Under consideration for publication in Knowledge and Information Systems

# Characterizing Pattern Preserving Clustering

Hui Xiong[1], Michael Steinbach[2], Arifin Ruslim[2], and Vipin Kumar[2]

[1] Department of Management Science and Information Systems
Rutgers, the State University of New Jersey, USA;
Email: hxiong@rutgers.edu

[2] Department of Computer Science and Engineering
University of Minnesota - Twin Cities
Email: {steinbac, aruslim, kumar}@cs.umn.edu

**Abstract.** This paper describes a new approach for clustering—pattern preserving clustering—which produces more easily interpretable and usable clusters. This approach is motivated by the following observation: while there are usually strong patterns in the data—patterns that may be key for the analysis and description of the data—these patterns are often split among different clusters by current clustering approaches. This is, perhaps, not surprising, since clustering algorithms have no built in knowledge of these patterns and may often have goals that are in conflict with preserving patterns, e.g., minimize the distance of points to their nearest cluster centroids. In this paper, our focus is to characterize (1) the benefits of pattern preserving clustering and (2) the most effective way of performing pattern preserving clustering. To that end, we propose and evaluate two clustering algorithms, **HI**erarchical **C**lustering with p**A**ttern **P**reservation (**HICAP**) and bisecting **K**-means **C**lustering with p**A**ttern **P**reservation (**K-CAP**). Experimental results on document data show that HICAP can produce overlapping clusters that preserve useful patterns, but has relatively worse clustering performance than bisecting K-means with respect to the clustering evaluation criterion of entropy. By contrast, in terms of entropy, K-CAP can perform substantially better than the bisecting K-means algorithm when data sets contain clusters of widely different sizes—a common situation in the real-world. Most importantly, we also illustrate how patterns, if preserved, can aid cluster interpretation.

**Keywords:** Pattern Preserving Clustering; K-means Clustering; Hyperclique Pattern; Hierarchical Clustering

*Received Jun 21, 2007*
*Revised Mar 16, 2008*
*Accepted Apr 12, 2008*

## 1. Introduction

Clustering and association analysis are important techniques for analyzing data. Cluster analysis (Jain and Dubes, 1988) provides insight into the data by dividing the objects into groups (clusters) of objects, such that objects in a cluster are more similar to each other than to objects in other clusters. Association analysis (Agrawal, Imielinski and Swami, 1993), on the other hand, provides insight into the data by finding a large number of patterns—frequent patterns and other patterns derived from them—in the data set. Frequent patterns identify sets of items (attributes) by finding attributes that occur together within a sufficiently large set of transactions. Thus, noting that clustering and association analysis can be performed either on objects or attributes, and restricting our discussion to binary transaction data, clustering and association analysis are both concerned with finding groups of *strongly related* objects or attributes, although at different levels. Association analysis finds strongly related objects or attributes on a local level, i.e., with respect to a subset of attributes or objects, while cluster analysis finds strongly related objects or attributes on a global level, i.e., by using all of the attributes or objects to compute similarity values.

Recently, we have defined a new pattern for association analysis—the *hyperclique pattern* (Xiong, Tan and Kumar, 2003; Xiong, Tan and Kumar, 2006)—that demonstrates a particularly strong connection between the overall similarity of a set of attributes (or objects) and the itemset (local pattern) in which they are involved. The hyperclique pattern is described in more detail later, but possesses the *high affinity property*: the attributes (objects) in a hyperclique pattern have a guaranteed level of global pairwise similarity to one another as measured by the cosine measure (uncentered Pearson's correlation coefficient[1]) (Strehl, Ghosh and Mooney, 2000). Since clustering depends on similarity, it seems reasonable that the hyperclique pattern should have some connection to clustering. To this end, we posed the following question: *What happens to hyperclique patterns when data is clustered by standard clustering techniques, e.g., how are they distributed among clusters?*

We found that hyperclique patterns are mostly destroyed by standard clustering techniques, i.e., standard clustering schemes do not preserve the hyperclique patterns, but rather, the objects or attributes comprising them are typically split among different clusters. To understand why this is not desirable, consider a set of hyperclique patterns for documents. The high affinity property of hyperclique patterns requires that these documents must be similar to one another; the stronger the hyperclique, the more similar the documents. Thus, for strong patterns, it would seem desirable (from a clustering viewpoint) that documents in the same pattern end up in the same cluster in many or most cases. As mentioned, however, this is not what happens for traditional clustering algorithms. This is not surprising since traditional clustering algorithms have no built in knowledge of these patterns and may often have goals that are in conflict with preserving patterns, e.g., minimize the distance of points from their closest cluster centroid.

More generally, the breaking of these patterns is also undesirable from an application point of view. Specifically, in many application domains, there are fundamental patterns that dominate the description and analysis of data within

[1] When computing Pearson's correlation coefficient, the data mean is not subtracted

that area, e.g., in text mining, collections of words that form a topic, and in biological sciences, a set of proteins that form a functional module (Xiong, He, Ding, Zhang, Kumar and Holbrook, 2005). If these patterns are not respected, then the value of a data analysis is greatly diminished for end users. If our interest is in patterns, such as hyperclique patterns, then we need a clustering approach that preserves these patterns, i.e., puts the objects or attributes of these patterns in the same cluster. Otherwise, the resulting clusters will be harder to understand since they must be interpreted solely in terms of objects instead of well-understood patterns.

To address the above challenges, we propose a pattern preserving clustering approach. Our goal is to characterize pattern preserving clustering with respect to the following two issues.

1. The benefits of pattern preserving clustering.
2. The most effective way of performing pattern preserving clustering.

One benefit of pattern preserving clustering is to provide a better cluster interpretation than traditional clustering approaches by considering the patterns found in clusters. To achieve this benefit, it is necessary to determine the most effective approach for pattern preserving clustering. One important aspect of this is the choice of which association pattern to use. Indeed, as demonstrated in the paper, the hyperclique pattern is a better candidate than frequent patterns (frequent itemsets) for pattern preserving clustering. First, the hyperclique pattern possesses the *high-affinity property.* As a result, hyperclique patterns tend to include objects that are from the same class (cluster), and thus, are well suited for pattern preserving clustering. In contrast, frequent patterns often contain objects from different classes that have low pairwise similarity. Second, hyperclique patterns have much better coverage of the targeted objects than frequent patterns since hyperclique patterns can be identified at low levels of support. Third, the computational cost of finding hyperclique patterns is significantly lower than that of frequent patterns (Xiong et al., 2003). Finally, there are fewer hyperclique patterns than frequent patterns, and thus, they are more manageable during the clustering process.

For instance, due to the high affiliation within hyperclique patterns, pattern preserving clustering naturally lends itself to various applications in search engines. For instance, instead of a long ranked list of keyword queries, it can be better to return clustered search results by topic. This can be accomplished by showing only the documents in the hyperclique patterns, which are more compact and representative of those topics. For a query session, instead of returning the documents where all the query words just co-occur, we can return the documents from hyperclique patterns that connect the queries and embody their common topics.

In our preliminary work (Xiong, Steinbach, Tan and Kumpar, 2004), we introduced the Hierarchical Clustering with Pattern Preservation (HICAP) algorithm, which is a pattern preserving clustering technique that utilizes hyperclique patterns to create the initial clusters, and then performs a Group Average agglomerative hierarchical clustering (also known as UPGMA (Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990)). To the best of our knowledge, HICAP is the first approach that is based on the idea of preserving patterns. While HICAP can produce overlapping clusters—non-overlapping clustering schemes tend to break patterns—and has better interpretations for clustering results, it is computationally more expensive than K-means. More importantly, the cluster quality of HICAP with respect to entropy is worse than bisecting K-means with respect to entropy (Xiong et al., 2004). As we know, k-means, which is a variant of traditional K-means algorithm whose performance for clustering document data is among the best (Steinbach, Karypis and Kumar, 2000; Zhao and Karypis, 2004).

In this paper, we present a bisecting **K**-means **C**lustering with p**A**ttern **P**reservation (**K-CAP**) algorithm that exploits key properties of bisecting K-means and the hyperclique pattern. K-CAP provides the following two important benefits.

1. *Improved Clustering Quality.* For data with widely differing cluster sizes, which is the normal case for many real-world data sets, K-CAP typically produces better quality results than bisecting K-means. For data with almost uniform clusters sizes, K-CAP has a clustering performance (with respect to entropy) that is comparable to that of bisecting K-means.This is noteworthy because, as mentioned, bisecting K-means is one of the best document clustering techniques (Steinbach et al., 2000; Zhao and Karypis, 2004).

2. *Pattern Preservation.* Strong patterns are preserved during the clustering process; that is, all objects in the same pattern end up in the same cluster. Traditional bisecting K-means clustering cannot preserve those strong patterns, since it has no built in knowledge of these patterns and may often have goals that are in conflict with preserving patterns; e.g., minimize the distance of points from their closest centroid.

The reason that K-CAP can produce better clustering quality than bisecting K-means for data sets with widely differing cluster sizes is explained in more detail later, but we briefly summarize the approach here. Before clustering, the group of objects in a hyperclique pattern is replaced by their centroid, thus reducing the size of the data set. Larger clusters are reduced more than smaller clusters and the distribution of objects among clusters becomes more uniform. Clustering is then performed using bisecting K-means, and the K-CAP algorithm assigns all the objects in a hyperclique pattern to the cluster containing their corresponding centroid to produce the final clustering results.

**Outline:** Section 2 provides a background in clustering and also describes related work. Section 3 introduces the hyperclique pattern, while Section 4 presents the details of the HICAP and K-CAP algorithms. Experimental results are given in Section 5. Section 6 provides a brief conclusion and an indication of our plans for future work.

## 2. Clustering Background and Related Work

Cluster analysis (Berkhin, 2002; Jain, Murty and Flynn, 1999; Brecheisen, Kriegel and Pfeifle, 2006; Hinneburg and Keim, 2003; Zhong and Ghosh, 2005) has been the focus of considerable work, both within data mining and in other fields such as statistics, machine learning, and pattern recognition. Several recent surveys may be found in (Berkhin, 2002; Jain et al., 1999), while more discussions of clustering are provided by the following books (Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990; Anderberg, 1973). The discussion in this section is, of necessity, quite limited.

While there are innumerable clustering algorithms, almost all of them can be classified as being either *partitional*, i.e., producing an un-nested set of clusters that partitions the objects in a data set into disjoint groups (Gondek and Hofmann, 2007), or *hierarchical*, i.e., producing a nested sequence of partitions, with a single, all-inclusive cluster at the top and singleton clusters of individual points at the bottom (Koga, Ishibashi and Watanabe, 2007). While this standard description of hierarchical versus partitional clustering assumes that each object belongs to a single cluster (a single cluster within one level, for hierarchical clustering), this requirement can be relaxed to allow clusters to overlap. Thus, in this paper we will describe clustering algorithms as hierarchical or partitional and as overlapping or non-overlapping.[2]

Perhaps the best known and most widely used partitional clustering technique is K-means (MacQueen, 1967), which aims to cluster a dataset into $K$ clusters—$K$ specified by the user—so as to minimize the sum of the squared distances of points from their closest cluster centroid. (A cluster centroid is the mean of the points in the cluster.) K-means is simple and computationally efficient, and a modification of it, bisecting K-means (Steinbach et al., 2000), can also be used for hierarchical clustering. Indeed, K-means is one of the best ways for generating a partitional or hierarchical clustering of documents (Steinbach et al., 2000; Zhao and Karypis, 2002). We use K-means, as implemented by CLUTO (Karypis, 2006), as one of the methods to compare with our approach.

Traditional hierarchical clustering approaches (Jain and Dubes, 1988) build a hierarchical clustering in an *agglomerative* manner by starting with individual points or objects as clusters, and then successively combining the two most similar clusters, where the similarity of two clusters can be defined in different ways and is what distinguishes one agglomerative hierarchical technique from another. These techniques have been used with good success for clustering documents and other types of data. In particular, the agglomerative clustering technique known as Group Average or UPGMA (Jain and Dubes, 1988; Kaufman and Rousseeuw, 1990), which defines cluster similarity in terms of the average pairwise similarity between the points in the two clusters, is widely used because it is more robust than many other agglomerative clustering approaches. Furthermore, a recent study found UPGMA to be the best of the traditional agglomerative clustering techniques for clustering text (Zhao and Karypis, 2002).

As far as we know, there are no other clustering methods based on the idea of preserving patterns. However, we mention three other types of clustering approaches that share some similarity with what we are doing here: constrained clustering, co-clustering, and frequent itemset based clustering. Constrained clustering (Tung, Ng, Lakshmanan and Han, 2001) is based on the idea of using standard clustering approaches, but restricting the clustering process. Our approach can be viewed as constraining certain objects to stay together during the clustering process. However, our constraints are automatically enforced by putting objects in hypercliques together, before the clustering process begins, and thus, the general framework for constrained clustering is not necessary for our approach. Also, co-clustering (Madeira and Oliveira, 2004) finds an optimal partitioning by co-clustering of both rows and columns of a data matrix. In a document clustering setting, co-clustering of both words and documents can pro-

[2] We admit that it is a bit strange to use the phrase, *overlapping partitional*, but in this case, we use *patitional* to mean *un-nested*.

vide more understandable clustering results. Viewed in this light, co-clustering is relevant to our pattern preserving clustering approach. However, our approach starts from a set of strong local patterns (hypercliques). Co-clustering schemes have to examine data in a global level and have no built in knowledge of these strong local patterns.

Our pattern preserving clustering technique is based on an association pattern, the hyperclique pattern, but there have been other clustering approaches that have used frequent patterns or other patterns derived from them (Beil, Ester and Xu, 2002; Fung, Wang and Ester, 2003; Wang, Xu and Liu, 1999; Ozdal and Aykanat, 2004). Specifically, Wang et al. (Wang et al., 1999) proposed a clustering approach based on the intuition that intra-cluster members should share many frequent items, while inter-cluster members should have little overlap in terms of frequent items. Beil et al. (Beil et al., 2002) proposed Hierarchical Frequent Term-based clustering (HFTC) for clustering documents. This technique uses a greedy approach to pick a set of frequent terms that has minimum overlap in terms of their document coverage. To improve HFTC, Fung et al. (Fung et al., 2003) proposed the Frequent Itemset-based Hierarchical Clustering (FIHC) method for document clustering. FIHC finds frequent terms and uses the documents covered by these frequent terms to create the initial clusters. Then, hierarchical clustering is performed using an intercluster similarity measure defined in terms of frequent patterns. However, these techniques are not designed for pattern preserving clustering. Finally, the hypergraph clustering approach in (Han, Karypis, Kumar and Mobasher, 1998) creates a hypergraph based on frequent itemsets and association rules, and then uses a hypergraph partitioning technique for finding clusters. Although hypergraph clustering inspired the clustering approach in the original hyperclique paper, this approach is not towards pattern preserving.

## 3. Basic Concepts of Association Patterns

The hyperclique pattern was the inspiration for pattern preserving clustering, and thus, the pattern that we use to explore this idea. In this section, we describe the concept of hyperclique patterns (Xiong et al., 2003), after first introducing the concepts on which it is based: the frequent itemset and the association rule (Agrawal et al., 1993).

### 3.1. Frequent Itemsets and Association Rules

We quickly review some standard definitions related to association rule mining, which is an important technique for mining market basket data (Agrawal et al., 1993).

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let $T$ be a set of transactions, where each transaction $t$ is a set of items such that $t \subseteq I$. An itemset is a set of items $X \subseteq I$. The **support** of $X$, $supp(X)$, is the fraction of transactions containing X. If the support of X is above a user-specified minimum, i.e., $supp(X) > minsup$, then we say that $X$ is a **frequent itemset**.

An association rule captures the fact that the presence of one set of items may imply the presence of another set of items, and is of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$. The **confidence** of the rule $X \rightarrow Y$ is written as $conf(X \rightarrow Y)$ and is defined as $conf(X \rightarrow Y) = supp(X \cup Y)/supp(X)$, where $supp(X \cup Y)$ is the **support** of the rule. For example, suppose 70% of all transactions contain bread and milk, while 50% of the transactions contain bread, milk, and cookies. Then, the support of the rule {bread, milk} → {cookies} is 50% and its confidence is 50%/70% = 71%.

## 3.2. Hyperclique Patterns

A hyperclique pattern (Xiong et al., 2003) is a new type of association pattern that contains items that are highly affiliated with each other.[3] By high affiliation, we mean that the presence of an item in a transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure (Xiong et al., 2003) is specifically designed to measure the strength of this association.

**Definition 1.** The **h-confidence** of an itemset $P = \{i_1, i_2, \cdots, i_m\}$, denoted as $hconf(P)$, is a measure that reflects the overall affinity among items within the itemset. This measure is defined as $min\{conf\{i_1 \rightarrow i_2, \ldots, i_m\}, conf\{i_2 \rightarrow i_1, i_3, \ldots, i_m\}, \ldots, conf\{i_m \rightarrow i_1, \ldots, i_{m-1}\}\}$, where $conf$ follows from the conventional definition of association rule confidence as given above.

For instance, consider an itemset $P = \{A, B, C\}$. Assume that $supp(\{A\}) = 0.1$, $supp(\{B\}) = 0.1$, $supp(\{C\}) = 0.06$, and $supp(\{A, B, C\}) = 0.06$, where $supp$ is the support of an itemset. Then

$$conf\{A \rightarrow B, C\} = supp(\{A, B, C\})/supp(\{A\}) = 0.6$$
$$conf\{B \rightarrow A, C\} = supp(\{A, B, C\})/supp(\{B\}) = 0.6$$
$$conf\{C \rightarrow A, B\} = supp(\{A, B, C\})/supp(\{C\}) = 1$$

Hence, $hconf(P) = min\{conf\{B \rightarrow A, C\}, conf\{A \rightarrow B, C\}, conf\ \{C \rightarrow A, B\}\} = 0.6$.

**Definition 2.** Given a transaction database and the set of all items $I = \{I_1, I_2, \ldots, I_n\}$, an itemset $P$ is a **hyperclique pattern** if and only if

1. $P \subseteq I$ and $|P| > 0$.
2. $hconf(P) \geq h_c$, where $h_c$ is a user-specified minimum h-confidence threshold.

Table 1 shows some hyperclique patterns identified from words of the LA1 dataset, which is part of the TREC-5 collection (TREC, 1996) and includes articles from various news categories such as 'financial,' 'foreign,' 'metro,' 'sports,' and 'entertainment.' One hyperclique pattern in that table is {*mikhai*, *gorbachev*}, who is the ex-president of the former Soviet Union. Certainly, the presence of *mikhai* in one document strongly implies the presence of *gorbachev* in the same document and vice-versa.

**Definition 3.** A hyperclique pattern is a **maximal hyperclique pattern** if no superset of this hyperclique pattern is also a hyperclique pattern.

In this paper, we use maximal hyperclique patterns as the patterns that we wish to preserve during the clustering process. Therefore, the study scope of this paper is on binary data.

[3] The hyperclique pattern is also known as the all-confidence pattern (Omiecinski, 2003).

**Table 1.** Examples of Hyperclique Patterns from words of the LA1 Data set

| LA1 Dataset | | |
|---|---|---|
| Hyperclique patterns | Support | H-confidence |
| {gorbachev, mikhail} | 1.4% | 93.6% |
| {photo, graphic, writer} | 14.5% | 42.1% |
| {sentence, convict, prison} | 1.4% | 32.4% |
| {rebound, score, basketball} | 3.8% | 40.2% |
| {season, team, game, play} | 7.1% | 31.4% |

### 3.3. Properties of the H-confidence measure

The h-confidence measure has three important properties, namely the anti-monotone property, the cross-support property, and the strong affinity property. Detailed descriptions of these three properties were provided in our earlier paper (Xiong et al., 2003). Here, we provide only the following brief summaries.

The **anti-monotone property** guarantees that if an itemset $\{i_1, \ldots, i_m\}$ has an h-confidence value greater or equal to $h_c$, then every subset of size $m-1$ also has an h-confidence value greater or equal to $h_c$. This property is analogous to the anti-monotone property of the support measure in association-rule mining (Agrawal et al., 1993) and allows us to use h-confidence-based pruning to speed the search for hyperclique patterns in the same way that support-based pruning is used to speed the search for frequent itemsets.

The **cross-support property** provides an upper bound for the h-confidence of itemsets that contain items from different levels of support. The computation of this upper bound is much cheaper than the computation of the exact h-confidence, since it only relies on the support values of individual items in the itemset. Using this property, we can design a partition-based approach that allows us to efficiently prune patterns involving items with different support levels.

The **strong affinity property** guarantees that if a hyperclique pattern has an h-confidence value above the minimum h-confidence threshold, $h_c$, then every pair of items within the hyperclique pattern must have a cosine similarity (Rijsbergen, 1979) greater than or equal to $h_c$. As a result, the overall affinity of hyperclique patterns can be controlled by setting an h-confidence threshold.

As demonstrated in our previous paper (Xiong et al., 2003), the anti-monotone and cross-support properties allow the design of an efficient hyperclique mining algorithm that has much better performance than frequent itemset mining algorithms, particularly at low levels of support. Also, the number of hyperclique patterns is significantly less than the number of frequent itemsets.

## 4. Algorithm Descriptions

In this section, we first discuss why hyperclique patterns are better than frequent itemsets for pattern preserving clustering. Then, we present the details of the **HI**erarchical **C**lustering with the p**A**ttern **P**reservation (**HICAP**) algorithm and the bisecting **K**-means **C**lustering with p**A**ttern **P**reservation (**K-CAP**) algorithm.

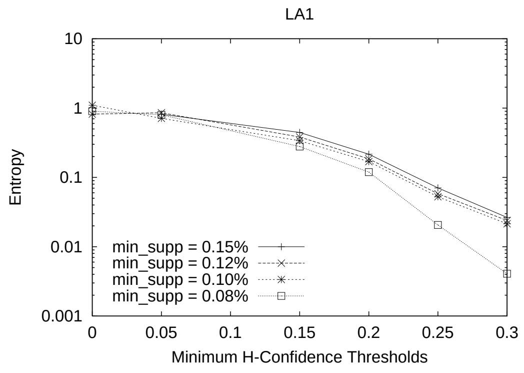


**Fig. 1.** Illustration of the high-affinity property of hyperclique patterns on the LA1 data set.

## 4.1. Association Pattern Selection

Here, we discuss the choice of an association pattern for pattern preserving clustering. Specifically, we present the results of an experiment that illustrates why the hyperclique pattern is a good pattern to use for pattern preserving clustering, but frequent itemsets are not. In this experiment, we employed entropy, a commonly used measure of purity. A formal definition of entropy is given below.

**Entropy.** To compute the entropy of a set of clusters, we first calculate the class distribution of the objects in each cluster, i.e., for each cluster $j$ we compute $p_{ij}$, the probability that a member of cluster $j$ belongs to class $i$. Given this class distribution, the entropy, $E_j$, of cluster $j$ is calculated using the standard entropy formula as follows.

$$E_j = -\sum_i p_{ij} \log(p_{ij}),$$

where the sum is taken over all classes and the log is log base 2. The total entropy for a set of clusters is computed as the weighted sum of the entropies of each cluster

$$E = \sum_{j=1}^{m} \frac{n_j}{n} E_j,$$

where $n_j$ is the size of cluster $j$, $m$ is the number of clusters, and $n$ is the total number of data points.

Figure 1 shows, for the LA1 data set (a document data set containing news articles from the Los Angeles Times), the entropy of the discovered hyperclique patterns for different minimum h-confidence and support thresholds. Note that when the minimum h-confidence threshold is zero, we actually have frequent itemsets instead of hyperclique patterns. Figure 1 shows that the entropy of hyperclique patterns decreases dramatically as the minimum h-confidence threshold increases. For instance, when the h-confidence threshold is higher than 0.25, the entropy of hyperclique patterns is less than 0.1 for all the given minimum support thresholds. This indicates that hyperclique patterns are very pure patterns for certain h-confidence thresholds. In other words, a hyperclique pattern includes objects that are naturally from the same class category. In contrast,

```
HICAP Algorithm
Input:      $D$: a document data set.
            $\theta$: a minimum h-confidence threshold.
            $\alpha$: a minimum support threshold.
Output:     CR: the hierarchical clustering result.
Variables:  S: the hyperclique pattern set.
            MS: the maximal hyperclique pattern set.
            LS: a set of objects that are not covered by maximal hyperclique
                patterns
            CS: a set containing target clustering objects
Method
Phase I: Maximum Hyperclique Pattern Discovery
 1.         S = hyperclique_miner($\theta$, $\alpha$, $D^T$) /* $D^T$ is the transpose of D */
 2.         MS = maximal_hyperclique_pattern(S)

Phase II: Hierarchical Clustering
 3.         LS = uncovered_objects(MS, D)
 4.         CS = LS $\cup$ {objects in MS}
 5.         for (i=1; i < |CS|; i++ )
 6.             Find the pair of elements with max group average cosine value
                from the set CS;
 7.             merge the identified pair, and update CS and CR accordingly;
 8.         endfor
 9.         OUTPUT CR
 10.        End
```

**Fig. 2.** The HICAP Algorithm

the entropy of frequent patterns is high—close to 1—for all the given minimum support thresholds. This means that frequent patterns include objects from different classes. Thus, with respect to purity as measured by entropy, the hyperclique pattern is a better candidate than frequent itemsets for pattern-preserving clustering.

Another trend that we can observe in Figure 1 is that, as the minimum support threshold decreases, the entropy of hyperclique patterns from the LA1 data set trends downward. This indicates that high affinity patterns can appear at very low levels of support. As mentioned, frequent itemset mining algorithms have difficulties at identifying frequent itemsets at low levels of support. In contrast, the hyperclique pattern mining algorithm has much better computational performance at low levels of support (Xiong et al., 2003).

## 4.2. HICAP: Hierarchical Clustering With Pattern Preservation

HICAP is based on the Group Average agglomerative hierarchical clustering technique, which is also known as UPGMA (Jain and Dubes, 1988). However, unlike the traditional version of UPGMA, which starts from clusters consisting of individual objects or attributes, HICAP uses hyperclique patterns to define the initial clusters, i.e., the objects or attributes of each hyperclique pattern become an initial cluster.

Figure 2 shows the pseudocode of the HICAP algorithm. This algorithm consists of two phases. In phase I, HICAP finds maximal hyperclique patterns, which

```
K-CAP Algorithm
Input:    A data set D
          A minimum h-confidence threshold $\theta$
          A minimum support threshold $\alpha$
          A target number of clusters $k$
Output: Clustering Result CR
Method
/* Phase I */
 1.   S←hyperclique_miner($\theta, \alpha, D^T$) /* $D^T$ is the transpose of D */
 2.   MS←maximal_hypercliques($S$)
 3.   DMS←disjoint_maximal_hypercliques($MS$)
/* Phase II */
 4.   MD← $D$
 5.   for $i = 1$ to $n_{DMS}$
 6.       $C$ ← Centroid(DMS[$i$]) /*Compute the centroid of the
              objects in the pattern DMS[i]*/
 7.       RemoveObjects(DMS[$i$], MD) /*Remove the objects in the
              pattern DMS[i] from MD*/
 8.       InsertCentroid(C,MD) /*Insert the centroid C into MD*/
 9.   CR ← Cluto(MD,$k$) /*Run CLUTO Bisecting K-means on the
              modified data set MD*/
10.   for $i = 1$ to $n_{DMS}$
11.       RemoveCentroid(DMS[$i$],MD) /*Remove the centroid of the
              objects in the pattern DMS[i] from MD*/
12.       InsertObjects(DMS[$i$],MD) /*Insert the objects in the
              pattern DMS[i] into the cluster of their centroid*/
13.   return CR
```

**Fig. 3.** Bisecting K-means Clustering with Pattern Preserving (K-CAP)

are the patterns we want to preserve in the HICAP algorithm. We use only maximal hyperclique patterns since any non-maximal hyperclique will, during the clustering process, tend to be absorbed by its corresponding maximal hyperclique pattern and will, therefore, not affect the clustering process significantly. Indeed, the use of non-maximal hypercliques would add complexity without providing any compensating benefits.

In phase II, HICAP conducts hierarchical clustering and outputs the clustering results. We highlight several important points. First, since hyperclique patterns can be overlapping, some of the resulting clusters will be overlapping. Second, identified maximal hyperclique patterns typically cover (contain) only 10 to 20% of all objects, and thus, HICAP also includes each uncovered object as a separate initial cluster, i.e., the hierarchial clustering starts with maximal hyperclique patterns and uncovered objects. Finally, the similarity between clusters is calculated using the average of the pairwise similarities between objects, where the similarity between objects is computed using the cosine measure.

## 4.3. K-CAP: bisecting K-means Clustering with pAttern Preservation

In this subsection, we describe the details of the bisecting **K**-means **C**lustering with p**A**ttern **P**reservation (K-CAP) algorithm. First, Figure 3 shows the pseudocode for K-CAP.

**Table 2.** Some Characteristics of Experimental Data Sets.

| Data set | Source | #docs | #terms | #classes | Minsize | Maxsize | Min/Max size |
|---|---|---|---|---|---|---|---|
| re0 | Reuters-21578 | 1504 | 2886 | 13 | 11 | 608 | 0.018 |
| re1 | Reuters-21578 | 1657 | 3758 | 25 | 10 | 371 | 0.027 |
| west5 | West | 311 | 1156 | 10 | 24 | 38 | 0.632 |
| wap | WebACE | 1560 | 8460 | 20 | 5 | 341 | 0.015 |
| oh8 | TREC | 839 | 2836 | 10 | 48 | 175 | 0.274 |
| la1 | TREC | 3204 | 31472 | 6 | 273 | 943 | 0.290 |
| hitech | TREC | 2301 | 126373 | 6 | 116 | 603 | 0.192 |
| tr12 | TREC | 313 | 5804 | 8 | 9 | 93 | 0.097 |
| tr32 | TREC | 516 | 7998 | 9 | 8 | 131 | 0.061 |
| fbis | TREC | 2463 | 2000 | 17 | 38 | 506 | 0.075 |

K-CAP consists of two phases. In the first phase, K-CAP computes the disjoint maximal hyperclique patterns. There are several implementation details. First, we use only maximal hyperclique patterns that contain at least three objects. A hyperclique pattern with three or more objects is less likely to be spurious, because, by definition, the cosine similarity of each pair of objects in the pattern must be greater than the minimum h-confidence threshold. When finding disjoint hyperclique patterns, the K-CAP algorithm gives higher priority to the hyperclique patterns containing more objects. For maximal hyperclique patterns that contain the same number of objects, the K-CAP algorithm gives higher priority to the patterns with higher h-confidence values. In case of ties, an arbitrary pattern is selected by the algorithm. The resulting hyperclique patterns are disjoint sets of highly correlated objects and these patterns are the input for the second phase of K-CAP.

In the second phase, the K-CAP algorithm removes the objects in each disjoint maximal hyperclique pattern from the data set and inserts the centroid vector of all objects in the pattern. Because multiple object vectors of the hyperclique are replaced by their centroid, the size of the data set is reduced. The modified data is clustered using bisecting K-means. Once the clustering results are produced by bisecting K-means, K-CAP assigns all the objects in a maximal hyperclique pattern to the cluster containing their corresponding centroid.

Note that non-overlapping maximal hyperclique patterns are used in order to preserve independent strong concepts. Also, the K-CAP algorithm uses maximal hyperclique patterns because non-maximal hyperclique patterns tend to be absorbed by their corresponding maximal hyperclique patterns during the clustering phase. Furthermore, fewer hyperclique patterns result in less computation. Finally, the CLUTO implementation of bisecting K-means (Karypis, 2006) has been used for the K-CAP algorithm.

*Computation Analysis.* K-CAP is a very efficient pattern based clustering algorithm. First, the computation of hyperclique patterns is much cheaper than that of frequent patterns, specially at low levels of support (Xiong et al., 2003). Second, the number of data objects for bisecting K-means clustering can be significantly reduced when the objects in each pattern are replaced by their centroid. The computation cost is reduced accordingly. Because of this and the efficiency with which hyperclique patterns can be found, K-CAP retains the computational efficiency of bisecting K-means (Steinbach et al., 2000).

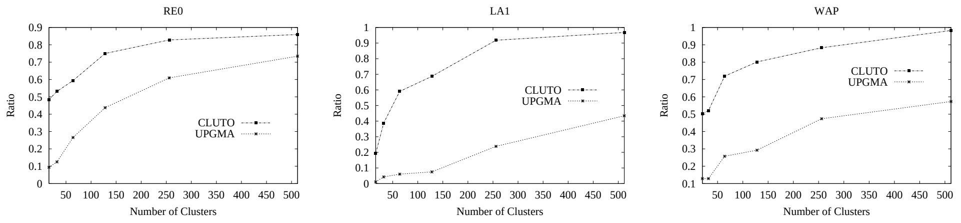


**Fig. 4.** The ratio of hyperclique patterns being split by UPGMA and CLUTO bisecting K-means.

## 5. Experimental Evaluation

In this section, we present an experimental evaluation of the HICAP and K-CAP algorithms. After a brief description of our document data sets, we first illustrate the poor behavior of traditional clustering approaches in terms of pattern preservation, and show how hyperclique patterns can be used to interpret the clustering results produced by pattern preserving clustering. In addition, we evaluate the clustering performance of HICAP, K-CAP, UPGMA, and K-means with respect to the entropy. Finally, we show how K-CAP can reduce the skewness of the class distribution.

*Experimental Data Sets.* For our experiments, we used various real-world data sets that are widely used in document clustering research. Some characteristics of these data sets are shown in Table 2. The data sets `RE0` and `RE1` are from the Reuters-21578 text categorization test collection Distribution 1.0 (Lewis, 2004). The `WEST5` data set came from the Thompson Publishing Group and was derived from legal documents. The data set `WAP` is from the WebACE project (WAP) (Han, Boley, Gini, Gross, Hastings, Karypis, Kumar, Mobasher and Moore, 1998); each document corresponds to a web page listed in the subject hierarchy of Yahoo!. The `OH8` data set was derived from the TREC-5 collection (TREC, 1996); the `LA1` data set is part of the TREC-5 collection (TREC, 1996) and contains news articles from the Los Angeles Times. Datasets `TR12` and `TR32` were derived from the TREC-5 (TREC, 1996), TREC-6 (TREC, 1996), and TREC-7 (TREC, 1996) collections. The `HITECH` data set contains documents about computers, electronics, health, medical, research, and technology. Finally, the `FBIS` data set is from the Foreign Broadcast Information Service data of the TREC-5 collection (TREC, 1996). For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm (Porter, n.d.).

*Evaluation.* To evaluate the quality of the clusters produced by the different clustering techniques, we employed entropy, which was introduced in Section 4. Entropy is an 'external' criterion; i.e., it uses external information—class labels in this case. Specifically, entropy measures the purity of the clusters with respect to the given class labels. Thus, if all clusters consist of objects with only a single class label, the entropy is 0. However, as the class labels of objects in a cluster become more varied, the entropy increases.

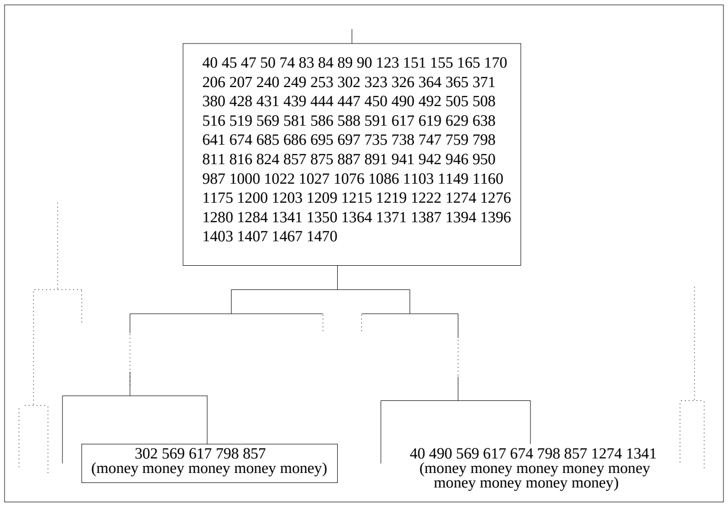


**Fig. 5.** Cluster Interpretation I

**Table 3.** Classes of Documents in an Example Cluster from the REO data set.

| |
|---|
| money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money interest money money interest money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money money |

## 5.1. Preserving Patterns

By design, HICAP and KCAP preserve all hyperclique patterns throughout the clustering process. However, as we show in this experiment, traditional clustering algorithms—UPGMA and bisecting K-means—tend to break hyperclique patterns. Figure 4 shows, for different number of clusters, the ratio of hyperclique patterns being split by the UPGMA and bisecting K-means algorithms. For every data set, the minimum number of clusters is specified as the original number of classes in that data set. In the figure, we observe that the ratio of patterns being split for both algorithms increases as the number of clusters increases. Furthermore, even when the number of clusters equals the number of classes, UPGMA and bisecting K-means still break patterns. Finally, bisecting K-means breaks more patterns than UPGMA, because its preference for relatively uniform cluster sizes tends to break long hyperclique patterns.

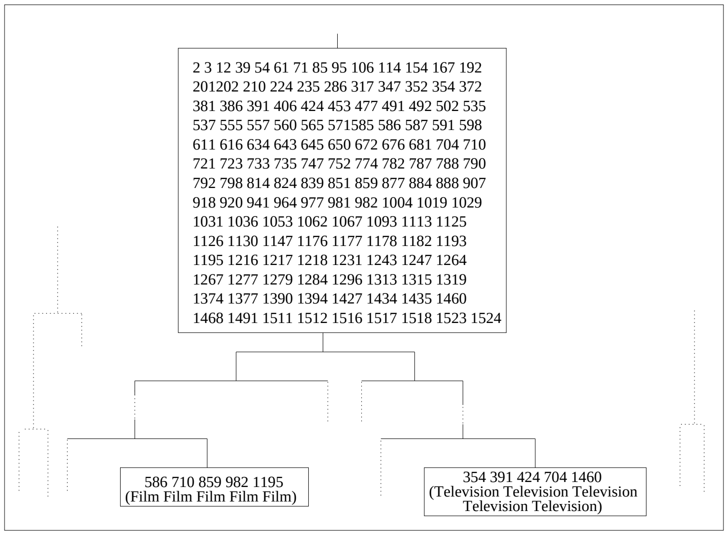


**Fig. 6.** Cluster Interpretation II

## 5.2. Interpretation of Clusters Using Hyperclique Patterns

In this experiment, we provide two types of evidence to illustrate the usefulness of patterns for interpreting clustering results: specific examples and an analysis of the clusters on one level of the cluster hierarchy. For the first example, we picked two clusters at random from the hierarchical clustering generated by HICAP, and then looked at the hyperclique patterns that they contained to see if the nature of these hypercliques, which include only a fraction of the documents or words in the cluster, are useful for understanding the nature of the cluster. As we show below, this was indeed the case.

**Cluster Contains Hypercliques of the Same Class** Figure 5 shows a cluster randomly selected from the HICAP clustering results on the `RE0` data set. One cluster with document IDs is presented. On further analysis, we found that two hyperclique patterns are in this cluster, and, as shown in the figure, both hyperclique patterns belong to the 'money' category. Since HICAP is based on the Group Average agglomerative clustering approach, it is natural to expect that other documents in the given cluster should have a significant level of similarity to the documents in two hyperclique patterns. In other words, if these two hyperclique patterns act as a 'kernel,' the documents merged into this cluster are likely to have the same class label as the documents in these two hyperclique patterns. As a result, we might expect that a large population of documents in this cluster would have the class, 'money.' To verify this, we show the class labels of the cluster objects in Table 3. As suggested by the two hyperclique patterns, all of the documents, except two, belong to the class, 'money.'

**Cluster Contains Hypercliques of Different Classes** Figure 6 shows another cluster randomly picked from the HICAP clustering of the `WAP` data set. This cluster contains two hyperclique patterns with documents from two different categories: 'Film' and 'Television.' As a result, we would expect that this cluster should be a hybrid cluster with documents mainly from two categories: 'Film'

**Table 4.** Classes of Documents in an Example Cluster from the WAP data set.

| |
|---|
| Television Television Television Film Film Television Stage Television Television Television Film Cable Television Television Television Variety Film Television Film Television Stage Television Film Television Film Television Film Television Film Television Television Film Cable Television Stage Film Television People Television Film People Television Cable Film Television Television Media Stage Television Film Television Film Television Television Stage Film Television Film Television Television Stage Film Television Film Television Television Film Film Television Television Cable Television Television People Television Film Television Film Television Television Film Television Variety Variety Television Film Film Cable Film Television Television Film Television Television Film Television Television Television Film Film Television Film Film Television Television Television Film Television Television Television Film Television Film Television Film Television Film Television Film Variety Film Television Film Industry Television Film Television Art Television Television Film Media Industry Stage Television Television Television Television Television |

**Table 5.** Statistics of interpretable clusters.

| CNo | size | #unmatch | #hyperclique | Classes of hypercliques |
|---|---|---|---|---|
| 1 | 49 | 5 | 16 | People/Online |
| 2 | 66 | 0 | 9 | Sports |
| 3 | 169 | 59 | 10 | Business/Tech/Politics |
| 4 | 313 | 2 | 49 | Health |
| 5 | 33 | 3 | 4 | Film |
| 6 | 61 | 9 | 9 | Politics |
| 7 | 18 | 0 | 7 | Culture |
| 8 | 44 | 2 | 1 | Television |
| 9 | 25 | 0 | 7 | Sports |
| 10 | 22 | 4 | 1 | People |
| 11 | 8 | 0 | 2 | Television/Stage |
| Total | 808 | 84 | 115 | |

and 'Television.' Table 4 shows the class labels of the cluster objects in this cluster. Once again, the interpretation based on hyperclique patterns matches the classes found in the cluster.

**Analyzing Clusters on one Level of the Cluster Hierarchy** To further validate the hypothesis that the nature of the hyperclique patterns contained in a cluster tells us something about the nature of the cluster, we decided to look

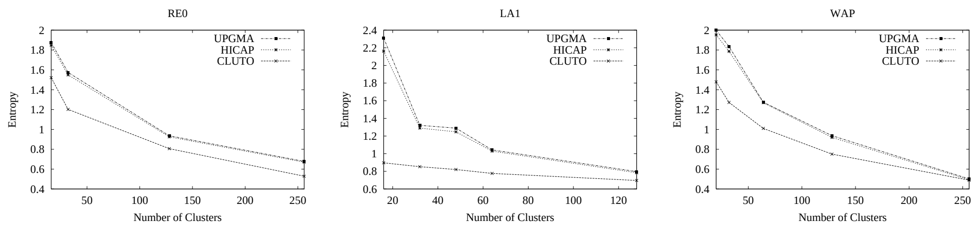


**Fig. 7.** Entropy Comparison on the LA1, RE0, and WAP data sets.

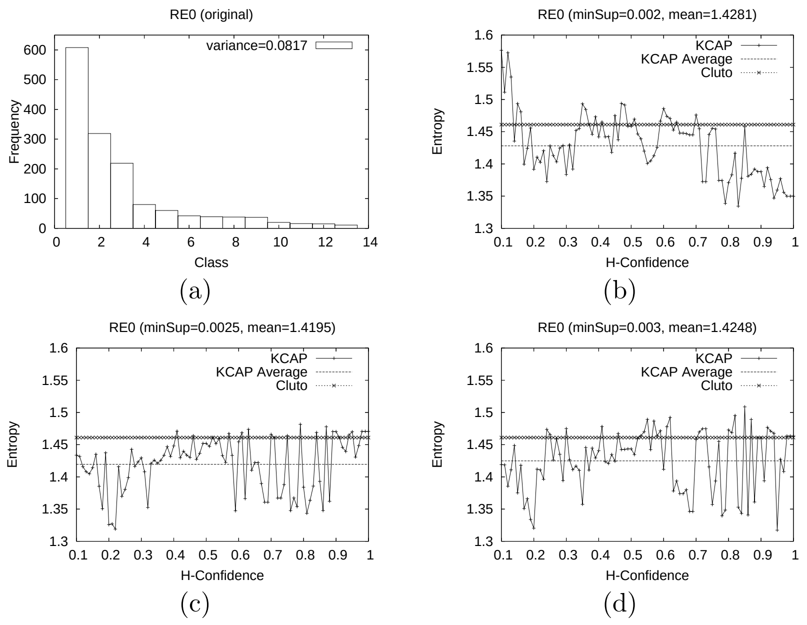


**Fig. 8.** (a) The original RE0 class distribution. (b), (c), and (d) show the entropy values of K-CAP and the CLUTO implementation of bisecting K-means at minimum support thresholds: 0.002, 0.0025, and 0.003 respectively (the natural number of classes = 13).

at the clusters on one level of the cluster hierarchy. We first identified the class of each of the hyperclique patterns—there were 115 of these patterns in the `WAP` data set, which together covered 265 out of 1560 documents. Finding the class of each hyperclique was an easy task since the hypercliques almost always consisted of objects of a single class, and if not, were predominantly of one class. Then, we found which of the 128 clusters contained hypercliques—there were 11 such clusters, which covered 808 of the 1560 documents (The skewed distribution of cluster sizes is a result of the skewed distribution of class sizes.). We further analyzed each cluster with respect to the classes of documents that it contained and whether the classes of the documents in the cluster matched the classes of the hypercliques in the cluster. The results of this analysis are contained in Table 5. ('CNo' is cluster number, 'size' is the number of objects in the cluster, '#unmatch' is the number of objects in the cluster that do not match a class of the hypercliques in the cluster, '# hyperclique' is the number of hypercliques in the cluster, and 'Classes of hypercliques' is the classes of the hypercliques.)

The results confirm the observations suggested by the previous two examples. If the hypercliques in a cluster are of one class, then the objects in that cluster are predominantly of the same class. On the other hand, if the hypercliques in a cluster are of mixed classes, then the objects in the cluster are also of mixed class, although they tend to be very heavily composed of the classes of the hypercliques. The worst case in the table is cluster 3, which has hyperclique patterns from three classes, and has 59 out of 169 documents that are not of these three classes. The documents in the other clusters almost always match the labels of the corresponding hyperclique patterns.

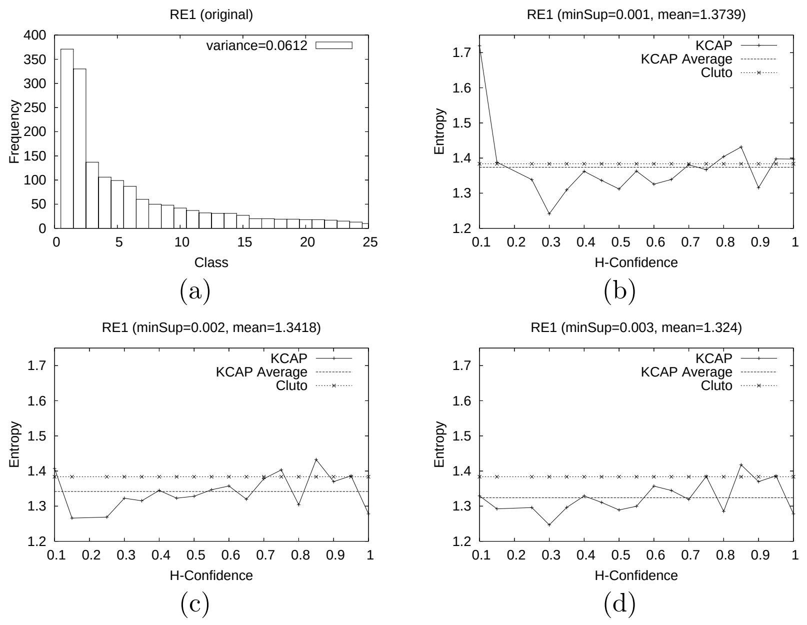


**Fig. 9.** (a) The original RE1 class distribution. (b), (c), and (d) show the entropy values of K-CAP and the CLUTO implementation of bisecting K-means at minimum support thresholds: 0.001, 0.002, and 0.003 respectively (the natural number of classes = 25).

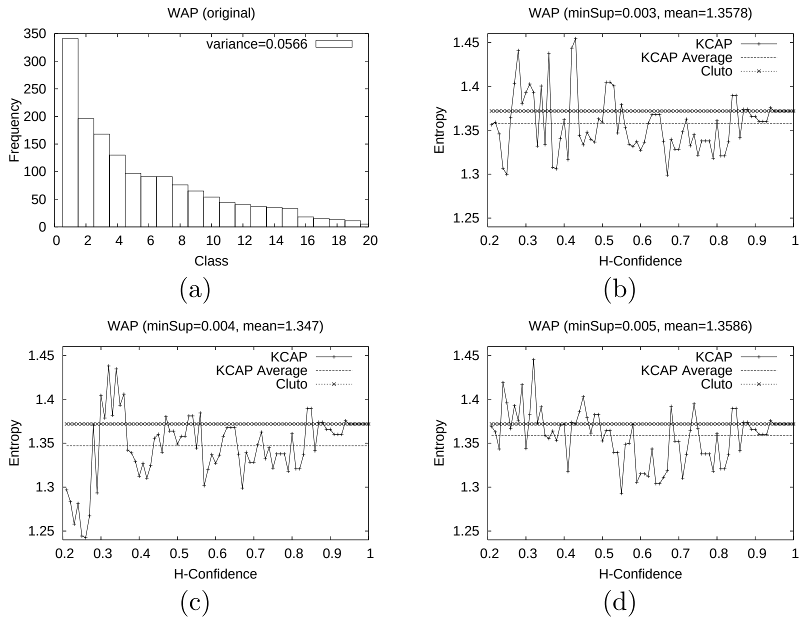


**Fig. 10.** (a) The original WAP class distribution. (b), (c), and (d) show the entropy values of K-CAP and the CLUTO implementation of bisecting K-means at minimum support thresholds: 0.002, 0.0025, and 0.003 respectively (the natural number of classes = 20).

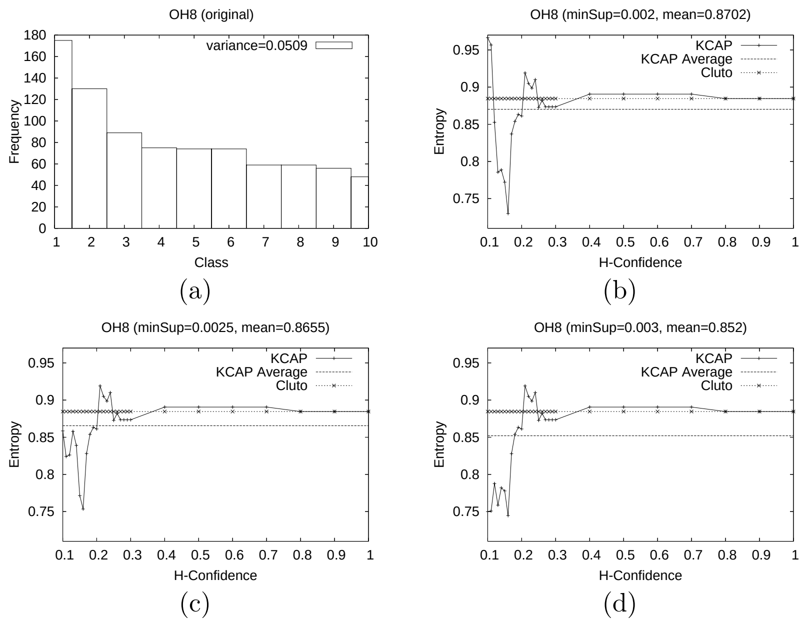


**Fig. 11.** (a) The original OH8 class distribution. (b), (c), and (d) show the entropy values of K-CAP and the CLUTO implementation of bisecting K-means at minimum support thresholds: 0.002, 0.0025, and 0.003 respectively (the natural number of classes = 10).

## 5.3. The Clustering Evaluation of HICAP using Entropy

Figure 7 shows the entropy values of the clustering results from HICAP, UPGMA, and bisecting K-means at different user-specified numbers of clusters. Bisecting K-means yields significantly better entropy values than HICAP and UPGMA for all three data sets. This is due to the fact that the entropy measure favors clustering algorithms, such as bisecting K-means, that produce clusters that have relatively uniform cluster size. Also, for all three clustering algorithms, entropy values tend to decrease as the number of clusters increases. The reason for this is that, when the number of clusters is increased, the resulting clusters tend to be more pure. Thus, the difference in entropy among the three algorithms decreases as we increase the number of clusters.

Another observation from Figure 7 is that HICAP performs slightly better than UPGMA in most cases for the given data sets. However, the performance difference between HICAP and UPGMA is tiny. This is not surprising since UPGMA starts from individual objects, while HICAP starts from hyperclique patterns (and the uncovered objects).

## 5.4. The Clustering Effect of K-CAP

Here, we compare clustering results of K-CAP and bisecting K-means on data sets with a skewed class distribution. Figures 8, 9, 10, and 11 show the entropy values of K-CAP and the CLUTO implementation of bisecting K-means at different minimum h-confidence settings for the data sets RE0, RE1, WAP, and OH8.

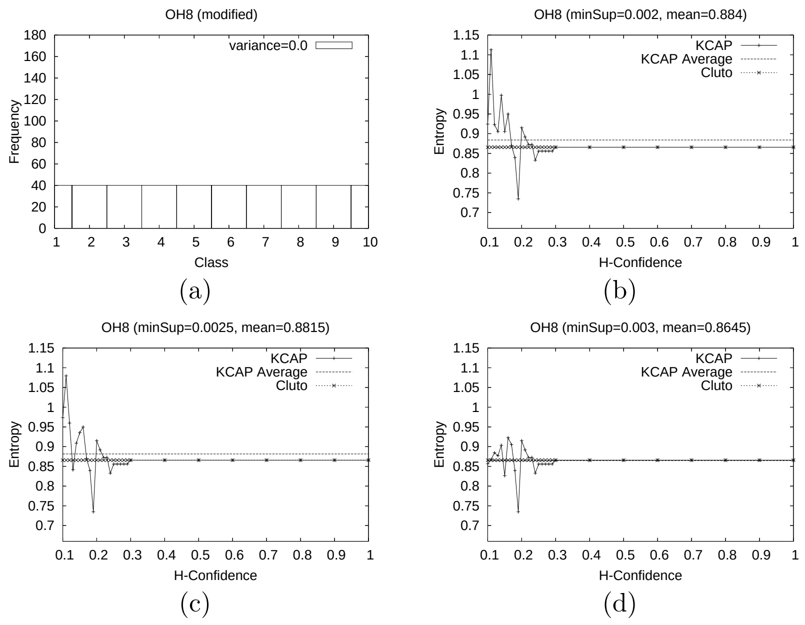


**Fig. 12.** Modified OH8 distribution and the corresponding K-CAP entropy (cluster=10).

All these data sets have very skewed class distributions as shown in Figures 8(a), Figures 9(a),10(a), and 11(a). For RE0, Figures 8 (b), (c), and (d) show the entropy values at different support thresholds. For all cases, the average entropy values achieved by K-CAP at different h-confidence thresholds is lower than the entropy value of CLUTO. In other words, the K-CAP algorithm can improve the clustering quality of bisecting K-means. Similar trends can also be observed for RE1, WAP, and OH8, as shown in Figures 9, 10, and 11, respectively.

To show the effect of skewness of class distribution on the clustering results of K-CAP, we present an additional experiment in which we take equal sized samples of each of the OH8 classes. Figure 12 shows the clustering results for this case. As shown in the figure, for the data set with uniform class distribution, the clustering quality of K-CAP is only comparable to that of K-means. However, if we sample the classes so that the class distribution becomes even more skewed than it was originally, the K-CAP algorithm tends to do a much better job than bisecting K-means, as illustrated in Figures 13 and 14. Note that all sample data sets have equal numbers of documents and there are multiple iterations for each test case.

## 5.5. Performance Comparison of Different Clustering Algorithms

In the previous subsection, we have showed that K-CAP can have better clustering quality than bisecting K-means on data sets with a skewed class distribution. In the literature, there are several clustering methods that are generally believed to perform well on data sets with a skewed class distribution, such as the unweighted pair-group average algorithm (UPGMA) (Jain and Dubes, 1988), biased

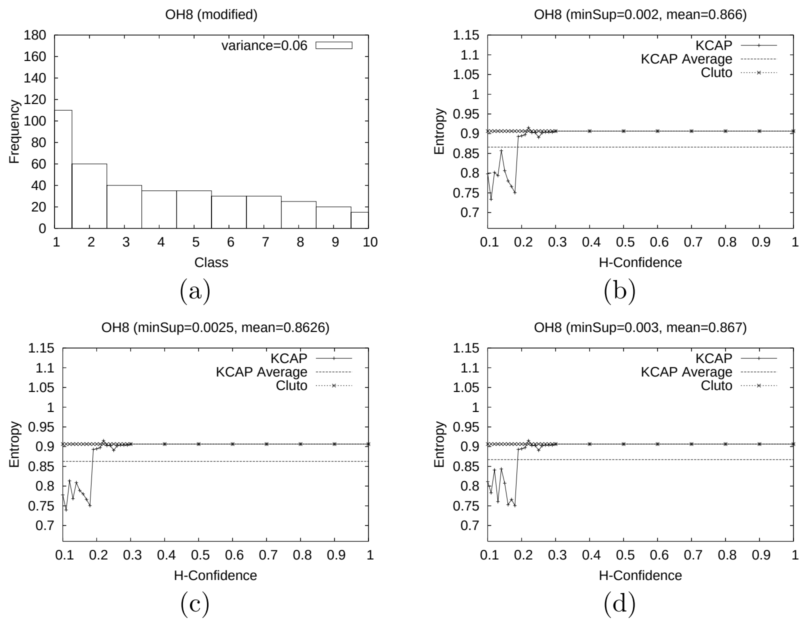


**Fig. 13.** Modified OH8 distribution and the corresponding K-CAP entropy (cluster=10).

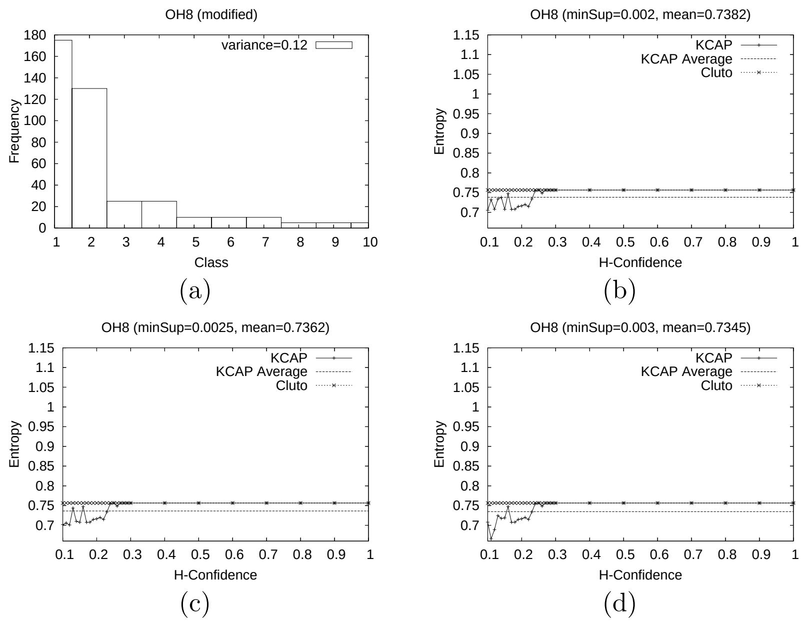


**Fig. 14.** Modified OH8 distribution and the corresponding K-CAP entropy (cluster=10).

**Table 6.** A Comparison of Different Clustering Algorithms on Data Sets with a Skewed Class Distribution

| Data | #Clu | Entropy | | | DBSCAN Entropy | | | DBSCAN Information | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | UPGMA | BAgglo UPGMA | Bisecting K-means | noise | w/o noise | noise reassign | Pts | Noise Pts | EPS | min Pts |
| re0 | 13 | 2.244 | 1.593 | **1.461** | 2.520 | 2.542 | 2.556 | 1335 | 169 | 0.32 | 4 |
| re0 | 30 | 1.510 | 1.204 | **1.087** | 2.277 | 2.249 | 2.279 | 1162 | 342 | 0.38 | 4 |
| re0 | 60 | 1.352 | 1.125 | **0.884** | 1.932 | 1.598 | 1.794 | 968 | 536 | 0.46 | 3 |
| re1 | 25 | 1.877 | 1.487 | **1.384** | 3.441 | 3.409 | 3.444 | 1503 | 154 | 0.28 | 3 |
| oh8 | 10 | 1.519 | 0.756 | **0.885** | 2.973 | 2.957 | 2.987 | 643 | 196 | 0.24 | 4 |
| oh8 | 30 | 1.256 | 0.660 | **0.617** | 2.512 | 2.190 | 2.372 | 533 | 306 | 0.29 | 3 |
| oh8 | 60 | 1.096 | 0.578 | **0.520** | 2.618 | 0.106 | 1.261 | 153 | 686 | 0.48 | 2 |
| west5 | 10 | 1.126 | 0.993 | **0.785** | 1.802 | 1.583 | 1.767 | 264 | 47 | 0.3 | 3 |
| la1 | 6 | 2.426 | 1.031 | **0.990** | 2.432 | 2.433 | 2.437 | 3136 | 68 | 0.11 | 2 |
| hitech | 6 | 2.383 | 1.664 | **1.571** | 2.403 | 2.406 | 2.405 | 2234 | 67 | 0.11 | 2 |
| wap | 20 | 1.942 | **1.367** | 1.372 | 3.494 | 3.569 | 3.602 | 1441 | 119 | 0.19 | 2 |
| tr12 | 8 | 1.797 | 0.832 | **0.812** | 2.280 | 2.307 | 2.341 | 287 | 26 | 0.21 | 2 |
| tr32 | 9 | 1.743 | **1.065** | 1.149 | 2.519 | 2.546 | 2.609 | 462 | 54 | 0.21 | 2 |
| fbis | 17 | 1.746 | **1.387** | 1.413 | 2.658 | 2.312 | 2.618 | 1715 | 748 | 0.45 | 7 |

agglomerative UPGMA (BAgglo-UPGMA)(Zhao and Karypis, 2004), and DBSCAN (Sander, Ester, Kriegel and Xu, 1998). Here, we compare the performance of bisecting K-means, UPGMA, DBSCAN, and BAgglo-UPGMA on data sets with a skewed class distribution.

Table 6 shows the entropy values of these algorithms on various different data sets for different parameter settings. In the table, we can observe that DBSCAN generally had poor clustering quality in most cases, regardless of whether noise objects were removed or not. This is not surprising since DBSCAN is a density-based clustering algorithm and does not perform well for data sets with high dimensionality. Although UPGMA has better performance than DBSCAN, it is still worse than BAgglo-UPGMA and Bisecting K-means. The reason is that UPGMA is a hierarchical clustering method that is sensitive to the choice of objects for initial clustering creation. If two objects are erroneously put in the same cluster at a lower level of hierarchy, this error cannot be corrected at a higher level. Finally, BAgglo-UPGMA is a hybrid approach that first applies K-means to form small groups and then uses hierarchical clustering to obtain the final set of clusters. This approach can alleviate problems due to erroneous merges during the initial stages and tends to produce a better quality clusters in terms of entropy. Nonetheless, in most cases, the clustering quality of BAgglo-UPGMA is still worse than bisecting K-means.

From the above, we know that bisecting K-means performs best among all these clustering algorithm, even when data sets have skewed class distributions.

## 5.6. Pattern Preservation for Reducing Skewness of the Class Distribution

The previous experiments reveal that a pattern preserving approach to clustering based on the hyperclique pattern can improve K-means clustering quality for data sets with a skewed class distribution. In this section, we provide an explanation for why K-CAP improves clustering quality for these cases. Specifically, larger classes tend to have a greater reduction in size than smaller classes when the objects in a hyperclique are replaced by the centroid of the hyperclique. Since it is well-known that K-means has trouble when clusters are of widely varying sizes, we hypothesize that this reduction in the skewness of the class distribution results in an improvement in K-means performance.

To illustrate the reduction of skewness of the class distribution, we performed two experiments. Figures 15 and Figure 16 show the change of class distributions

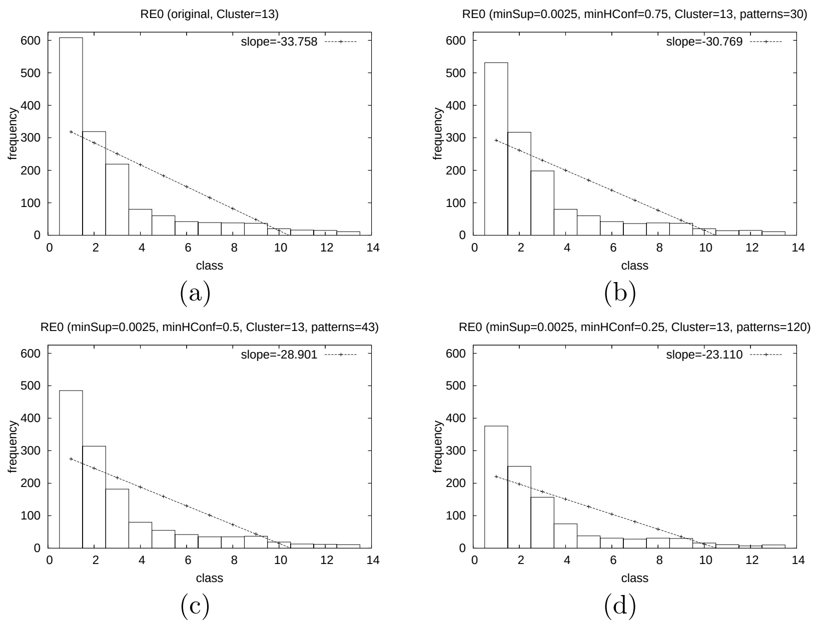


**Fig. 15.** The evolving class distribution with More patterns being preserved for the REO data set

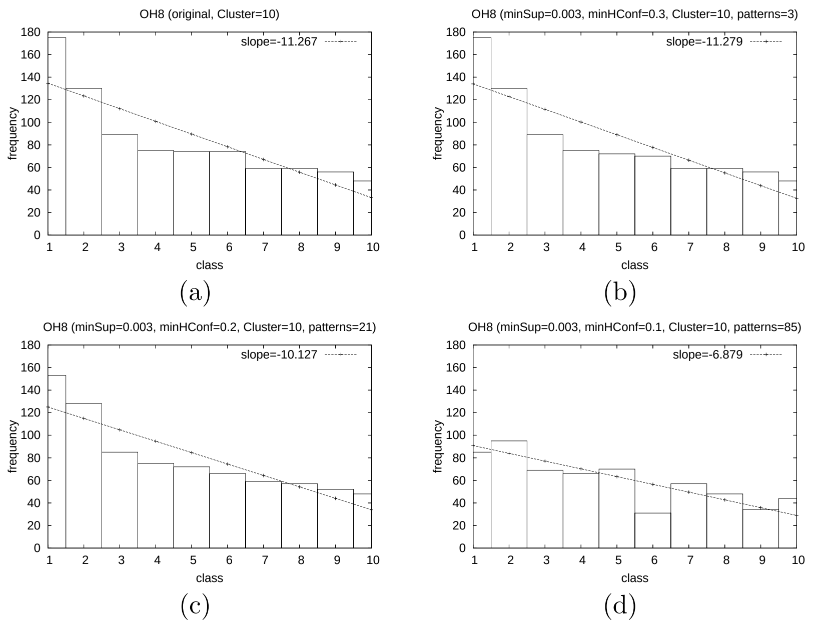


**Fig. 16.** The evolving class distribution with more patterns being preserved for the OH8 data set.

with the increase in the number of hyperclique patterns for the RE0 and OH8 data sets, respectively. To better show the skewness of class distribution, we have added a linear regression line to each of the figures. For both data sets, the slope of this line decreases. In other words, the skewness of the class distribution is reduced as the number of hyperclique patterns increases. For instance, when there are 85 hyperclique patterns for the OH8 data set, the slope of class distribution becomes -6.879. The original distribution had a slope of -11.267.

## 6. Conclusions

In this paper, we have introduced a new goal for clustering algorithms, namely, the preservation of patterns, such as hyperclique patterns, that capture strong connections between groups of objects. Without such an explicit goal, clustering algorithms tend to find clusters that split the objects or attributes in these patterns between different clusters. However, keeping these patterns together has the potential to greatly aid cluster interpretation.

To that end, we presented two pattern preserving clustering algorithms: **HI**erarchical **C**lustering with p**A**ttern **P**reservation (**HICAP**) and bisecting **K**-means **C**lustering with p**A**ttern **P**reservation (**K-CAP**). HICAP is based on the Group Average (UPGMA) agglomerative clustering technique and uses maximal hyperclique patterns to define the initial clusters. In contrast, K-CAP exploits the best properties of the hyperclique pattern and bisecting K-means. As demonstrated by our experimental results, HICAP can produce overlapping clusters (thus, not splitting patterns), and has a better interpretation on the clustering results. However, the cluster quality of HICAP is worse than bisecting K-means with respect to entropy (Xiong et al., 2004) and HICAP is computationally expensive. In contrast, our experimental results showed that K-CAP, which also preserves hyperclique patterns, produces better quality clustering results than bisecting K-means for data with widely differing clusters sizes, and retains the computational efficiency of bisecting K-means.

There are several potential directions for future research. We plan to further quantify the performance gains of K-CAP with respect to bisecting K-means by performing analysis for data with different types of class distributions. We also plan to further investigate why K-CAP reduces the skewness of the class distributions. Specifically, why are larger classes reduced by a larger factor than smaller classes? Finally, we propose to extend our methodology to handle data sets with continuous variables by using the continuous hyperclique approach we developed in (Steinbach, Tan, Xiong and Kumar, 2004).

## 7. Acknowledgments

We thank the editor and three anonymous reviewers for their very useful comments and suggestions. This work was partially supported by NSF ITR grant # ACI-0325949, NSF grant # IIS-0308264, the Center for Information Management, Integration and Connectivity (CIMIC) at Rutgers University, and the Wireless Information Network Laboratory (WINLAB) at Rutgers University. Also, this research was supported in part by the Rutgers Seed Funding for Collaborative Computing Research and a Faculty Research Grant from Rutgers Business School- Newark and New Brunswick. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

## References

Agrawal, R., Imielinski, T. and Swami, A. (1993), Mining association rules between sets of items in large databases, *in* 'Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data', pp. 207–216.

Anderberg, M. R. (1973), *Cluster Analysis for Applications*, Academic Press, New York.

Beil, F., Ester, M. and Xu, X. (2002), Frequent term-based text clustering, *in* 'Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM Press, pp. 436–442.

Berkhin, P. (2002), Survey of clustering data mining techniques, Technical report, Accrue Software, CA.

Brecheisen, S., Kriegel, H.-P. and Pfeifle, M. (2006), 'Multi-step density-based clustering', *Knowl. Inf. Syst.* **9**(3), 284–308.

Fung, B., Wang, K. and Ester, M. (2003), Hierarchical document clustering using frequent itemsets, *in* 'Proceedings of the Third SIAM International Conference on Data Mining', SIAM.

Gondek, D. and Hofmann, T. (2007), 'Non-redundant data clustering', *Knowl. Inf. Syst.* **12**(1), 1–24.

Han, E.-H., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B. and Moore, J. (1998), Webace: A web agent for document categorization and exploration, *in* 'Proceedings of the second International Conference on Autonomous Agents'.

Han, E.-H. S., Karypis, G., Kumar, V. and Mobasher, B. (1998), 'Hypergraph based clustering in high-dimensional data sets: A summary of results', *Bulletin of the Technical Committee on Data Engineering* **21**(1).

Hinneburg, A. and Keim, D. A. (2003), 'A general approach to clustering in large databases with noise', *Knowl. Inf. Syst.* **5**(4), 387–415.

Jain, A. K. and Dubes, R. C. (1988), *Algorithms for Clustering Data*, Prentice Hall Advanced Reference Series, Prentice Hall, Englewood Cliffs, New Jersey.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999), 'Data clustering: A review', *ACM Computing Surveys* (3).

Karypis, G. (2006), 'Cluto: Software for clustering high-dimensional datasets', http://www.cs.umn.edu/∼karypis.

Kaufman, L. and Rousseeuw, P. J. (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, John Wiley and Sons, New York.

Koga, H., Ishibashi, T. and Watanabe, T. (2007), 'Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing', *Knowl. Inf. Syst.* **12**(1), 25–53.

Lewis, D. (2004), Reuters-21578 text categorization text collection 1.0., *in* 'http://www.daviddlewis.com/resources/testcollections/reuters21578/'.

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations, *in* L. M. L. Cam and J. Neyman, eds, 'Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume I, Statistics', University of California Press.

Madeira, S. C. and Oliveira, A. L. (2004), 'Biclustering algorithms for biological data analysis: A survey', *IEEE/ACM Trans. Comput. Biology Bioinform.* **1**(1), 24–45.

Omiecinski, E. (2003), 'Alternative interest measures for mining associations in databases', *IEEE Trans. Knowl. Data Eng.* **15**(1), 57–69.

Ozdal, M. M. and Aykanat, C. (2004), 'Hypergraph models and algorithms for data-pattern based clustering', *Data Mining and Knowledge Discovery* **9**(1), 29–57.

Porter, M. F. (n.d.), An algorithm for suffix stripping, *in* 'Program, 14(3)'.

Rijsbergen, C. J. V. (1979), *Information Retrieval (2nd Edition)*, Butterworths, London.

Sander, J., Ester, M., Kriegel, H.-P. and Xu, X. (1998), 'Density-based clustering in spatial databases: The algorithm gdbscan and its applications', *Data Mining and Knowledge Discovery* **2**(2), 169–194.

Steinbach, M., Karypis, G. and Kumar, V. (2000), A comparison of document clustering techniques, *in* 'KDD Workshop on Text Mining'.

Steinbach, M., Tan, P.-N., Xiong, H. and Kumar, V. (2004), Generalizing the notion of support, *in* 'KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM Press, New York, NY, USA, pp. 689–694.

Strehl, A., Ghosh, J. and Mooney, R. (2000), Impact of similarity measures on web-page clustering, *in* 'Proc. of AAAI: Workshop of Artificial Intelligence for Web Search', AAAI, pp. 58–64.
**URL:** *citeseer.nj.nec.com/strehl00impact.html*

TREC (1996), *in* 'http://trec.nist.gov'.

Tung, A. K. H., Ng, R. T., Lakshmanan, L. V. S. and Han, J. (2001), Constraint-based clustering in large databases, *in* J. V. den Bussche and V. Vianu, eds, 'Database Theory - ICDT 2001, 8th International Conference', pp. 405–419.

Wang, K., Xu, C. and Liu, B. (1999), Clustering transactions using large items, *in* 'Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management', pp. 483–490.

Xiong, H., He, X., Ding, C., Zhang, Y., Kumar, V. and Holbrook, S. (2005), Identification of functional modules in protein complexes via hyperclique pattern discovery, *in* 'Proc. of the Pacific Symposium on Biocomputing'.

Xiong, H., Steinbach, M., Tan, P.-N. and Kumpar, V. (2004), HICAP: Hierarchial Clustering with Pattern Preservation, *in* 'Proceedings of 2004 SIAM International Conference on Data Mining (SDM)', pp. 279–290.

Xiong, H., Tan, P. and Kumar, V. (2003), Mining strong affinity association patterns in data sets with skewed support distribution, *in* 'Proceedings of the third IEEE International Conference on Data Mining (ICDM)', pp. 387–394.

Xiong, H., Tan, P.-N. and Kumar, V. (2006), 'Hyperclique pattern discovery.', *Data Mining and Knowledge Discovery Journal* **13**(2), 219–242.

Zhao, Y. and Karypis, G. (2002), Evaluation of hierarchical clustering algorithms for document datasets, *in* 'Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management', ACM Press, pp. 515–524.

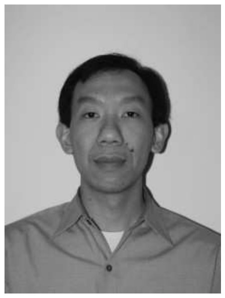
Zhao, Y. and Karypis, G. (2004), 'Criterion functions for document clustering: Experiments and analysis', *Machine Learning* **55**(3), Pages: 311–331.

Zhong, S. and Ghosh, J. (2005), 'Generative model-based document clustering: a comparative study', *Knowl. Inf. Syst.* **8**(3), 374–384.

## Author Biographies

**Hui Xiong** is currently an Assistant Professor in the Management Science and Information Systems department at Rutgers, the State University of New Jersey. He received the B.E. degree in Automation from the University of Science and Technology of China, China, the M.S. degree in Computer Science from the National University of Singapore, Singapore, and the Ph.D. degree in Computer Science from the University of Minnesota, USA. His general area of research is data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications. He has published over 50 technical papers in peer-reviewed journals and conference proceedings. He is the co-editor of Clustering and Information Retrieval (Kluwer Academic Publishers, 2003) and the co-Editor-in-Chief of Encyclopedia of GIS (Springer, 2008). He is an associate editor of the Knowledge and Information Systems journal and has served regularly in the organization committees and the program committees of a number of international conferences and workshops, and has also been a reviewer for the leading academic journals in his fields. He is a senior member of the IEEE, and a member of the ACM, the ACM SIGKDD, and Sigma Xi.

**Michael Steinbach** earned his B.S. degree in Mathematics, a M.S. degree in Statistics, and M.S. and Ph.D. degrees in Computer Science from the University of Minnesota. He is currently a research associate in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. Previously, he held a variety of software engineering, analysis, and design positions in industry at Silicon Biology, Racotek, and NCR. His research interests are in the area of data mining, bioinformatics, and statistics. He has authored over 20 research articles, and is a co-author of the data mining textbook, Introduction to Data Mining, published by Addison-Wesley. He is a member of the IEEE Computer Society and the ACM.



**Arifin Ruslim** received his M.S. degree in Computer Science from University of Minnesota in 2005, and the B.S. degree in Computer Science from University of Minnesota. He is currently a Senior Consultant in the Infrastructure Solutions practice of BearingPoint, with a focus on IT Services Management. He is the team leader for BearingPoint's Service Delivery Management solution which helps corporations to run IT as a business by improving the way they deliver and manage IT Services. Prior to full-time graduate study, Mr. Ruslim was a Principal Consultant at Computer Associates International (CA), focusing on Enterprise System Management technologies. Mr. Ruslim has an extensive experience in leading process and technology implementations in medium to large corporations to achieve IT operational excellence through data-driven continuous process improvement.

**Vipin Kumar** is currently William Norris Professor and Head of the Computer Science and Engineering Department at the University of Minnesota. Kumar's current research interests include data mining, bioinformatics and high-performance computing. He has authored over 200 research articles, and has coedited or coauthored 9 books including widely used text books "Introduction to Parallel Computing" and "Introduction to Data Mining", both published by Addison Wesley. Kumar has served as chair/co-chair for many conferences/workshops in the area of data mining and parallel computing, including IEEE International Conference on Data Mining (2002), International Parallel and Distributed Processing Symposium (2001), and SIAM International Conference on Data Mining (2001). Kumar is a founding co-editor-in-chief of Journal of Statistical Analysis and Data Mining, editor-in-chief of IEEE Intelligent Informatics Bulletin, and editor of Data Mining and Knowledge Discovery Book Series published by CRC Press/Chapman Hall. Kumar serves as co-chair of the steering committee of the SIAM International Conference on Data Mining, and is a member of the steering committee of the IEEE International Conference on Data Mining and the IEEE International Conference on Bioinformatics and Biomedicine. He is a Fellow of the ACM, IEEE and AAAS, and a member of SIAM. Kumar received the 2005 IEEE Computer Society's Technical Achievement Award for contributions to the design and analysis of parallel algorithms, graph-partitioning, and data mining.

*Correspondence and offprint requests to*: Hui Xiong, Management Science and Information Systems Department, Rutgers University, Newark, NJ 07102, USA. Email: hxiong@rutgers.edu