datasets. ROCK with its cubic complexity exhibited the worst runtimes. CLOPE's performance suffered on these datasets with the implementation we had available. CLICKS' clustering quality was shown to outperform previous methods like ROCK, STIRR and CACTUS (17). Note that Table 2 reports HIERDENC total clustering runtimes, and

Dataset *objects* × *attribs*	*zoo* 101 × 16			*soybean-large* 376 × 35			*car* 1728 × 6			*mushroom* 8124 × 22			*nursery* 12960 × 8		
Algorithm	*F*	*k*	*secs*	*F*	*k*	*secs*	*F*	*k*	*secs*	*F*	*k*	*secs*	*F*	*k*	*secs*
HIERDENC	96	8	0.08	97	37	0.31	90	5	0.33	100	22	1.81	90	5	1.841
k-Modes	80	7	0.01	85	20	0.03	60	7	0.1	65	20	8.69	65	20	10.6
ROCK	< 60	7	~ 10	< 60	40	~ 10	< 60	40	~ 20	< 60	40	~ 300	< 60	40	~ 600
AutoClass	< 60	2	0.04	77	3	0.17	< 60	2	2.47	< 60	7	1.29	< 60	2	4.33
CLOPE	< 60	3	0.5	< 60	15	1	< 60	23	1	< 60	10	2	< 60	15	4
CLICKS	85%	12	0.5	< 60	1	1	< 60	2	1	< 60	2	11	< 60	2	35

Table 2. F1-measures and runtimes achieved on the UCI categorical datasets for various clustering algorithms.

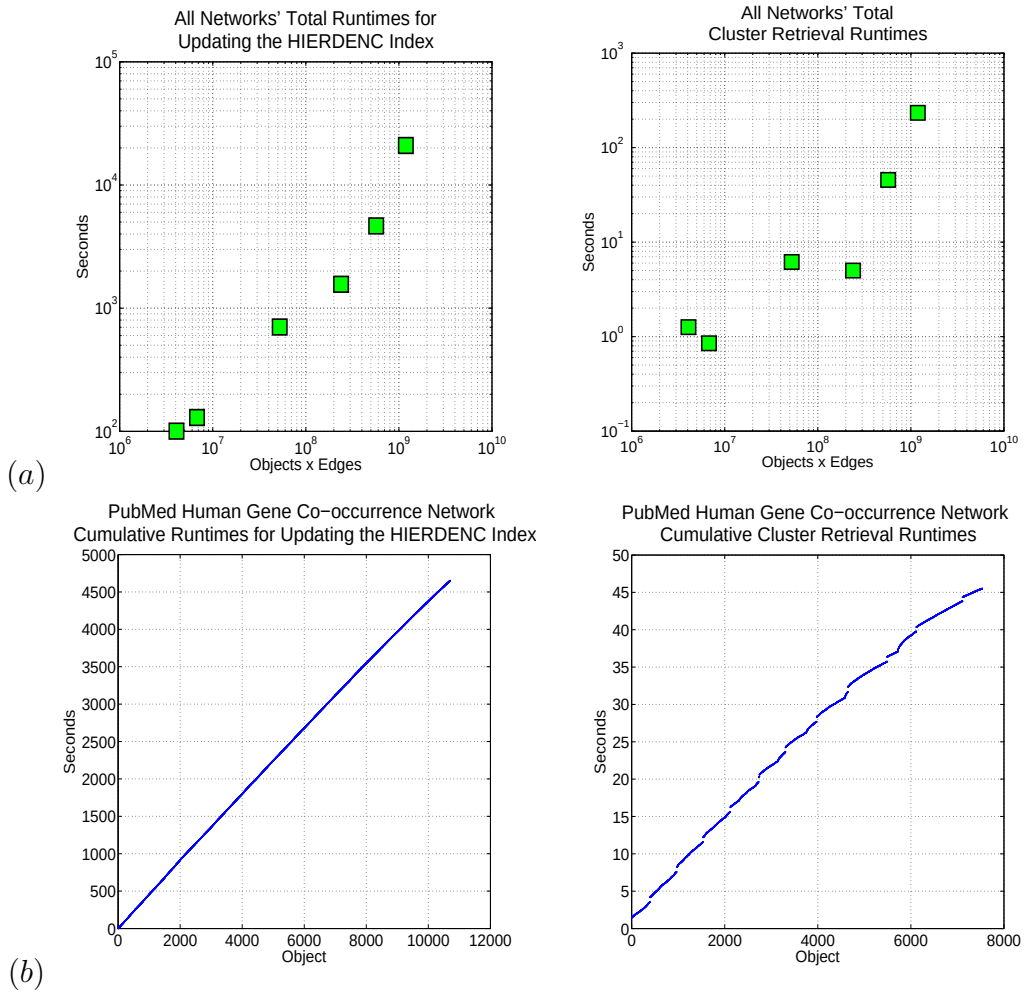


Fig. 9. Runtimes. (a) All networks: total runtimes (y-axis) for updating and retrieval by network size (x-axis). (b) Pubmed human gene co-occurrences network: cumulative runtimes (y-axis) for updating as new objects are introduced (x-axis), and retrieving the densest subspace centered at each object (x-axis).

not the dense subspace retrieval runtimes of Figures 8 and 9.

Figure 6 shows the HIERDENC tree for *zoo*, with 17 leaf clusters. 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 $r \geq 4$ values. The rest of the leaf clusters were created for lower $r < 4$ values. We cut off the HIERDENC *zoo* tree at level $r = 1$, resulting in 8 clusters. A low number of clusters results in improved F1-measures. There are a few cases of incorrectly clustered objects by cutting at $r = 1$.

6 Overview of Biomedical Informatics Applications

In this section we provide an overview of the diverse biomedical problems to which we applied HIERDENC. These applications show that HIERDENC is useful for biomedical datasets that grow quickly, requiring scalable runtimes as well as good results.

6.1 Application#1: Clustering networks to find bicliques

HIERDENC clustering is suitable for finding bicliques in a network, since it can find fast nodes with any given number of common neighbors. We use the Hamming distance for binary strings (section 2.1), such that the similarity of two nodes in a network is defined as the number of their common neighbors.

In our network representation, each node is represented as a row by its unique name followed by the names of its neighbors, i.e., the node(s) to which it is connected:

```
node : connected to node(s)
node1 : nodeA nodeC ... nodeZ
nodeN : nodeA nodeB ... nodeZ
```

In our biclique representation, a cluster or biclique is represented by two attributes, *includes* lists the member nodes, and *comm_nei* represents common neighbors of all the member nodes:

```
clusterC includes: node1 ... nodeN
clusterC comm_nei: nodeA ... nodeZ
```

A node that appears twice or more in the second column of the network representation should appear in at least one biclique's *comm_nei* row; the corresponding nodes in the first column of the network representation should appear together in at least one biclique's *includes* row.

The clustering process ensures that if node1 and node2 have one or more common neighbors, then node1 and node2 will be together in a cluster and their common neighbors collapsed.

(1) For $\sigma \in range(size(network) - 1, \cdots, 1)$:
(2) For $\mathbf{x_0} \in \lambda_\sigma$ ordered by $\lambda_\sigma(\mathbf{x_0})$:
(3) $C = \{\mathbf{x} \in network | similarity(\mathbf{x}, \mathbf{x_0}) = \sigma\}$.
(4) Collapse σ common neighbors of $(\mathbf{x}, \mathbf{x_0})$.
(5) Return C.

Network	*original edges*	*removed*	*collapsed bicliques*	*edge reduction*
Gavin06	93881	84526	3258	86.5%
Krogan06	14292	13808	2803	77%
Stelzl05	2668	2464	653	67.87%
Rual05	3618	3202	849	65%
Internet topology	59582	51068	5322	76.77%
Pubmed gene co-occs	53319	51496	9899	78%

Table 3
Edge reduction achieved on the networks with HIERDENC.

Figure 10 illustrates an example of HIERDENC network clustering, where every common neighbor is collapsed. For visualization, we can remove any *duplicates*, where a cluster is a subset of other cluster(s) that have the same *comm_nei* nodes.

We evaluate our success in finding bicliques in a network via the edge reduction achieved, since many edges are collapsed. Edge reduction in a network will be high if every biclique is found and collapsed. Table 3 shows the edge reduction that HIERDENC achieved for all networks, which was as high as 86.5%. Figure 11 shows the edges reduced along the process of iterating through all nodes. As shown, initially the highest edge reduction is achieved for a low number of indirect second-level neighbors; the reason is that there are few objects that share the largest neighborhood in a network (power-law). As the clustering progresses, a similar edge reduction is achieved for progressively larger numbers of indirect second-level neighbors that have smaller shared neighborhoods.

Finding bicliques was used in the past for visualization of biological networks, such as protein-protein interaction, homology, and gene regulatory networks. The "power graphs" tool focuses on visualization of networks as hierarchically organised bicliques, where the bicluqes are collapsed "power" edges (5). However, the tool did not address the speed of finding bicliques nor dealing with very large networks, which we have addressed.

6.2 Application#2: Clustering biomedical images and PubMed abstracts for a query

We used 30,000 biomedical images published in BMC during 2000-2007. We built an online HIERDENC-based system that is updated with image captions represented as word vectors. This system supports retrieving clusters

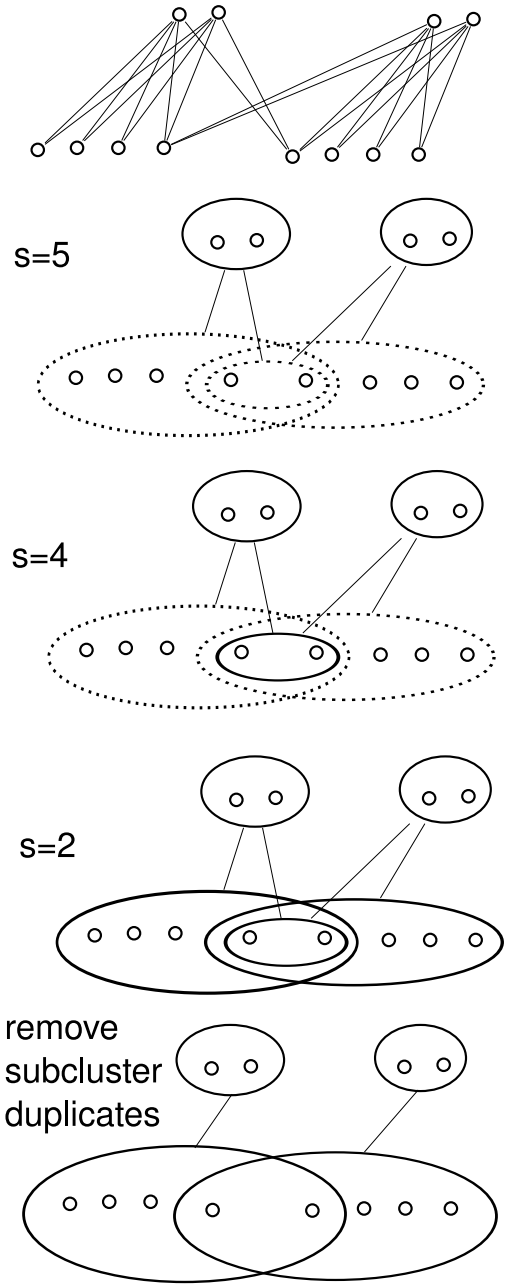


Fig. 10. Network clustering with HIERDENC that leads to finding bicliques. The similarity criterion s decreases progressively. The solid clusters are created at the corresponding step s, and the dotted clusters remain to be created in a future step. All common neighbors are found and collapsed.

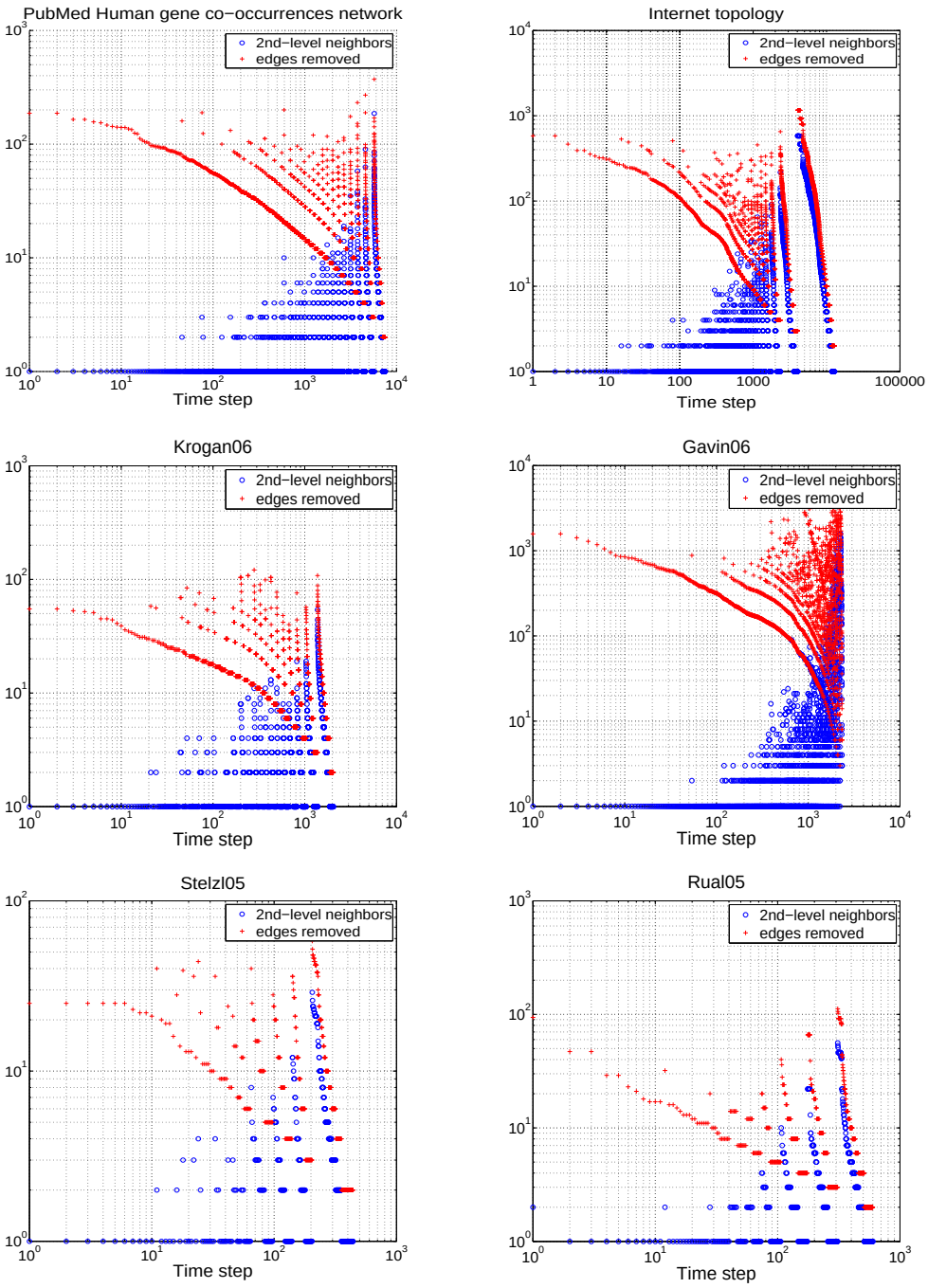
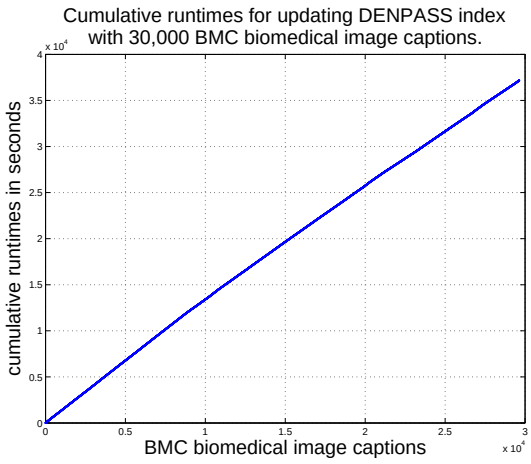


Fig. 11. Edges reduced and indirect second-level neighbors, as the clustering process iterates through all nodes (x axis): (a) Pubmed human gene co-occurrence network, (b) Internet topology network, (c) Krogan06, (d) Gavin06, (e) Stelzl05, (f) Rual05.

of BMC biomedical images, by querying the captions `http://141.30.193.16/HIERDENC/images.html`. Figure 12 shows that the system is updateable efficiently with newly introduced image captions. All of our clusters for this application were retrieved in less than 1 second.



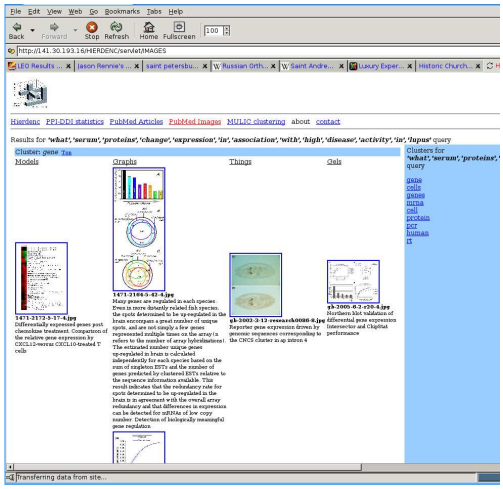


Fig. 12. *a*. HIERDENC index updating cumulative runtimes for 30,000 BMC biomedical image captions, and *b*. Querying our online service with the question: "What serum [PROTEINS] change expression in association with high disease activity in lupus?". The clusters are listed on the right-hand sidebar.

The PubMed database of biomedical literature has become a data mining resource. Given a query GO term, we want to quickly retrieve all relevant PubMed articles, such that articles with similar sets of GO terms are clustered together. Our HIERDENC application to this problem can prove useful for navigating biomedical literature. Figure 13 shows visualizations of clusters given a query on GO term "molecular transport". As shown, most of these abstracts are in clusters that are associated with a cellular meaning such as "nucleus", but a few abstracts are on a medical meaning such as "neuromuscular process controlling balance". Thus, such a HIERDENC clustering application could help a physician or bioresearcher navigate through the biomedical literature.

6.3 Application#3: Clustering Force-Distance curves

Our last biomedical application involves applying HIERDENC to retrieve clusters of aligned Force-Distance (FD) curves, which represent the force needed to pull a protein structure from cellular membranes using special machines. We first detect the peaks as long regions of an ascent and descent. Then, we match peaks between curves that are close to one another. This application is similar to the previous one, in that FD curves are analogous to image captions or PubMed abstracts and peaks are like words. Figure 14 shows an example of the FD curves that we cluster, and the runtimes for updating the HIERDENC index with 4,000 such curves.

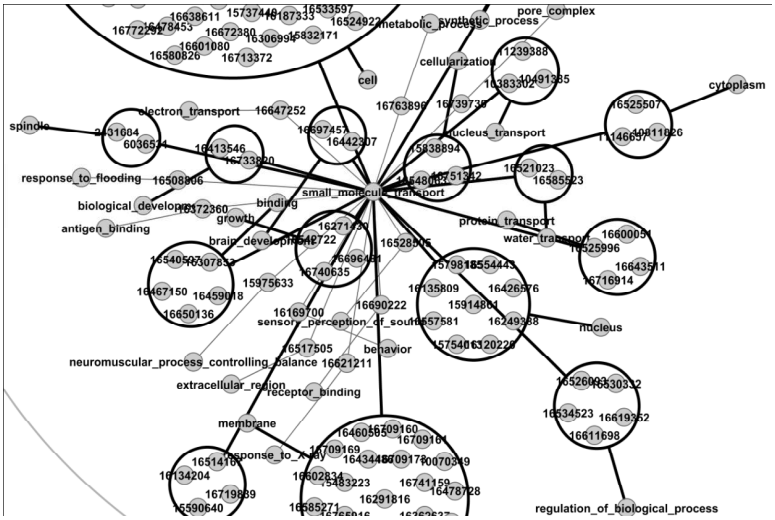


Fig. 13. Visualization of clusters of PubMed abstracts retrieved with HIERDENC, after querying with GO term "molecular transport". Numbers are PubMed abstract IDs and words are GO terms. All clusters are connected to another GO term, which acts as the cluster label. Most clusters are labeled with cell-related meanings, such as "protein transport".

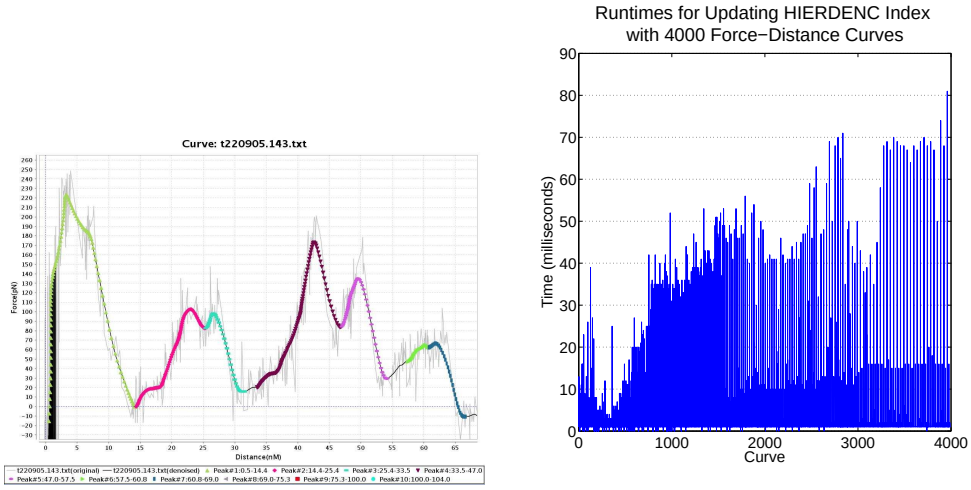


Fig. 14. *a*. An example of an FD curve. Colored segments denote peaks. *b*. Runtimes for updating HIERDENC index with the peaks found in 4,000 FD curves. On average the runtimes remain constant across curves, although there are some hikes too.

7 Conclusion

We presented the HIERDENC clustering algorithm for categorical data and networks. In HIERDENC the radius relaxes gradually, resulting in *layered clusters* where a central subspace often has a higher density. We presented the complementary HIERDENC index that supports efficient clustering. The HIERDENC index supports scalable runtimes for clustering categorical datasets

and networks. On networks, HIERDENC resulted in efficient extraction of bicliques. Further benefits of HIERDENC include: insensitivity to order of object input, no re-clustering needed when new objects are presented, no user-specified input parameters required, and ability to find clusters within clusters. Future work will include applying HIERDENC on real-world quickly growing databases for which it is suitable, such as PubMed images and documents, as well as curves and large sequence databases (4).

Availability

Python source code, all test datasets, and experimental results, are available under: `http://www.cse.yorku.ca/~billa/HIERDENC/`

Acknowledgements

We are grateful for the financial support of the Natural Science and Engineering Research Council (NSERC), the Ontario Graduate Scholarship (OGS), the EU Sealife project, Dresden-exists, and the Nanobrain project.

References

[1] Bill Andreopoulos, Aijun An, Xiaogang Wang, Michalis Faloutsos and Michael Schroeder. Clustering by common friends finds locally significant proteins mediating modules, Bioinformatics, Oxford University Press, 23(9): 1124-1131, 2007.

[2] B. Andreopoulos. Clustering Algorithms for Categorical Data. PhD Thesis, Dept of Computer Science & Engineering, York University, Toronto, Canada, 2006

[3] Bill Andreopoulos, Aijun An and Xiaogang Wang. Finding Molecular Complexes through Multiple Layer Clustering of Protein Interaction Networks. International Journal of Bioinformatics Research and Applications (IJBRA), 3(1):65-85, 2007.

[4] A. Morgulis, G. Coulouris, Y. Raytselis, T.L. Madden, R. Agarwala and A.A. Schaeffer. Database Indexing for Production MegaBLAST Searches. Bioinformatics (advance access June 2008).

[5] Loic Royer, Matthias Reimann, Bill Andreopoulos and Michael Schroeder. Unravelling the modular structure of protein networks with power graph analysis. PLoS Computational Biology. In press.

[6] Y. Zhang, A.W. Fu, C.H. Cai, P.A. Heng. Clustering Categorical Data. ICDE 2000
[7] G. Karypis, E.H. Han, V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. IEEE Computer 32(8): 68-75, 1999
[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] M. Ankerst, M. Breunig, H.P. Kriegel, J. Sander. OPTICS: Ordering Points to Identify the Clustering Structure. SIGMOD 1999
[10] A. Hinneburg, D.A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. KDD 1998
[11] S. Guha, R. Rastogi, K. Shim. ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Systems 25(5): 345-366, 2000
[12] J. Kleinberg, C. Papadimitriou, P. Raghavan. Segmentation Problems. STOC 1998
[13] R. Krauthgamer, J.R. Lee. The black-box complexity of nearest neighbor search. ICALP 2004
[14] P. Berkhin. Survey of Clustering Data Mining Techniques. Accrue Software, Inc. TR, San Jose, USA, 2002
[15] A. Gionis, A. Hinneburg, S. Papadimitriou, P. Tsaparas. Dimension Induced Clustering. KDD 2005
[16] J. Grambeier, A. Rudolph. Techniques of Cluster Algorithms in Data Mining. Data Mining and Knowledge Discovery 6: 303-360, 2002
[17] M.J. Zaki and M. Peters. CLICKS: Mining Subspace Clusters in Categorical Data via K-partite Maximal Cliques. In Proc. of the 21st International Conference on Data Engineering (ICDE) 2005.
[18] C.J. Mertz, P. Merphy. UCI Repository of Machine Learning Databases, 1996
[19] C. Aggarwal, J. Han, J. Wang and P.S. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In Proc. 30th VLDB Conference (VLDB'04), Toronto, Canada, 2004.
[20] C. Ding, X. He, H. Zha. Adaptive dimension reduction for clustering high dimensional data. ICDM 2002, pp.107-114, 2002.
[21] J. Han, M. Kamber. Data Mining: Concepts and Techniques, 2nd edition, Morgan Kaufmann, 2006.
[22] R. Xu. Survey of Clustering Algorithms. IEEE Transactions on Neural Networks, 16(3), May 2005.
[23] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. SIGMOD 1998
[24] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In Proceedings of the SIGMOD, pages 70-81, 2000
[25] V. Ganti, J. Gehrke, R. Ramakrishnan. CACTUS-clustering categorical data using summaries. KDD 1999

[26] D. Gibson, J. Kleiberg, P. Raghavan. Clustering Categorical Data: an Approach based on Dynamical Systems. VLDB 1998
[27] Y. Yang, S. Guan, J. You. CLOPE: a fast and effective clustering algorithm for transactional data. KDD 2002
[28] J. Sander, M. Ester, H.P. Kriegel and X. Xu. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. In Data Mining and Knowledge Discovery 2, 2, 169-194, 1998.
[29] Y. Chen, K.D. Reilly, A.P. Sprague and Z. Guan. SEQOPTICS: a protein sequence clustering system. BMC Bioinformatics, 7(Suppl 4):S10, 2006.
[30] 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
[31] P. Langfelder, B. Zhang, and S. Horvath. Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. Bioinformatics 24(5):719-720, 2008.
[32] R. Mojena, Hierarchical grouping methods and stopped rules: An evaluation. The Computer Journal, 20(4), 359-63, 1977
[33] T. Li, S. Ma, M. Ogihara. Entropy-Based Criterion in Categorical Clustering. ICML 2004
[34] H. Akaike. A new look at the statistical model identification. IEEE TAC, 19, 716-23, 1974
[35] 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
[36] J. Stutz and P. Cheeseman. Bayesian Classification (AutoClass): Theory and results. Advances in Knowledge Discovery & Data Mining, 153-180, 1995