CA-Tree: A Hierarchical Structure for Efficient and Scalable Coassociation-Based Cluster Ensembles

Tsaipei Wang, *Member, IEEE*

***Abstract*—Cluster ensembles have attracted a lot of research interests in recent years, and their applications continue to expand. Among the various algorithms for cluster ensembles, those based on coassociation matrices are probably the ones studied and used the most because coassociation matrices are easy to understand and implement. However, the main limitation of coassociation matrices as the data structure for combining multiple clusterings is the complexity that is at least quadratic to the number of patterns N. In this paper, we propose CA-tree, which is a dendogram-like hierarchical data structure, to facilitate efficient and scalable cluster ensembles for coassociation-matrix-based algorithms. All the properties of the CA-tree are derived from base cluster labels and do not require the access to the original data features. We then apply a threshold to the CA-tree to obtain a set of nodes, which are then used in place of the original patterns for ensemble-clustering algorithms. The experiments demonstrate that the complexity for coassociation-based cluster ensembles can be reduced to close to linear to N with minimal loss on clustering accuracy.**

***Index Terms*—Cluster ensemble, coassociation matrix, multiple clusterings.**

I. Overview

A. Introduction

CLUSTERING is the process of identifying the underlying groups or structures in a set of patterns without the use of class labels. While there have been a large set of clustering algorithms (see [1]–[3] for reviews), they all have their limitations in terms of data characteristics that can be processed and types of clusters that can be found. The performance of many clustering algorithms also strongly depends on proper choices of parameters and/or initializations. As a result, the choice of appropriate clustering algorithms and/or parameters is highly problem dependent and often involves lots of heuristic choices or trial and error.

Following the success of classifier ensembles [4]–[6], cluster ensembles, the process of obtaining a combined clustering of the data using the information from multiple clusterings, have attracted much research interest in the past several years. The motivation behind cluster ensembles is that it is more likely to generate more stable, more reliable, or more meaningful clustering results with less sensitivity to parameter choices or initializations by combining multiple different clusterings of the same data. According to [7], ensembles of clusterings have advantages over individual clusterings in several aspects: robustness (better average performance), novelty (new solution different from the individual clusterings), stability and confidence estimation, and parallelism and scalability (combination of clusterings obtained at different times and locations or with different features). Ensemble clustering is identified in [8] as one of three major frontiers of clustering techniques in recent years. Many existing experimental results have indicated improved clustering results compared to the results of single-clustering runs.

B. Relevant Works in Cluster Ensembles

The algorithms for cluster ensembles consist of two main components. The first component is the method for generating the base clusterings (the individual clusterings in the ensemble), including the source of diversity (differences among the base clusterings). One possibility is to use different clustering algorithms for each base clustering. This is the approach in [9], where the goal is to identify "robust" clusters, each defined as a set of patterns that are in the same cluster in every base clustering. When a single algorithm is employed to produce all the base clusterings, the most common choices include expectation maximization (EM), as in [10] and [11], and k-means, as in [12]–[14]. Both EM and k-means have the built-in source of diversity from different initializations and, as in [12], different numbers of clusters in the base clusterings. Other possible sources of diversity include different orderings of the patterns for online clustering algorithms [15] and different linkage types for hierarchical agglomeration [16]. The different subsampling of data has been studied as well [17]. The relation between the degree of diversity and the quality of the combined clustering is the subject of [18].

A popular approach is to generate the base clusterings using different subspace projections of the original feature space [11], particularly when there are more than just a few features. It is indicated in [11] that such clustering ensembles work better than dimensionality reduction techniques such as principal component analysis. Some have also employed a similar approach, where a randomly selected subset of features is used in each base clustering [10], [14]. An extreme example is to use a large number of random 1-D projections, which is studied in [19], to demonstrate the effectiveness of combining many weak clusterings.

The second major component is sometimes called the "consensus function" and is concerned about the integration and representation of the combined information from multiple clusterings and the extraction of a final clustering from this representation. An example is to use a prototype in the original feature space to represent a cluster in a base clustering [20].

Manuscript received September 21, 2009; revised April 30, 2010; accepted September 10, 2010. Date of publication November 11, 2010; date of current version May 18, 2011. This paper was recommended by Associate Editor N. Chawla.

The author is with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: wangts@cs.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2010.2086059

However, this method is not applicable to the "knowledge-reuse framework," as suggested by Strehl and Ghosh [21], where the consensus function only has access to the base cluster labels of the patterns but not to the original features.

A straightforward approach when using only the base cluster labels is to consider the cluster ensemble as a transform of each pattern into a "label space," where each feature is the cluster label of a pattern in a base clustering. Some early works employ a voting scheme to select the final cluster label of each pattern [22], [23]. However, this requires solving the correspondence problem among the base clusterings through relabeling, a process that is somewhat problematic particularly when the base clusterings have different numbers of clusters. More recently, Ayad and Kamel [24] have described cumulative voting methods that are able to handle different numbers of clusters in the base clusterings. One can also apply a clustering algorithm to the patterns in the label space. Examples include the use of k-modes [25], EM [7], and the optimization of quadratic mutual information [7].

Graph-based representations have also been popular for combining the information from multiple clusterings. Representative algorithms include cluster-based similarity partitioning algorithm (CSPA), hyper-graph partitioning algorithm (HGPA), and metaclustering algorithm (MCLA), all three proposed in [21], as well as hybrid bipartite graph formulation (HBGF) [26]. An appropriate graph-cut algorithm can then be applied to determine the final clustering according to the type of graph (patterns as vertices for CSPA, clusters as hyperedges for HGPA, clusters as vertices in a metagraph for MCLA, and a pattern-cluster bipartite graph for HBGF).

The coassociation matrix [12], [27] is perhaps the most widely used data structure for combining the information from multiple clusterings. Example applications to practical problems include [28]–[31]. A coassociation matrix is a square matrix where each element represents the similarity between two patterns. This similarity is given by the probability of these two patterns being in the same cluster among all the base clusterings. In other words, a coassociation matrix provides a relational representation of a data set according to the base clusterings. Luo *et al.* [32] describe a method for building the coassociation matrix for data sets with mixed numerical and categorical features.

The final clustering can be derived from the coassociation matrix using many different clustering algorithms for relational data. The most common choices are probably hierarchical agglomeration with various linkage types [11], [12], [27], [30]. Graph-cut algorithms are used in CSPA [21], as well as in [28], with the coassociation matrix treated as a graph where each vertex represents a pattern and the weight of each edge represents the similarity between two patterns. Other examples include the use of spectral clustering in [32] and fuzzy k-means in [14], where each column is treated as a new transformed feature vector.

C. Motivations and Contributions

The main drawback of using coassociation matrices in cluster ensembles is its complexity. Since a coassociation matrix has N^2 elements (N being the number of patterns), it has a memory complexity of $O(N^2)$, and the time complexity is also at least $O(N^2)$ for both its creation and its partition into the final clustering. This quadratic complexity makes coassociation matrices unsuitable for large data sets. This problem seriously limits the applicability of related algorithms. For example, in [10] where several different algorithms for cluster ensembles are compared, CSPA, which uses coassociation matrices, is only applied to data sets of no more than 2000 points.

A straightforward approach to solve this complexity problem is to simply subsample the data. Slightly more robust is to first group the patterns into a large number of small preclusters (we use the term "preclusters" here to distinguish them from the clusters generated by the subsequent clustering algorithms) using a common algorithm like k-means. The prototypes of these preclusters are just their centroids in the feature space. The subsequent ensemble clustering is then applied to these prototypes instead of the original patterns, and the final cluster labels of the original patterns are determined from the final cluster labels of the prototypes according to the nearest neighbor rule. This is the approach taken by Fred and Jain [12], which also adds an additional step of outlier removal based on shared nearest neighbor analysis. An example of applying ensemble clustering to 301 prototypes for a data set of 4000 patterns is given in [12], although there is no discussion regarding the effect of using prototypes on clustering results.

While the aforementioned prototype-based method for complexity reduction is intuitive, it has several limitations that affect its applicability. First, there is no general rule for selecting the number of prototypes. If the number of prototypes is just set to a fixed percentage of N, which is the approach in [33], the complexity of using coassociation matrices is still at least quadratic to N. Second, the prototypes are points in the feature space. This means that we need to use all the features at once in the preclustering process. As a result, we are unable to use this method in multiview clustering (multiple clusterings obtained using different subsets of features or projections to subspaces). In addition, it becomes a complicated, if not impossible, task to incorporate existing clustering results in the form of cluster labels, which is a scenario mentioned in the discussion of the knowledge-reuse framework in [21].

Our goal in this paper is to present a new method for reducing the time and memory complexity of coassociation-based cluster ensemble algorithms, therefore significantly extending their applicability to larger data sets. A hierarchical structure called a coassociation tree (CA-tree for short) that is similar to a dendogram is built using the base cluster labels. A cut of this "dendogram" at a given threshold gives a preliminary partition of the data set into disjoint groups similar to the preclusters. We then compute the coassociation matrix and obtain the final clustering using the representatives of these groups. The name CA-tree arises from the fact that the size of a node is the minimum "degree of coassociation" (defined in the same way as the elements of an ordinary coassociation matrix) between the representative of this node and all its descendants. We list several advantages of CA-trees in the following.

1) The procedure for building CA-trees utilizes only the cluster labels of the patterns in the base clusterings without the need to access the original features. Therefore, CA-trees are applicable to multiview clustering and are able to incorporate past clustering results, therefore resolving the limitations of the prototype-based methods.

2) Compared with prototype-based methods, we do not have the poorly defined task of determining the number of prototypes. Instead, we only need to select a threshold. In addition, our experimental results indicate that the same threshold (at a fixed ratio of the number of base clusterings) can work well across different data sets.
3) We experimentally observe that the number of groups resulting from a cut to a CA-tree approximately follows a power-law relation with respect to N, with the exponent less than one and often less than 0.5. Therefore, the complexity of coassociation-based ensembles using these groups, being quadratic to their numbers, is much lower than $O(N^2)$ and often lower than $O(N)$. For the latter case, the complexity with respect to N is $O(N)$ from the procedure for building the CA-tree.
4) We propose a node reduction scheme to further reduce the number of groups, as well as the resulting complexity. We also point out potentially valuable information regarding the actual cluster characteristics (compact or elongated/curvilinear) by comparing the numbers of nodes before and after node reduction (see Section IV-F).

D. Summary of Notations

1) Data Set and Base Clusterings:

N	the number of patterns;
$X = \{\boldsymbol{x}_i\}$	the set of patterns $(1 \le i \le N)$;
H	the number of base clusterings;
P_h	the hth base clustering $(1 \le h \le H)$;
k_h	the number of clusters in P_h;
$k_{\min}/k_{\max}$	lower and upper bounds of k_h;
λ_{ih}	base cluster label of $\boldsymbol{x}_i$ in P_h;
$\boldsymbol{\lambda}_i$	label vector (the vector of all the λ_{ih}) of $\boldsymbol{x}_i$;
$\boldsymbol{\Lambda}$	the set of all the distinct $\boldsymbol{\lambda}_i$.

2) Properties of CA-Tree Nodes:

$\boldsymbol{z}$	a node in the CA-tree
$X(\boldsymbol{z})$	the set of data points associated with $\boldsymbol{z}$
$G(\boldsymbol{z})$	the set of distinct $\boldsymbol{\lambda}_i$ in $X(\boldsymbol{z})$
$\boldsymbol{\lambda}(\boldsymbol{z})$	partial label vector shared by $\boldsymbol{z}$ and all its descendants
$\boldsymbol{\lambda}_R(\boldsymbol{z})$	representative label vector of $\boldsymbol{z}$
$D_r(\boldsymbol{z})$	"radius" of $\boldsymbol{z}$ in the space of label vectors
$D'_r(\boldsymbol{z})$	an estimated upper bound of $D_r(\boldsymbol{z})$
$Z'(\boldsymbol{z})$	a subset of the descendants of $\boldsymbol{z}$ used for computing $D'_r(\boldsymbol{z})$
n_{des}	a bound on the size of $Z'(\boldsymbol{z})$

3) Node Selection:

τ	the threshold used to select a set of nodes;
$Z(\tau)$	the set of selected nodes at τ;
$N_Z(\tau)$	the number of nodes in $Z(\tau)$;
γ	the ratio of retained patterns in node reduction;
$Z^*(\tau)$	the version of $Z(\tau)$ without node reduction;
$N^*_Z(\tau)$	the version of $N_Z(\tau)$ without node reduction;
$\alpha(\tau)$	growth rate of $N_Z(\tau)$ relative to N.

4) Combined Clustering:

$\boldsymbol{S}^*(\tau) = [s^*_{ij}(\tau)]$	coassociation matrix obtained using $Z(\tau)$;
$Q(\tau)$	clustering accuracy at τ;
t_{total}	total time for getting the final clustering from a data set;
t_{CE}	time for getting the final clustering from base clusterings.

E. Paper Organization

For the rest of this paper, we first review the definition of coassociation matrices in Section II. Definitions and algorithms related to our tree structure are covered in Section III. Section IV presents the experimental results. We provide the analysis of complexity, as well as the experiments on computation time, in Section V, followed by the conclusions in Section VI.

II. Coassociation Matrices

Assume that $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ is a data set containing N patterns. Let $P = \{C_1, C_2, \ldots, C_k\}$ be a crisp clustering (partition) of X. Here, k is the number of clusters in P. The clusters $C_1, C_2, \ldots, C_k$ are disjoint nonempty subsets of X, with their union being X. It is possible to obtain many different partitions of X. Let a cluster ensemble consists of H clusterings of $X : P_1, P_2, \ldots, P_H$. These are called the base clusterings of the ensemble. We allow each base clustering to have a different number of clusters and use k_h to represent the number of clusters in $P_h (1 \le h \le H)$. We also define an H-element label vector for each $\boldsymbol{x}_i$ as

$$\boldsymbol{\lambda}_i = [\lambda_{i1} \quad \lambda_{i2} \quad \cdots \quad \lambda_{iH}] \tag{1}$$

with its hth element, denoted as λ_{ih}, being the cluster label of $\boldsymbol{x}_i$ in P_h. We use the Hamming distance between two label vectors $\boldsymbol{\lambda}_i$ and $\boldsymbol{\lambda}_j$ (i.e., the number of different cluster labels) as their dissimilarity, denoted as $d(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j)$.

We define an $N \times N$ matrix $\boldsymbol{S}^{(h)} = [s^{(h)}_{ij}]$ for each base clustering P_h according to

$$s^{(h)}_{ij} = \begin{cases} 1, & \lambda_{ih} = \lambda_{jh} \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

The overall coassociation matrix of a cluster ensemble, denoted as $\boldsymbol{S}^* = [s^*_{ij}]$, is simply the average of all the $\boldsymbol{S}^{(h)}$'s:

$$s^*_{ij} = \frac{1}{H} \sum_{1 \le h \le H} s^{(h)}_{ij}. \tag{3}$$

An equivalent definition using the label vectors is

$$s^*_{ij} = 1 - \frac{1}{H} d(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j). \tag{4}$$

III. CA-Tree

A. Core Groups

The first source of size reduction of a coassociation matrix arises from the observation that, often, multiple patterns share the same label vector. A set of patterns with the same label vector is indistinguishable for the cluster ensemble algorithms. As they belong to the same cluster in each base clustering, we shall expect that they are assigned to the same cluster in the final clustering regardless of the actual cluster ensemble algorithm used. Therefore, these patterns can be treated as a single entry, resulting in a smaller coassociation matrix and the reduction of associated computational time and memory requirement.

For this purpose, we introduce here the concept of core groups, defined as subsets of X that satisfy the following condition: Two patterns $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same core group if and only if $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_j$. Let $\boldsymbol{\Lambda}$ be the set of all the different

TABLE I
CORE GROUP LABEL VECTORS FOR FIG. 1

Core Group Index	1	2	3	4	5	6	7	8	9
Label Vector	1	1	1	1	2	2	3	3	3
	2	2	3	3	2	4	1	1	4
	2	5	4	4	1	1	3	3	3
	1	3	5	6	4	4	2	5	2
Number of Patterns	12	13	2	23	24	1	10	3	12

label vectors in X given the base clusterings. Since each core group has a unique label vector, the number of core groups is simply $\|\mathbf{\Lambda}\|$, which is the cardinality of $\mathbf{\Lambda}$. It is easy to understand that $\|\mathbf{\Lambda}\| \leq N$. The number of core groups depends on the distribution of patterns, as well as the algorithms and parameters used to generate the base clusterings. The procedure for identifying the core groups from the base clusterings is included in the next section. We note here that the concept of core groups has been previously mentioned in [16], although the method there for identifying the core groups is very different from ours and is only designed to work with a particular cluster ensemble algorithm.

The following example demonstrates the use of a $\|\mathbf{\Lambda}\| \times \|\mathbf{\Lambda}\|$ coassociation matrix of the core groups in place of an $N \times N$ coassociation matrix of the original patterns. Using a 2-D synthetic data set of four Gaussian clusters of 25 points each [Fig. 1(a)], four base clusterings are generated using k-means with k being three, four, five, and six, respectively. The base clusterings are shown in Fig. 1(d)–(g), with the numbers in the plots being the cluster labels. These four base clusterings result in $\|\mathbf{\Lambda}\| = 9$ core groups, as shown in Fig. 1(b), where the numbers represent the indices of the core groups. Here, we see that a problem size of 100^2 in terms of the number of elements in the coassociation matrix is reduced to 9^2 simply by taking advantage of the redundancy in base cluster labels. We list in Table I the associated label vector, as well as the number of patterns of each core group. Fig. 1(c) shows the resulting 9×9 coassociation matrix according to (3).

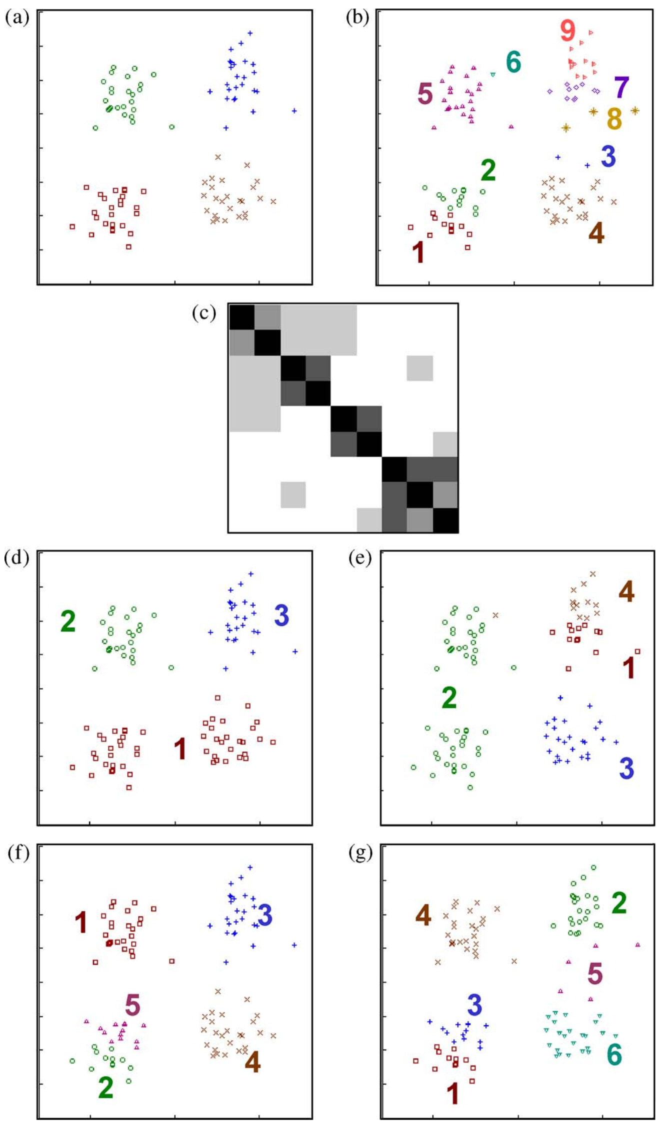


Fig. 1. (a) Four-cluster data set. (b) Nine core groups. The numbers in the plot indicate the core group indices. (c) Coassociation matrix of the core groups. (d)–(g) Four base clusterings used to obtain the core groups.

B. CA-Tree Construction

Further reduction of computational complexity can result from this assumption: Patterns with similar label vectors are more likely to be assigned to the same cluster in the final clustering. This leads to the use of groups that contain several similar core groups, instead of the individual core groups, as the units for building the coassociation matrix. Such an approach results in even fewer groups than $\|\mathbf{\Lambda}\|$ and further reduces the amount of computation and memory requirement.

This leaves us with the problem of how to form these groups from the core groups. Since our purpose is to lower the computational complexity below the quadratic complexity of the original coassociation matrix, we specifically exclude any option that starts with a matrix of pairwise similarity or dissimilarity among the original patterns or the core groups. Instead, with each base clustering of the data, we incrementally grow a tree structure that has some similarity to a dendogram formed with hierarchical clustering algorithms. When all the base clusterings have been processed, a threshold is applied to the tree to extract a set of nodes, each representing a group of core groups. We can then build a coassociation matrix of these groups as the input for final clustering extraction.

Each node of the tree contains one or more core groups of X. For a given node $\boldsymbol{z}$, we define $X(\boldsymbol{z})$ as the union of these core groups (i.e., the elements of $X(\boldsymbol{z})$ are the original patterns) and $G(\boldsymbol{z})$ as the set of label vectors associated with these core groups. We initialize the tree with a single root node where $X(\boldsymbol{z}) = X$. For each additional base clustering P, if $X(\boldsymbol{z})$ of a leaf node $\boldsymbol{z}$ belongs to more than one cluster in P, we add a child to $\boldsymbol{z}$ for each different cluster label of $X(\boldsymbol{z})$ in P. The purpose is to ensure that $X(\boldsymbol{z})$ of a leaf node $\boldsymbol{z}$ is always a core group. This process is described in the following pseudocode:

```
Initialize the tree with a single node $z_1$.
$X(\boldsymbol{z}_1) \leftarrow X$
$\boldsymbol{\lambda}(\boldsymbol{z}_1) \leftarrow$ an empty vector
$Z_{\text{leaf}} \leftarrow \{\boldsymbol{z}_1\}$ /* the current set of leaf nodes */
$m \leftarrow 1$ /* the current count of nodes */
For $h = 1$ to $H$
    $Z_{\text{leaf-temp}} \leftarrow \phi$ /* the set of newly created nodes */
    For each node $\boldsymbol{z}_i$ in $Z_{\text{leaf}}$
```

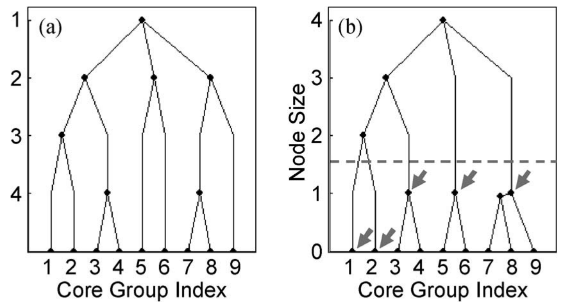


Fig. 2. (a) CA-tree generated with the data and base clusterings in Fig. 1. The vertical axis represents the base clustering at which the children of a node are created. (b) Same tree as in (a) with the vertical axis being the node size. The arrows point out the five nodes selected through thresholding (threshold indicated by the dashed horizontal line).

```
            $A \leftarrow \{t|X(\boldsymbol{z}_i) \cap C_t \neq \phi; C_t$ is a cluster in $P_h\}$
            If $\|A\| > 1$
                For each $t$ in $A$
                    $m \leftarrow m + 1$
                    Create a new node $\boldsymbol{z}_m$
                    $parent(\boldsymbol{z}_m) \leftarrow \boldsymbol{z}_i$
                    $Z_{\text{leaf-temp}} \leftarrow Z_{\text{leaf-temp}} \cup \{\boldsymbol{z}_m\}$
                    $\boldsymbol{\lambda}(\boldsymbol{z}_m) \leftarrow [\boldsymbol{\lambda}(\boldsymbol{z}_i)\ t]$
                    $X(\boldsymbol{z}_m) \leftarrow X(\boldsymbol{z}_i) \cap C_t$
                End For
            End If
        End For
        $Z_{\text{leaf}} \leftarrow Z_{\text{leaf}} \cup Z_{\text{leaf-temp}}$
    End For
```

All the processing within the outermost loop can be implemented with a single scan through all the patterns in X. As a result, the complexity of this algorithm is $O(NH)$, not including the generation of the base clusterings. As the algorithm terminates, each $\boldsymbol{\lambda}(\boldsymbol{z})$ is the partial label vector that is shared among $\boldsymbol{z}$ and all its descendants. If $\boldsymbol{z}$ is a leaf node, $X(\boldsymbol{z})$ is a single core group, and $G(\boldsymbol{z})$ contains a single label vector which is $\boldsymbol{\lambda}(\boldsymbol{z})$. The largest possible branching factor in the tree is $k_{h(\max)} \equiv \max(k_h)(1 \leq h \leq H)$, and the largest possible depth is H. We note here that the exact tree generated depends on the ordering of the clusterings. However, we do not expect the variation of clustering results among different orderings to be more significant than the variation among different ensembles generated under the same conditions. An example tree from the experiment for Fig. 1 is given in Fig. 2(a). The leaf nodes (nodes that correspond to the core groups) are all at the bottom. The vertical position of each nonleaf node indicates the index of the base clustering (index h in the pseudocode) at which its children are created.

C. Determining Node Sizes and Representatives

If we can define the "size" (in the space of label vectors) of a node $\boldsymbol{z}$ to represent the consistency of the label vectors in $G(\boldsymbol{z})$, we can redraw the tree more like a dendogram, with the vertical axis being the node size. Fig. 2(b) shows such an example (the computational detail of node sizes is described later). By applying a threshold on the node size, we cut the tree at a particular level. The horizontal dashed line in Fig. 2(b) is just an example of a threshold that results in a five-group partition; each of the five groups here corresponds to a node $\boldsymbol{z}$ [marked by arrows in Fig. 2(b)] such that the size of $\boldsymbol{z}$ is no more than the threshold while the size of $parent(\boldsymbol{z})$ is above the threshold. Some relevant definitions for this purpose are given hereinafter.

First, we define the distance between a label vector $\boldsymbol{\lambda}$ and a node $\boldsymbol{z}$ as the largest distance between $\boldsymbol{\lambda}$ and all the label vectors in $G(\boldsymbol{z})$

$$d(\boldsymbol{\lambda}, \mathbf{z}) = \max_{\boldsymbol{\lambda}' \in G(\mathbf{z})} d(\boldsymbol{\lambda}, \boldsymbol{\lambda}'). \tag{5}$$

For each node $\boldsymbol{z}$, the label vector $\boldsymbol{\lambda}$ in $G(\boldsymbol{z})$ that minimizes $d(\boldsymbol{\lambda}, \boldsymbol{z})$ is selected as the node representative $\boldsymbol{\lambda}_R(\boldsymbol{z})$

$$\boldsymbol{\lambda}_R(\mathbf{z}) = \arg\min_{\boldsymbol{\lambda} \in G(\mathbf{z})} d(\boldsymbol{\lambda}, \mathbf{z}). \tag{6}$$

Next, we define the size of a node $\boldsymbol{z}$ in the space of cluster labels as

$$D_r(\mathbf{z}) = d(\boldsymbol{\lambda}_R(\mathbf{z}), \mathbf{z}). \tag{7}$$

The subscript "r" indicates that its meaning is similar to radius. However, one problem with this definition is that its computation involves all the possible label vectors in $G(\boldsymbol{z})$. For those nodes that are several levels above the leaf nodes, their $G(\boldsymbol{z})$ may contain many label vectors, making the computation of (5)–(7) quadratic to $\|G(\boldsymbol{z})\|$ and, hence, too time consuming. This is inconsistent with our overall goal to reduce computational complexity. As a result, we opt for estimating the upper bound of $D_r(\boldsymbol{z})$ in place of the exact value. From here on, we use $D'_r(\boldsymbol{z})$ to represent the estimated upper bound of $D_r(\boldsymbol{z})$. The definitions (5)–(7) are replaced with the following estimations based on the estimated $D'_r(\boldsymbol{z})$. First, we need to select a subset $Z'(\boldsymbol{z})$ of the descendants of $\boldsymbol{z}$ (the actual procedure of selecting $Z'(\boldsymbol{z})$ is described later). Equation (5) is replaced with

$$d'(\boldsymbol{\lambda}, \mathbf{z}) = \min\left\{H, \max_{\mathbf{z}' \in Z'(\mathbf{z})} \left[d\left(\boldsymbol{\lambda}, \boldsymbol{\lambda}_R(\mathbf{z}')\right) + D'_r(\mathbf{z}')\right]\right\}. \tag{8}$$

The node representative is now determined as

$$\boldsymbol{\lambda}_R(\mathbf{z}) = \arg\min_{\boldsymbol{\lambda} \in G'(\mathbf{z})} d'(\boldsymbol{\lambda}, \mathbf{z}) \tag{9}$$

where we consider only the node representatives of those nodes in $Z'(\boldsymbol{z})$

$$G'(\mathbf{z}) = \left\{\boldsymbol{\lambda}_R(\mathbf{z}') | \mathbf{z}' \in Z'(\mathbf{z})\right\}. \tag{10}$$

The size of $\boldsymbol{z}$ is now estimated as

$$D'_r(\mathbf{z}) = d'\left(\lambda_R(\mathbf{z}), \mathbf{z}\right). \tag{11}$$

If $\boldsymbol{z}$ is a leaf node, we simply set $D'_r(\boldsymbol{z}) = 0$ and $\boldsymbol{\lambda}_R(\boldsymbol{z}) = \boldsymbol{\lambda}(\boldsymbol{z})$. The actual computation of (8)–(11) for all the nodes is done in a bottom–up order; this ensures that the processing of a node occurs only after all its descendants are already processed. During this process, we can also determine $G(\boldsymbol{z})$ and $X(\boldsymbol{z})$ according to

$$G(\mathbf{z}) = \bigcup_{\mathbf{z}', parent(\mathbf{z}')=\mathbf{z}} G(\mathbf{z}') \tag{12}$$

$$X(\mathbf{z}) = \bigcup_{\mathbf{z}', parent(\mathbf{z}')=\mathbf{z}} X(\mathbf{z}'). \tag{13}$$

Now, let us explain the procedure for determining $Z'(\boldsymbol{z})$. The purpose of $Z'(\boldsymbol{z})$ is to reduce the computational complexity for node size and representative determination from proportional to $\|G(\boldsymbol{z})\|^2$ to proportional to $\|Z'(\boldsymbol{z})\|^2$ per node. Here, we need a parameter n_{des} to control $\|Z'(\boldsymbol{z})\|$. We initialize $Z'(\boldsymbol{z})$ to contain only the immediate children of $\boldsymbol{z}$ and iteratively replace the largest node in $Z'(\boldsymbol{z})$ with its immediate children. This process is continued until $\|Z'(\boldsymbol{z})\| \geq n_{\text{des}}$ or all the nodes in $Z'(\boldsymbol{z})$ are leaf nodes. Overall, $\|Z'(\boldsymbol{z})\|$ satisfies the condition that

$$\|Z'(\mathbf{z})\| \leq \min\left(\|G(\mathbf{z})\|, k_{h(\max)} + n_{\text{des}}\right). \qquad (14)$$

The actual value of n_{des} used in our experiments is 32, unless specified otherwise. The complexity of this step is therefore the number of nodes times the upper bound of $\|Z'(\boldsymbol{z})\|^2$ or $O(N \cdot H \cdot (k_{h(\max)} + n_{\text{des}})^2)$, as there are no more than $2N$ nodes. The factor H results from the need to compute the Hamming distances between label vectors.

The extraction of a set of nodes from the tree is very similar to the process of extracting a partition from the dendogram in hierarchical clustering algorithms. We determine $Z(\tau)$, which is the extracted set of nodes given a threshold τ, according to

$$Z(\tau) = \{\mathbf{z} | D_r(\mathbf{z}) \leq \tau \quad \text{and} \quad D_r(parent(\mathbf{z})) > \tau\}. \quad (15)$$

It is those nodes in $Z(\tau)$ that are used to build the coassociation matrix. We also define $N_Z(\tau) = \|Z(\tau)\|$. Let us index the nodes in $Z(\tau)$ as $\boldsymbol{z}_1(\tau), \boldsymbol{z}_2(\tau), \ldots, \boldsymbol{z}_{N_Z(\tau)}(\tau)$. A straightforward method for computing the elements of the resulting coassociation matrix, denoted as $\boldsymbol{S}^*(\tau)$, is just to use the pairwise similarities among the node representatives in a form similar to (4)

$$s^*_{ij}(\tau) = 1 - \frac{1}{H} d\left[\boldsymbol{\lambda}_R(\mathbf{z}_i(\tau)), \boldsymbol{\lambda}_R(\mathbf{z}_j(\tau))\right]. \qquad (16)$$

In Fig. 3(a)–(d), we show the partitions of X obtained by thresholding the tree in Fig. 2(b) with τ being 0, 1, 2, and 3, respectively. The values of $N_Z(\tau)$ is 9, 5, 4, and 3, respectively. It is interesting to see that the partition in Fig. 3(c) is completely consistent with the ground truth.

D. Node Reduction

One interesting observation in Fig. 3(a) is that several of the nine core groups contain very few patterns. These core groups are generally located at low-density regions or regions between actual clusters in the feature space. On the other hand, the core groups in high-density regions are more likely to contain more patterns. Here, the six largest core groups out of nine contain 94% of the patterns. Similar phenomena also occur for the nodes extracted at different τ's (i.e., not just the core groups) after more experiments with larger data sets; the results are included in the next section.

Based on this observation, we believe it is possible to further reduce $N_Z(\tau)$ by keeping only the important nodes (i.e., nodes that contain substantial numbers of patterns) for building the coassociation matrix. Instead of specifying the number of nodes to keep, we define a parameter $\gamma (0 < \gamma \leq 1)$ such that we retain enough nodes, on the order of decreasing number of patterns, to include at least a total of γN patterns. For example, for the data in Fig. 3, using $\gamma = 0.8$ results in $N_Z(\tau)$ of 5, 4, 4, and 3 for τ of 0, 1, 2, and 3, respectively. The choice of γ in a particular application depends on the desired tradeoff between clustering accuracy and efficiency; we can always err on a larger γ to prevent significant degradation of the final cluster accuracy. The value 0.9 (in addition to the default of 1.0) is used in some experiments in the next section and seems to work well for all our data sets. Since now both $Z(\tau)$ and $N_Z(\tau)$ are also affected by γ, we use $Z^*(\tau)$ and $N^*_Z(\tau)$ to represent the versions without node reduction (i.e., when $\gamma = 1$) when there might be ambiguity.

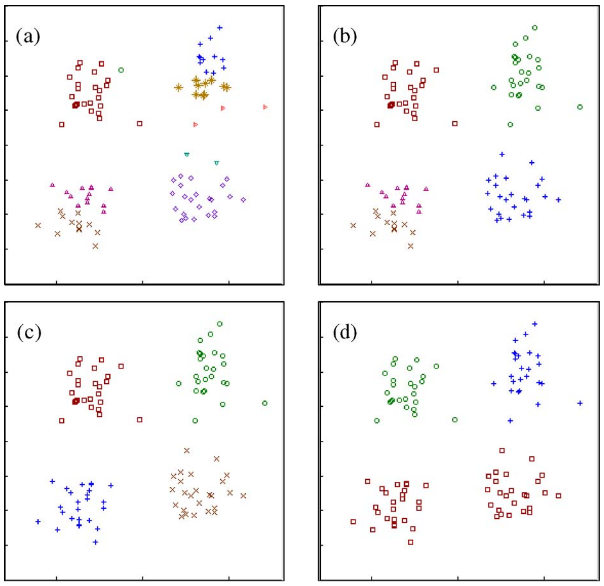


Fig. 3. (a)–(d) Partitions of the data set in Fig. 1 obtained with the threshold being 0, 1, 2, and 3, respectively.

As the final clustering is now derived from the retained nodes, a problem remains regarding how to assign the final cluster labels for the patterns in the excluded nodes. We first extract a subtree consisting of only the retained nodes and their ancestors. For each excluded node $\boldsymbol{z}$, we perform a search starting at the root of this subtree. In each step, we select the child that is most similar to $\boldsymbol{z}$. This search continues until we reach a leaf node, which is a retained node included in the final clustering. The final cluster label for this leaf node is then assigned to $\boldsymbol{z}$ as well. The complexity of this step is $O(N^*_Z(\tau) \cdot H \cdot k_{h(\max)})$. On the other hand, the complexity of selecting the nodes to retain is $O(N^*_Z(\tau) \cdot \log N^*_Z(\tau))$, as this involves sorting all the nodes in $Z(\tau)$ according to their numbers of patterns.

IV. Experimental Results

A. Experimental Settings

Although any clustering algorithm for relational data can be used to obtain the final clustering from the coassociation matrix, we focus our experiments on the framework of evidence-accumulation clustering (EAC), as described in [12]. The common k-means algorithm is used to generate the base clusterings, with the number of clusters k in each clustering randomly selected from an interval $[k_{\min}, k_{\max}]$. The actual values of $k_{\min}$ and $k_{\max}$ are data set dependent. However, we use a large range to ensure that the results are not too sensitive to the choice of an optimal k. The clustering runs are initialized using a randomly selected subset of the data points as the initial

TABLE II
SUMMARY OF DATA SETS

Data Set	N	L	k^*	$[k_{min}, k_{max}]$
Spherical5	250	16	5	[4, 10]
Half-rings	800	2	2	[8, 20]
3Rings	900	2	3	[20, 50]
8d5k [10]	1000	8	5	[4, 10]
Opt-digits	3823	64	10	[10, 40]
Pen-digits	10992	16	10	[10, 40]

prototypes. The experimental results presented here, unless otherwise noted, use EAC plus hierarchical agglomeration with average linkage (EAC-AL) for obtaining the final clustering, which is the best overall performer in [12]. The final cluster label of a pattern $\boldsymbol{x}$ is just the final cluster label assigned to the node $\boldsymbol{z}$, where $X(\boldsymbol{z})$ contains $\boldsymbol{x}$.

The quality of the final clustering is evaluated by matching the final cluster labels with the ground-truth cluster labels of the patterns. The ground truth consists of the class labels of real data sets and the cluster labels used when generating the synthetic data sets. We use the Hungarian algorithm to find the optimal assignment (the one that results in the largest number of correctly labeled patterns) between the two sets of labels. We then use the ratio of correctly labeled patterns given by this optimal assignment as the clustering accuracy measure. We use $Q(\tau)$ to represent the clustering accuracy at τ.

For the experiments described in the following, the results are always averaged over 20 ensembles. Unless noted otherwise, the user-specified parameters are $H = 20$, $n_{\text{des}} = 32$, and $\gamma = 1$ (without node reduction) or 0.9 (with node reduction).

B. Data Sets

Table II gives a summary of the data sets used in our experiments, including the intervals $[k_{\min}, k_{\max}]$ used. Here, N is the number of patterns, L is the data dimensionality (number of features), and k^* is the "natural" (ground-truth) number of clusters, taken as the number of classes for the real data sets and as the number of clusters used for generating a synthetic data set. The following provide more detailed descriptions to these data sets.

1) *Spherical5*: five touching spherical clusters with 50 patterns each in a 16-D space. The five clusters have the same size, and their centroids all fall on the plane of the first two dimensions.
2) *Half-rings*: two half rings with 200 and 600 patterns, respectively. The distribution of patterns is similar to the half-ring data set in [12].
3) *3-rings*: three concentric circles with 100, 400, and 400 patterns, respectively. The distribution of patterns is similar to the three-ring data set in [12].
4) *8d5k* [10]: This is a synthetic data set of five ellipsoidal clusters with 200 patterns each in an 8-D space. The five clusters are well separated and have identical covariance matrix.
5) *Opt-digits* (optical recognition of handwritten digits): a ten-class (each for one digit) 3823-pattern 64-D data set from the UCI Machine Learning Repository [34].
6) *Pen-digits* (pen-based recognition of handwritten digits): a ten-class (each for one digit) 10 992-pattern 16-D data set from the UCI Machine Learning Repository [34].

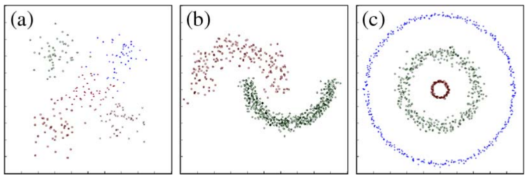


Fig. 4. Three synthetic data sets used in our experiments. (a) *Spherical5*. (b) *Half-rings*. (c) *3-rings*.

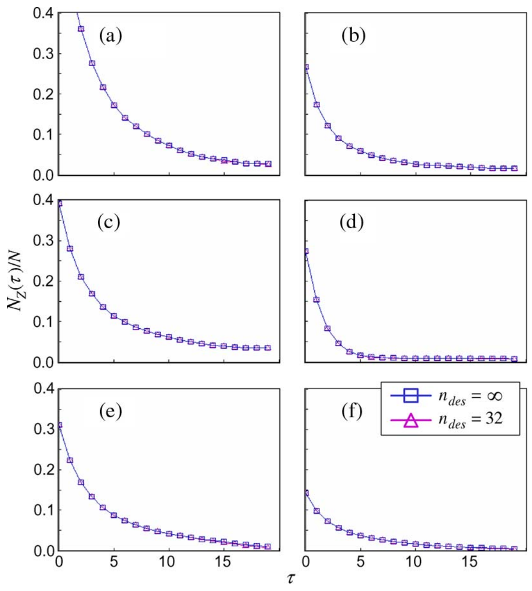


Fig. 5. Plots of the number of selected nodes (as a ratio to N) versus the threshold. Data sets: (a) *Spherical5*, (b) *Half-rings*, (c) *3-rings*, (d) *8d5k*, (e) *Opt-digits*, and (f) *Pen-digits*.

The first-three synthetic data sets (projected onto the first two dimensions if $L > 2$) are shown in Fig. 4. The last two (*Opt-digits* and *Pen-digits*) are selected because they are large for coassociation matrices. For example, the authors of [12] only use the first 100 patterns of each of the ten classes in *Opt-digits*.

C. Experiments on the Properties of CA-Trees

We start by showing in Fig. 5 how $N_Z(\tau)/N$ varies with τ for several data sets. The value of τ ranges from 0 to $H - 1$. When τ is zero, each node in $Z(\tau)$ is a core group. Therefore, $N_Z(\tau = 0)$ is the total number of core groups. We can see that $N_Z(\tau)/N$ is down to 0.1 or much lower at, for example, $\tau = H/2$, resulting in significant computational savings for the processing of the coassociation matrix.

Each plot in Fig. 5 actually has two curves corresponding to (blue curve) $n_{\text{des}} = 32$ and (magenta curve) $n_{\text{des}} = \infty$, respectively. The latter case corresponds to using the exact node sizes computed according to (5)–(7). In each plot, either the two curves are identical or they are only slightly different for larger τ. The small difference indicates that using $n_{\text{des}} = 32$ results in a good estimation of node sizes for our experimental settings. However, we want to note that a larger n_{des} may be needed if significantly more base clusterings are used, leading to a deeper tree. This is because the estimation error accumulates through the recursive processing of the tree, which is the reason why the difference is more evident at larger τ.

Fig. 5 does not tell us how to select an appropriate threshold. For this purpose, we are more interested in how the clustering

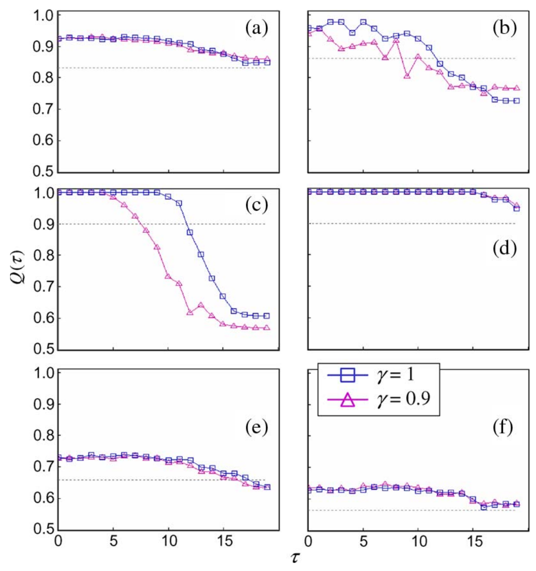


Fig. 6. Plots of clustering accuracy versus the threshold with and without node reduction. The horizontal dashed line in a plot indicates 90% of the accuracy at $\tau = 0$. Data sets: (a) *Spherical5*, (b) *Half-rings*, (c) *3-rings*, (d) *8d5k*, (e) *Opt-digits*, and (f) *Pen-digits*.

accuracy varies with τ. Fig. 6 shows the plots of $Q(\tau)$ versus τ for the same data sets as in Fig. 5. Here, we focus on the case where we select the final clustering with the known number (k^*) of clusters during hierarchical agglomeration. To help us choose an appropriate threshold, in each plot, we also show a horizontal dashed line indicating the clustering accuracy at $0.9Q(0)$, meaning a 10% degradation from the accuracy obtained by directly using the core groups. For example, a common threshold of $\tau = 10 (\tau/H = 0.5)$ is applicable for all six data sets if 10% is considered the acceptable level of degradation. To err on the conservative side, $0.4H$ and $0.2H$ seem to be reasonable choices for $\gamma = 1$ and $\gamma = 0.9$, respectively. We still get most of the benefit, as the most significant drop of $N(\tau)$ occurs at small values of τ, which is evident in Fig. 5.

A closer examination of Fig. 6 seems to indicate that there are two types of behaviors in terms of how clustering accuracy varies with τ. For four of the six data sets [Fig. 6(a), (d), (e), and (f)], the curves are quite flat, and $Q(\tau)$ only drops slightly (close to or less than 10%) even at $\tau = H - 1$. On the other hand, Fig. 6(b) and (c) seems to exhibit some critical threshold, after which $Q(\tau)$ drops dramatically. This can be explained by the fact that these two data sets have elongated curvilinear clusters, and $N_Z(\tau)$ at this "critical threshold" approximately corresponds to the number of representative points needed to preserve the cluster information in the data. $N_Z(\tau)$ values at $\tau = 10$ are 21 and 55 for data sets *Half-rings* and *3-rings*, respectively. This observation seems to suggest two different strategies for threshold selection if we have some information about the type of clusters: larger thresholds to get smaller $N_Z(\tau)$ and, hence, less computation for compact and spherical clusters, and smaller thresholds to ensure acceptable accuracy for clusters that are curvilinear or otherwise have noncompact shapes. While the knowledge about cluster shapes may not be available, the process of node reduction can provide us with some useful information on this. This is further explained later in Section IV-F.

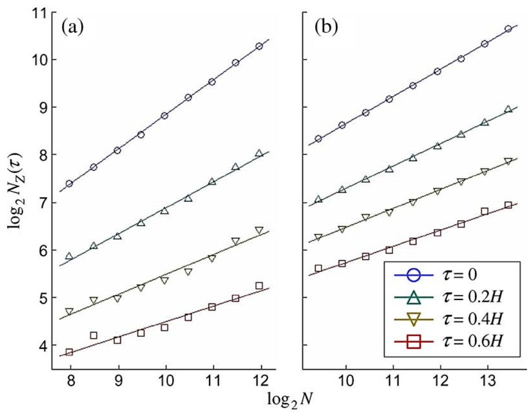


Fig. 7. Logarithmic plots of the number of selected nodes versus N at four different thresholds. Data sets: (a) *Spherical5* and (b) *Pen-digits*.

Our main goal is the reduction of complexity by using only the selected nodes to build the coassociation matrix. It is therefore important to analyze how $N_Z(\tau)$ scales with N. For this purpose, we generate additional data sets that have the same distributions as several synthetic data sets but different N's. For the three data sets from external sources (*8d5k*, *Opt-digits*, and *Pen-digits*), the additional data sets are obtained by random subsampling. Fig. 7 shows the logarithmic plots of $N_Z(\tau)$ versus N without node reduction at a few different τ's for the *Spherical5* and *Pen-digits* data sets. The nearly linear plots suggest that it is reasonable to assume that

$$N_Z(\tau) \propto N^{\alpha(\tau)} \tag{17}$$

where $\alpha(\tau)$ is data set dependent. The complexity associated with the coassociation matrix becomes $O(N^{2\alpha(\tau)})$ instead of $O(N^2)$. As long as $\alpha(\tau) < 1$, we have lower than quadratic complexity, and the cluster ensemble algorithm using our selected nodes will scale better to large data sets than the original version that uses all the patterns in the data set.

In Tables III and IV, we list more $\alpha(\tau)$ values for additional data sets using $\gamma = 1$ and $\gamma = 0.9$, respectively. It is evident that $\alpha(\tau)$ decreases fairly fast as the threshold increases. This is an indication that $N_Z(\tau)$ is more affected by N at very low thresholds and by the actual distribution at higher thresholds. At our recommended thresholds ($0.4H$ for $\gamma = 1$ and $0.2H$ for $\gamma = 0.9$), all the $\alpha(\tau)$ values are less than 0.5, resulting in the complexity associated with the coassociation matrix being below $O(N)$. A comparison between these two tables also shows that node reduction leads to substantially lower complexity at similar thresholds. Even when comparing $\tau = 0.4H$ for $\gamma = 1$ and $\tau = 0.2H$ for $\gamma = 0.9$, four of the $\alpha(\tau)$ values for $\gamma = 0.9$ are much less than the corresponding values for $\gamma = 1$, and for the other two (*3-Rings* and *8d5k*), the $\alpha(\tau)$ values are already close to zero. This is an indication that node reduction is an attractive option when we need to process very large data sets. Lower γ is expected to lead to even lower $\alpha(\tau)$ but can also result in lower clustering accuracy because fewer nodes are used to represent the data distribution.

D. Experiments With Very Large Data Sets

To further demonstrate the ability of CA-tree to make coassociation-based cluster ensembles scalable to very large

TABLE III
VALUES OF $\alpha(\tau)$ AT DIFFERENT THRESHOLDS ($\gamma = 1$)

			$\alpha(\tau)$						
Data Set	N_{min}	N_{max}	τ=0	τ= 0.1H	τ= 0.2H	τ= 0.3H	τ= 0.4H	τ= 0.5H	τ= 0.6H
Spherical5	250	4000	0.73	0.64	0.55	0.46	0.42	0.38	0.32
Half-rings	400	6400	0.40	0.23	0.18	0.17	0.16	0.16	0.12
3Rings	450	7200	0.40	0.18	0.10	0.07	0.05	0.03	0.02
8d5k	63	1000	0.63	0.42	0.18	0.08	0.03	0.01	0.01
Opt-digits	239	3823	0.63	0.50	0.42	0.38	0.35	0.32	0.28
Pen-digits	687	10992	0.57	0.52	0.47	0.43	0.40	0.37	0.34

TABLE IV
VALUES OF $\alpha(\tau)$ AT DIFFERENT THRESHOLDS ($\gamma = 0.9$)

			$\alpha(\tau)$						
Data Set	N_{min}	N_{max}	τ=0	τ= 0.1H	τ= 0.2H	τ= 0.3H	τ= 0.4H	τ= 0.5H	τ= 0.6H
Spherical5	250	4000	0.65	0.42	0.25	0.20	0.17	0.16	0.12
Half-rings	400	6400	0.22	0.12	0.09	0.06	0.06	0.06	0.04
3Rings	450	7200	0.26	0.13	0.07	0.05	0.03	0.02	0.02
8d5k	63	1000	0.51	0.28	0.13	0.08	0.04	0.02	0.02
Opt-digits	239	3823	0.54	0.26	0.07	0.00	0.00	0.00	0.01
Pen-digits	687	10992	0.28	0.14	0.10	0.09	0.09	0.09	0.08

data sets, we include here the example of clustering the pixels in an image. Fig. 8(a) shows a 640 $\times$ 480 image with a total of over 3×10^5 pixels. Using the RGB values of each pixel as a pattern to be clustered, we have a data set almost 30 times larger than the largest data set (*Pen-digits*) used in our experiments so far. We apply the EAC-AL algorithm with CA-tree using the following parameters: $H = 10$, $k_{\min} = 3$, $k_{\max} = 6$, $\tau = 0.2H$, $\gamma = 0.9$, and $k^* = 3$ for the three main regions (sky, road, and greenery). The result is shown in Fig. 8(b), where each color represents a cluster. We can see that the clustering result is quite accurate except for where the treetops meet the sky. Using 1000 manually labeled pixels randomly selected from the image, we estimate the clustering accuracy to be 0.979. This experiment clearly indicates the usefulness of CA-tree for applying cluster ensembles to very large data sets.

E. Comparisons With the Prototype-Based Method

We are interested in comparing our method with the prototype-based method in terms of how well the information of the original data set is condensed into the set of selected nodes (our method) or preclusters (prototype-based method). For the prototype-based method, three preclustering approaches are implemented for comparison: 1) random sampling, i.e., randomly selecting a subset of data points as the prototypes; 2) k-means; and 3) k-medoids. Both k-means and k-medoids are randomly initialized for each ensemble.

We base the comparison on how Q, which is the clustering accuracy, varies with N_Z. For the prototype-based methods, N_Z is the number of prototypes, and we obtain the results of these methods at several prespecified N_Z. For the CA-tree, we obtain results at different N_Z's by varying τ. Both similar clustering accuracy at smaller N_Z and higher accuracy at similar N_Z indicate a better ability for information condensation. The plots are shown in Fig. 9. Each plot contains five curves: two for CA-tree with $\gamma = 1$ and $\gamma = 0.9$, respectively, and three for

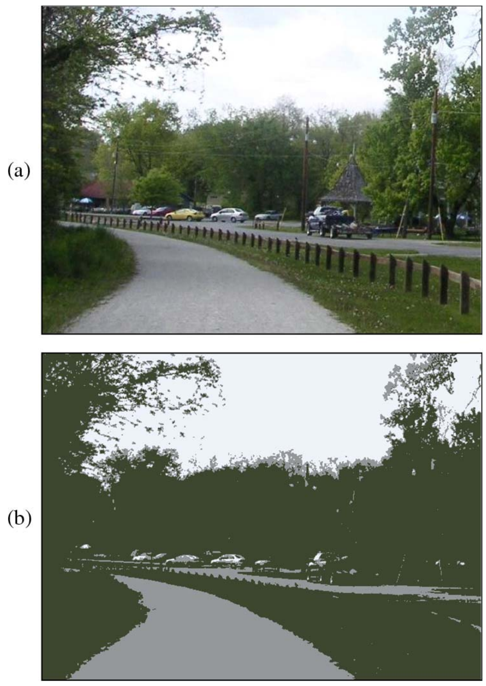


Fig. 8. Clustering of image pixels. (a) Original image. (b) Clustered pixels. Each color represents the pixels of a different cluster.

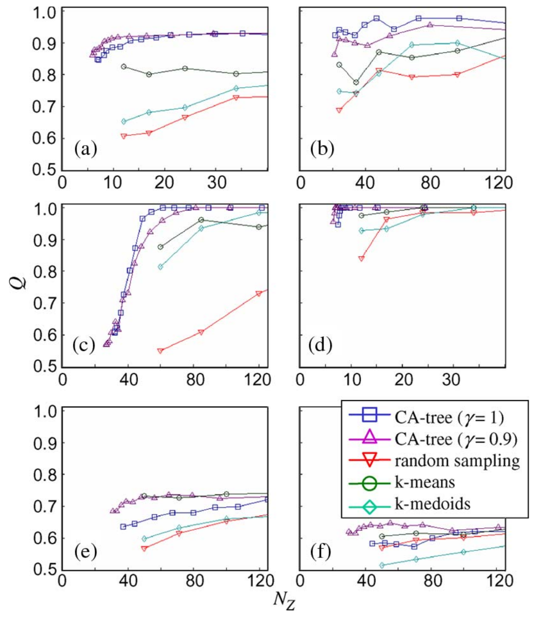


Fig. 9. Clustering accuracy versus the number of selected nodes, assuming known number of clusters. Data sets: (a) *Spherical5*, (b) *Half-rings*, (c) *3-rings*, (d) *8d5k*, (e) *Opt-digits*, and (f) *Pen-digits*.

the prototype-based methods. Our method with $\gamma = 0.9$ clearly outperforms the prototype-based methods for all the data sets except for in Fig. 9(e), where they are comparable.

Fig. 10 is similar to Fig. 9 except that the final clustering is selected according to the maximum-lifetime criterion in hierarchical agglomeration. Again, our method with $\gamma = 0.9$ is the best performer except for in Fig. 10(b) and (c), where the clustering accuracy is extremely low for all the methods.

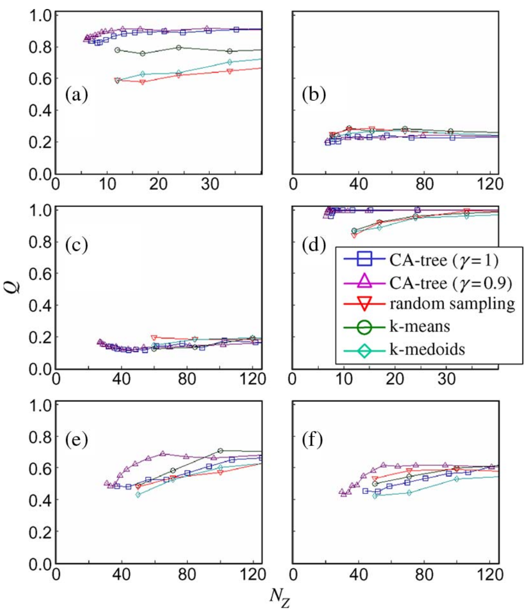


Fig. 10. Clustering accuracy versus the number of selected nodes, assuming unknown number of clusters. Data sets: (a) *Spherical5*, (b) *Half-rings*, (c) *3-rings*, (d) *8d5k*, (e) *Opt-digits*, and (f) *Pen-digits*.

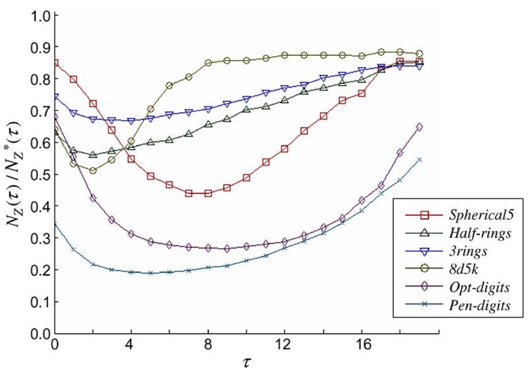


Fig. 11. Ratio of the numbers of selected nodes with and without node reduction. Each curve is for a different data set.

F. Node Reduction and Cluster Shapes

When comparing the results with and without node reduction in Figs. 9 and 10, the effect of node reduction seems to range from significantly better (*Opt-digits* and *Pen-digits*) and slightly better (*Spherical5* and *8d5k*) to somewhat worse (*Half-rings* and *3-rings*). To better understand these differences, we plot in Fig. 11 the ratios $N_Z(\tau)/N_Z^*(\tau)$. (As mentioned previously, when we need to disambiguate between the two, $N_Z(\tau)$ and $N_Z^*(\tau)$ represent the values with and without node reduction, respectively.) In general, the smaller the ratio, the stronger the information condensation is as more nodes are excluded. The minimal ratio for a particular data set seems to be a useful indicator of the effect of node reduction. These minimal ratios are 0.189 (*Pen-digits*), 0.267 (*Opt-digits*), 0.433 (*Spherical5*), 0.506 (*8d5k*), 0.557 (*Half-rings*), and 0.671 (*3-rings*). We can see that smaller minimal ratios somewhat correlate with the improvements from node reduction, although only qualitatively. As a starting point, we believe that this minimal ratio can be a useful rough guideline regarding whether to use node reduction. For example, node reduction is used when the minimal ratio is below a threshold of 0.5.

To better understand the information in the ratio $N_Z(\tau)/N_Z^*(\tau)$, let us consider the following scenarios: When we have compact actual cluster structure in the data, the regions between clusters have low data density. The "intersections" between base clusters in these regions are therefore likely to have few data points, forming small groups in the CA-tree. Such are the nodes that are excluded in node reduction. Since these groups are small, we can exclude many nodes at a given γ, giving a smaller ratio $N_Z(\tau)/N_Z^*(\tau)$. On the other hand, for elongated actual clusters, the intersections between (compact) base clusters have similar density as the regions inside the base clusters, resulting in more groups with similar data counts. Node reduction, in this case, gives larger $N_Z(\tau)/N_Z^*(\tau)$ (i.e., fewer excluded nodes) at a given γ. Overall, these arguments can help to explain our observation that the two data sets with known elongated cluster structures (*Half-rings* and *3-rings*) have the largest minimum $N_Z(\tau)/N_Z^*(\tau)$ ratios and support our idea that this ratio contains information about actual cluster shapes. We also suspect that there is more useful information regarding the underlying cluster shapes that can be deduced from the statistics of pattern counts in the nodes at different thresholds, which is a point that requires additional investigation.

V. Computational Complexity

A. Analysis of Complexity

In the following, we summarize the computational complexity involved in the various steps of our proposed method:

1) tree construction: $O(N \cdot H)$;
2) node size and representative determination: $O(N \cdot H \cdot (k_{h(\max)} + n_{des})^2)$;
3) computation of the coassociation matrix: $O(H \cdot N_Z(\tau)^2)$;
4) if node reduction is used, the selection of the retained nodes: $O(N_Z^*(\tau) \cdot \log N_Z^*(\tau))$;
5) if node reduction is used, the final cluster label assignment for the excluded nodes: $O(N_Z^*(\tau) \cdot H \cdot k_{h(\max)})$.

Here, we do not include the complexity involved in generating the base clusterings and extracting the final clustering, as there are many different algorithms applicable to these two steps and their differences are not our focus.

The combined expression for the complexity is

$$O\Big(N \cdot H \cdot (k_{\max} + n_{des})^2 + N^{2\alpha(\tau)} \cdot H + N^{\alpha*}(\tau) \cdot (\log N + k_{\max})\Big) \quad (18)$$

after substituting (17) for $N_Z(\tau)$ and an analogous expression

$$N_Z^*(\tau) \propto N^{\alpha*(\tau)} \quad (19)$$

for $N_Z^*(\tau)$. We have also used the user-specified parameter $k_{\max}$ as the upper bound for $k_{h(\max)}$. Since we expect that $\alpha^*(\tau) < 1$, the third term in (18) is always less than the first term and can be dropped. We end up with the following:

$$O\left(N \cdot H \cdot (k_{\max} + n_{\text{des}})^2 + N^{2\alpha(\tau)} \cdot H\right). \quad (20)$$

If we have $2\alpha(\tau) \leq 1$, which is the case in our experiments except for very low thresholds (see Table IV), then the overall complexity is linear with respect to N. Even when this condition is not satisfied, the savings over the original quadratic complexity is still very significant for large N. In addition, this lower complexity is even more important when the algorithm for the final clustering extraction has a complexity that is higher than quadratic. Such examples include average or complete linkage, both having $O(N^2 \log N)$ complexity using the existing efficient hierarchical agglomeration algorithms [35], [36].

Regarding space complexity, the coassociation matrix now has $N^{2\alpha(\tau)}$ elements. The tree itself has at most $2N$ nodes. Similar to time complexity, the space complexity with respect to N is $O(N)$ when $2\alpha(\tau) \leq 1$ and $O(N^{2\alpha(\tau)})$ if otherwise.

B. Experiments on Computation Time

This section consists of experimental results regarding the actual computation time spent in constructing a cluster ensemble. The purpose is to provide empirical evidences to the theoretical claim of reduced computational complexity in the previous section. The experimental environment is MATLAB R14 running on a 3.0-GHz Pentium IV PC with 1 GB of memory, while the majority of the time-consuming work is implemented in C. The timing data are obtained with the MATLAB Profiler.

Specifically, we compare the required execution time when the data structure for the ensemble (such as the coassociation matrix) is built using one of the following [referred to as methods (A)–(D) in this section]:

1) (A): the original data set;
2) (B): the core groups, corresponding to CA-tree with $\tau = 0$;
3) (C): the groups selected from the CA-tree with $\tau = 0.2$;
4) (D): the same as (C) plus node reduction ($\gamma = 0.9$).

The experimental settings are the same as those in Section IV-C, except that we use $H = 10$. Two synthetic data sets *Spherical5* and *3-rings* are used here. However, to emphasize the performance comparison for larger data sets, we increase the numbers of samples in these two data sets to 2000 and 1800, respectively, while maintaining the same distributions as the original data sets.

Tables V lists the execution time (in seconds) and clustering accuracy averaged over 20 ensembles. Here, t_{total} represents the total execution time, including the time spent on obtaining the base clusterings. On the other hand, t_{CE} includes only the time for extracting the final clustering from available base cluster labels. While our focus is on the improvement of t_{CE}, we list t_{total} as well to provide the information about the total time requirement when the base clusterings are not available.

In addition to EAC-AL, we also list the results obtained with three well-known graph-based cluster ensemble algorithms: HBGF [26], CSPA [21], and MCLA [21]. We use METIS [37] as the method that partitions the graph into final clusters. Among the three, only CSPA is based on coassociation matrices. The computational complexities of HBGF and MCLA are both linear with respect to N. The motivation here is to see whether and how much performance improvement is achievable by using CA-tree with such methods.

TABLE V
COMPARISON OF EXECUTION TIME AND CLUSTERING ACCURACY

Data set	*Spherical5* (N = 2000)			*3rings* (N = 1800)		
Method	t_{total}	t_{CE}	Q	t_{total}	t_{CE}	Q
EAC-AL(A)	111.54	111.29	0.95	82.29	82.02	1.00
EAC-AL(B)	1.19	0.94	0.95	0.79	0.54	1.00
EAC-AL(C)	0.36	0.11	0.95	0.35	0.10	1.00
EAC-AL(D)	0.31	0.06	0.94	0.34	0.08	0.99
CSPA(A)	7.76	7.51	0.95	4.14	3.88	0.74
CSPA(B)	0.68	0.43	0.95	0.54	0.27	0.74
CSPA(C)	0.40	0.16	0.94	0.38	0.13	0.76
CSPA(D)	0.37	0.12	0.92	0.38	0.13	0.75
HBGF(A)	0.51	0.26	0.95	0.51	0.26	0.78
HBGF(B)	0.43	0.18	0.95	0.44	0.19	0.77
HBGF(C)	0.38	0.14	0.95	0.40	0.15	0.77
HBGF(D)	0.38	0.13	0.93	0.41	0.15	0.78
MCLA(A)	0.46	0.22	0.95	2.93	2.68	0.77
MCLA(B)	0.41	0.16	0.95	1.27	1.02	0.77
MCLA(C)	0.39	0.14	0.95	1.02	0.77	0.76
MCLA(D)	0.39	0.15	0.94	0.97	0.72	0.76

Note: (A), (B), (C) and (D) represent different settings for the CA-tree in building the co-association matrix; please refer to the text for more detail.

The improvement in execution time is most significant with coassociation-based methods (EAC-AL and CSPA), as expected. For HBGF and MCLA, t_{CE} is also reduced by about 60%–75% from methods (A) to (D), with little loss of clustering accuracy. This indicates that CA-tree is still useful in improving the efficiency and scalability of cluster ensemble algorithms that are not based on coassociation matrices. The time used to obtain the base clusterings (approximately $t_{\text{total}} - t_{CE}$) stays approximately constant for all the methods, which is not surprising. For methods (C) and (D), we have reduced t_{CE} so much that the time for base clusterings mostly dominates t_{total}. We can also see that the improvement in execution time obtained with node reduction [from (C) to (D)] is not as significant as the improvement obtained with the CA-tree itself [from (A) to (C)]. This is because the time spent on processing node reduction is not negligible compared with clustering the already small set of groups from method (C). However, the improvement from node reduction is more significant for larger data sets. For example, if *Spherical5* is scaled to $N = 10\,000$, the t_{CE} values for EAC-AL become 0.61 and 0.09 for methods (C) and (D), respectively, a factor of about 6.7 compared to about 1.9 (0.11/0.06) when $N = 2000$.

One more important observation can be made about the results for *3-rings* (and more generally, data sets with noncompact clusters). We see that the clustering accuracies obtained with the three graph-based algorithms range from 0.74 to 0.78 and are far below the near perfect accuracy achieved by EAC-AL. This is previously known in [12] and is a major motivation behind using the inefficient hierarchical agglomeration algorithms to get the final clusterings. However, when considering methods (C) and (D), EAC-AL outperforms the three graph-based algorithms in *both* execution time and accuracy. In other words, CA-tree significantly alleviates the problem of computational cost for algorithms such as EAC-AL, allowing us to use the most appropriate cluster ensemble algorithms on different data sets without much worry about efficiency.

VI. Conclusion

In summary, in this paper, we have proposed the CA-tree, which is a new data structure for improving the scalability of ensemble-clustering algorithms for large data sets, particularly those based on coassociation matrices. When cut at a particular threshold for node sizes, a set of nodes is selected to approximate the data set in the space of base cluster labels. Subsequent ensemble-clustering algorithms are applied to this set of selected nodes instead of the original patterns, leading to reduced computational complexity. A node reduction scheme has also been proposed to further reduce the amount of computation by excluding nodes that contain few patterns. As demonstrated by our experiments with various data sets, we can often achieve a complexity of lower than linear to N for coassociation matrix processing with little loss in clustering accuracy, and the complexity for building the tree is linear to N. This improvement of efficiency is dramatic compared to the $O(N^2)$ or higher complexity of popular coassociation-based algorithms such as EAC and CSPA, therefore significantly extending their applicability to more clustering problems.

The CA-tree has several advantages over the prototype-based method that was previously employed to help solve the scalability problem. In addition to the overall better clustering accuracy, our method does not require the guesswork involved in determining the number of prototypes to use. Because the only information required for building the CA-tree is the base cluster labels, it is applicable to distributed multiview clustering, when a data set does not have all the features, and is able to incorporate existing base clusterings. These are two well-known ensemble-clustering scenarios that the prototype-based method is unable to address.

One important element of our method, which is the power-law relation between N and $N_Z(\tau)$ [see (17)], is currently supported by empirical results. In the future, we intend to provide a deeper understanding of such a relation from theoretical perspectives, as well as to investigate factors that affect $\alpha(\tau)$. Moreover, of theoretical interest is the information regarding the properties of the actual clusters that may exist in the statistics of pattern counts in the nodes at various thresholds, such as to tell whether the clusters are elongated or compact. While current results seem to hint at the existence of such information (see Section IV-F), more study, both experimental and theoretical, will be needed to provide a clearer picture. Another possible subject for future study is the extension of our method to fuzzy cluster ensembles, which is a subject that has attracted increasing interests in recent years.

The software implementing the CA-tree will be made available at the home page of the author: http://www.cis.nctu.edu.tw/~wangts/.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[2] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd ed. San Diego, CA: Academic, 2006.

[3] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[4] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[5] T. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop Multiple Classifier Syst.*, 2000, pp. 1–15.

[6] L. Lam, "Classifier combinations: Implementations and theoretical issues," in *Proc. 1st Int. Workshop Multiple Classifier Syst.*, 2000, pp. 78–86.

[7] A. Topchy, A. K. Jain, and W. Punch, "A mixture model for clustering ensembles," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 379–390.

[8] A. K. Jain and M. H. C. Law, "Data clustering: A user's dilemma," in *Pattern Recognition and Machine Intelligence*. Berlin, Germany: Springer-Verlag, 2005, pp. 1–10.

[9] P. Kellam, X. Liu, N. J. Martin, C. Orengo, S. Swift, and A. Tucker, "Comparing, contrasting and combining clusters in viral gene expression data," in *Proc. 6th Workshop Intell. Data Anal. Med. Pharmocol.*, 2001, pp. 56–62.

[10] K. Punera and J. Ghosh, "Soft cluster ensembles," in *Advances in Fuzzy Clustering and Its Applications*, J. Valente de Oliveira and W. Pedrycz, Eds. Hoboken, NJ: Wiley, 2007.

[11] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional clustering: A cluster ensemble approach," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 186–193.

[12] A. L. N. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.

[13] L. I. Kuncheva and D. P. Vetrov, "Evaluation of stability of k-means cluster ensembles with respect to random initialization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1798–1808, Nov. 2006.

[14] R. Avogadri and G. Valentini, "Ensemble clustering with a fuzzy approach," in *Supervised and Unsupervised Ensemble Methods and Their Applications*. New York: Springer-Verlag, 2008, pp. 49–69.

[15] P. Viswanath and K. Jayasurya, "A fast and efficient ensemble clustering method," in *Proc. Int. Conf. Pattern Recog.*, 2006, pp. 720–723.

[16] T.-Y. Lv, S.-B. Huang, X.-Z. Zhang, and Z.-X. Wang, "Combining multiple clustering methods based on core group," in *Proc. 2nd Int. Conf. Semantics, Knowl., Grid*, 2006, pp. 29–34.

[17] B. Minaei-Bidgoli, A. Topchy, and W. F. Punch, "Ensembles of partitions via data resampling," in *Proc. Int. Conf. Inf. Technol.*, 2004, pp. 188–192.

[18] L. I. Kuncheva and S. T. Hadjitodorov, "Using diversity in cluster ensembles," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2004, vol. 2, pp. 1214–1219.

[19] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.

[20] P. Hore, L. Hall, and D. Goldgof, "A cluster ensemble framework for large data sets," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2006, pp. 3342–3347.

[21] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, Mar. 2002.

[22] B. Fischer and J. M. Buhmann, "Bagging for path-based clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 11, pp. 1411–1415, Nov. 2003.

[23] S. Dudoit and J. Fridlyand, "Bagging to improve the accuracy of a clustering procedure," *Bioinformatics*, vol. 19, no. 9, pp. 1090–1099, Jun. 2003.

[24] H. G. Ayad and M. S. Kamel, "Cumulative voting consensus method for partitions with a variable number of clusters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 160–173, Jan. 2008.

[25] H. Luo, F. Kong, and Y. Li, "Combining multiple clusterings via k-modes algorithm," in *Advanced Data Mining and Applications*. Berlin, Germany: Springer-Verlag, 2006, pp. 308–315.

[26] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Proc. 21th Int. Conf. Mach. Learn.*, vol. 69, *ACM International Conference Proceeding Series*, 2004, p. 36.

[27] A. L. N. Fred and A. K. Jain, "Data clustering using evidence accumulation," in *Proc. 16th Int. Conf. Pattern Recog.*, 2002, pp. 276–280.

[28] Z. Yu, Z. Deng, H.-S. Wong, and X. Wang, "Fuzzy cluster ensemble and its application on 3D head model classification," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 569–576.

[29] J. Gllavata, E. Qeli, and B. Freisleben, "Detecting text in videos using fuzzy clustering ensembles," in *Proc. 8th IEEE Int. Symp. Multimed.*, 2006, pp. 283–290.

[30] A. Lourenco and A. Fred, "Ensemble methods in the clustering of string patterns," in *Proc. 7th IEEE Workshop Appl. Comput. Vis.*, 2005, vol. 1, pp. 143–148.

[31] R. Avogadri and G. Valentini, "Fuzzy ensemble clustering for DNA microarray data analysis," in *Applications of Fuzzy Sets Theory*. Berlin, Germany: Springer-Verlag, 2007, pp. 537–543.

[32] H. Luo, F. Kong, and Y. Li, "Clustering mixed data based on evidence accumulation," in *Advanced Data Mining and Application*. Berlin, Germany: Springer-Verlag, 2006, pp. 348–355.

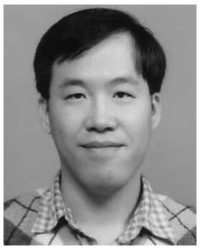
[33] D. Greene and P. Cunningham, "Efficient Ensemble Methods for Document Clustering," Trinity College, Dublin, Ireland, Tech. Rep. TCD-CS-2006-48, 2006.

[34] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*, Irvine, CA, Univ. California, 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[35] T. Kurita, "An efficient agglomerative clustering algorithm using a heap," *Pattern Recognit.*, vol. 24, no. 3, pp. 205–209, 1991.

[36] Y. El-Sonbaty and M. A. Ismail, "On-line hierarchical clustering," *Pattern Recognit. Lett.*, vol. 19, no. 14, pp. 1285–1291, Dec. 1998.

[37] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Aug. 1998.



Tsaipei Wang (M'05) received the B.S. degree in physics from National Tsing Hua University, Hsinchu, Taiwan, in 1989, the Ph.D. degree in physics from the University of Oregon, Eugene, in 1999, and the M.S. degree in computer science and the Ph.D. degree in computer engineering and computer science from the University of Missouri, Columbia, in 2002 and 2005, respectively.

He was a Postdoctoral Fellow with the University of Missouri. He is currently an Assistant Professor with the Department of Computer Science, National Chiao Tung University, Hsinchu. He was previously involved in research topics including frequency-selective optical memory, coherent transient phenomena, signal/image processing for land-mine detection, and automated vision development assessment with video photoscreening. His current research interests include fuzzy systems, pattern recognition, image processing, medical image analysis, and visualization.