Efficient Clustering Aggregation Based on Data Fragments

Ou Wu, *Member, IEEE*, Weiming Hu, *Senior Member, IEEE*, Stephen J. Maybank, *Senior Member, IEEE*, Mingliang Zhu, and Bing Li

***Abstract*—Clustering aggregation, known as clustering ensembles, has emerged as a powerful technique for combining different clustering results to obtain a single better clustering. Existing clustering aggregation algorithms are applied directly to data points, in what is referred to as the point-based approach. The algorithms are inefficient if the number of data points is large. We define an efficient approach for clustering aggregation based on data fragments. In this fragment-based approach, a data fragment is any subset of the data that is not split by any of the clustering results. To establish the theoretical bases of the proposed approach, we prove that clustering aggregation can be performed directly on data fragments under two widely used goodness measures for clustering aggregation taken from the literature. Three new clustering aggregation algorithms are described. The experimental results obtained using several public data sets show that the new algorithms have lower computational complexity than three well-known existing point-based clustering aggregation algorithms (Agglomerative, Furthest, and LocalSearch); nevertheless, the new algorithms do not sacrifice the accuracy.**

***Index Terms*—Clustering aggregation, comparison measure, computational complexity, data fragment, fragment-based approach, mutual information, point-based approach.**

I. Introduction

LET $X = \{x_1, \ldots, x_i, \ldots, x_N\}$ be a set of data points or vectors. The aim of data clustering is to partition X into disjoint subsets called clusters. There is a large number of data clustering algorithms proposed in the literature. A fixed set X may be partitioned or split in many different ways, depending on the choice of clustering algorithm. It is difficult to say which partition is the best without any further information. Fortunately, clustering aggregation offers the possibility of obtaining a better clustering by combining a number of given clusterings of X. Previous studies have shown that clustering aggregation can produce enhanced results [30].

Manuscript received January 1, 2011; revised July 17, 2011, October 14, 2011, and December 15, 2011; accepted December 26, 2011. This work is partly supported by NSFC (Grant No. 60825204, 60672040, 60723005, 61003115). This paper was recommended by Associate Editor E. Santos, Jr.

O. Wu, W. Hu, M. Zhu, and B. Li are with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: wuou@nlpr.ia.ac.cn; wmhu@nlpr.ia.ac.cn; mlzhu@nlpr.ia.ac.cn; bli@nlpr.ia.ac.cn).

S. J. Maybank is with School of Computer Science and Information Systems, Birkbeck College, University of London, WC1E 7HX London, U.K. (e-mail: sjmaybank@dcs.bbk.ac.uk).

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.2012.2183591

In its most general form, a clustering algorithm is precisely an algorithm that partitions X. Therefore, clustering aggregation is not only used in aggregating different clustering results but also can be applied in many different disciplines such as machine learning [31], [35], pattern recognition [23], bioinformatics [32], and information retrieval [33]. Some typical application settings include: categorical data clustering, heterogeneous data clustering, feature selection, outlier detection, distributed clustering, knowledge reuse, and image segmentation [9], [11], [23], [29].

A. Related Work

Given a fixed data set X, clustering aggregation involves two critical steps: 1) the creation of several different clusterings of X, and 2) aggregating these clusterings to obtain a new clustering. The new clustering is referred to as a "consensus clustering" in this paper. To obtain the different clusterings, approaches similar to those used in classifier combination can be applied. These approaches include tuning the parameters of a clustering algorithm and feature bagging. The most widely used approach for aggregating different clusterings into a single better one is to define a measure of the "goodness" of a candidate clustering, and then search for the optimal (or near-optimal) clustering via heuristic or mathematical optimization algorithms. The two widely used goodness measures are comparison measure [9] and mutual information measure [16]. The comparison measure calculates the disagreement between two clusterings. The optimal clustering has the minimum average disagreement from the input clusterings; on the contrary, the mutual information measure calculates the agreement between two clusterings. Naturally, under this goodness measure, an optimal solution has maximum average mutual information with the input clusterings.

The goodness measures used for clustering aggregation are usually discrete and difficult to optimize directly. Hence, existing methods usually take heuristic or approximate optimization strategies, which can be divided into four categories: voting based, graph based, search based, and mixture model based.

- Voting-based methods. Each input clustering in effect votes whether two given data points should be in the same cluster. Nguyen *et al.* [13] propose an expectation maximization-like iterative voting method. In each iteration, new cluster centers are selected based on the votes. The cluster label of each data point is updated according to the new cluster centers. The computational complexity is not analyzed in [13]. Since the algorithm requires

calculating the pair-wise distances between points, the computational complexity is no smaller than $O(N^2)$. Fred and Jain [7] propose an "evidence accumulation" clustering (EAC) ensemble method. In EAC, the voting information is mapped into a co-association matrix in which entries are viewed as votes on pair-wise co-occurrences in the same clusters. Linkage-based clustering algorithms are applied to the co-association matrix to find consensus clustering. As the algorithm requires generating the k-nearest neighbors for each point, the computational complexity is higher than $O(N^2)$.

- Graph-based methods. Strehl and Ghosh [16] present the three graph-based clustering aggregation algorithms: the cluster-based similarity partitioning algorithm, the hypergraph partitioning algorithm, and the metaclustering algorithm. Each algorithm constructs a graph from the initial set of clustering results, and then applies graph-clustering techniques to obtain a consensus clustering. As these three methods require the construction of an edge between each pair of points, all their complexities are at least $O(N^2)$. Fern *et al.* [5], [6] establish a bipartite graph on the input clustering solutions and then utilize graph partition methods to obtain an optimal consensus clustering. The bipartite graph models both points and clusters as vertices, and considers both the similarities between points and the similarities between clusters.
- Search-based methods. Some of the representative methods are Agglomerative, Furthest, and LocalSearch [9]. The goal in each search step is to improve the current solution according to a chosen goodness measure. For example, the Agglomerative method initially randomly splits the original data sets into several clusters and then exchanges the data points among clusters based on a heuristic rule. Other methods use evolutionary computing approaches such as the genetic algorithm [4].
- Mixture model-based methods. A parameterized generative probabilistic model is fitted to the data points, and the consensus clustering is deduced from the parameter values. Topchy *et al.* [18] propose a probabilistic mixture model for candidate consensus clustering based on the mixture multivariable distribution in the space of cluster labels. The parameter estimation is time consuming.

Some researchers focus on weighted clustering aggregation [2], [11], [14], [22], to favor some input clusterings over the others. Other researchers [1], [24] focus on a theoretical investigation of the conditions under which clustering aggregation generates better consensus clustering.

Classifier combination (or classifier ensembles) [10], [26], [27] is closely related to clustering aggregation. The main difference between clustering aggregation and classifier combination is that there is no supervised information in the former, while there is supervised information in the latter. The time complexity of clustering aggregation is highly sensitive to the data size. Many existing clustering aggregation algorithms have a time complexity quadratic, cubic, or even exponential in the number of data points N[44]. For example, the proposed algorithm in [44] has a complexity of $O(N^2 + N^3)$. When N is large, these algorithms are unsuitable for practical use.

B. Notion of Fragments

This paper is concerned with the time complexity of existing clustering aggregation algorithms. A clustering of a set X of data points is a partition of X, that is, a division of X into disjoint subsets such that the union of the disjoint subsets is X. Suppose that a set of partitions of X is given. A subset of X is called a data fragment if it is not divided by any of the given partitions of X, and meanwhile it is not contained in any other subsets which are also not divided by any of the given partitions of X. For simplicity, a data fragment may also be referred to as a fragment. The original set X can be seen as a set of data fragments instead of data points. In many clustering aggregation problems, the number of data fragments is much less than the number of data points. An interesting question is: Can we perform clustering aggregation using the data fragments rather than the individual points of X, while maintaining similar accuracies? If the answer is "Yes," the computational complexity of clustering aggregation based on data fragments is very likely to be much lower than that of clustering aggregation based on data points. We show that the answer is indeed yes for two widely used goodness measures: comparison measure and normalized mutual information (NMI) measure. A fragment-based clustering aggregation approach is introduced. We generate three new clustering aggregation algorithms as examples to show how to embed three existing clustering aggregation algorithms into the fragment-based approach. Experimental results demonstrate that the resulting fragment-based clustering aggregation algorithms are more efficient than the original point-based algorithms and also achieve similar or better results.

The remainder of this paper is organized as follows. Section II describes the data fragments. Section III contains our main theory and the corresponding proofs on the feasibility of data fragment-based clustering aggregation. Section IV introduces the fragment-based clustering aggregation approach and provides three examples of converting three existing point-based clustering aggregation algorithms into three new fragment-based algorithms. Section V reports our experiments on several public data sets. Conclusions are given in Section VI.

II. Data Fragment in Clustering Aggregation

A. Definition and Extraction

This subsection describes the notion of "data fragment" and gives an illustrative example of data fragments. Then, the computational complexity of fragment extraction is analyzed. Some notations used in the paper are defined in Table I.

Given a data set X and a set $\prod$ of H partitions, each partition divides X into a set of clusters $(\pi_{ij}, 1 \leq j \leq Pi)$. The labels for each data point x_j given by the input partitions $\prod$ form a ***label sequence***. The label sequence can be written as "$\pi_1(x_j)\pi_2(x_j)\ldots\pi_H(x_j)$." We take the data in Fig. 1 as an illustrative example. Fig. 1(a)–(c) shows three input partitions of a data set. Each partition divides the data set into three disjoint subsets (clusters). Now, we take the data points in the top and left corners as an example to show how to get the label sequence. We assume that the three clusters by lines in Fig. 1(a) are labeled as "1," "2," and "3" clockwise from top

TABLE I
NOTATIONS USED IN THE PAPER

Notation	Description
X	A data set $X = \{x_1, \cdots, x_i, \cdots, x_N\}$, where x_i represents a data point or a vector.
π_i	A partition, $\pi_i = \{\pi_{i1}, \cdots, \pi_{ij}, \cdots, \pi_{iPi}\}$, where π_{ij} is the j-th cluster in π_i. π_{ij} are disjoint for $1 \leq j \leq Pi$
$\prod$	A set of partitions $\prod = \{\pi_1, \cdots, \pi_i, \cdots, \pi_{\mathrm{H}}\}$
$\pi_i(x_k)$	The index of the cluster (in π_i) which contains x_k. Thus $\pi_i(x_k)=j$ means $x_k \in \pi_{ij}$
Z	The number of data fragments
$\mathbb{F}$	A set of data fragments $\mathbb{F} = \{F_1, \cdots, F_\zeta, \cdots, F_{\mathrm{Z}}\}$
$\lvert \cdot \rvert$	The cardinality of a set.

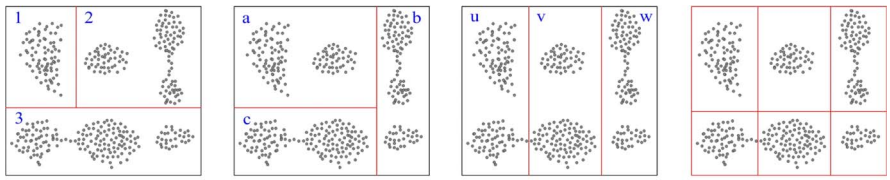


Fig. 1. (a), (b), and (c) are three input partitions; (d) shows the union of all the three solutions. Each data subset, enclosed by red lines and red borders in (d), is a data fragment. The number of fragments is 6.

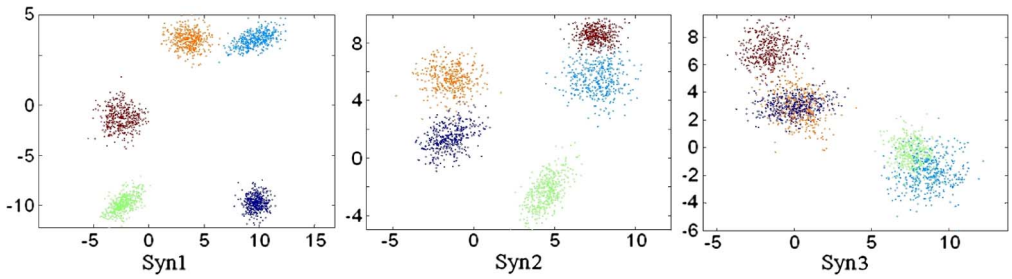


Fig. 2. Three synthetic data sets (Syn1, Syn2, and Syn3).

left; the three clusters by lines in Fig. 1(b) are labeled as "a," "b," and "c" clockwise from top left; the three clusters by lines in Fig. 1(c) are labeled as "u," "v," and "w" from left to right. Then, the label sequence of data points in the top and left corners is "1au." These data form a data fragment shown in Fig. 1(d) (the top left data subset). A data fragment is a data subset in which each point has the same label sequence. A fragment is either completely contained in a given subset belonging to one of the partitions or it is disjoint from the given subset. Let Z be the number of different label sequences. The fragments can be represented as $\mathbb{F} = \{F_1, \ldots, F_\zeta, \ldots, F_{\mathrm{Z}}\}$, where Z denotes the number of fragments. Fig. 1 has approximately 500 data points. Fig. 1(d) shows the fragments enclosed by red lines and red borders. There are six fragments, which is far fewer than the number of data points. If clustering aggregation could be based on the six fragments instead of the 500 or so points, then the computational complexity would be greatly reduced.

The fragment extraction can be illustrated by the following simple example. Assuming that there is a simple data set $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ and there are three input partitions $\pi_1 = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}, \{x_6, x_7, x_8\}\}$, $\pi_2 = \{\{x_1, x_2\}, \{x_3, x_4, x_5\}, \{x_6, x_7, x_8\}\}$, and $\pi_3 = \{\{x_1, x_2\}, \{x_3, x_6\}, \{x_4, x_5\}, \{x_7, x_8\}\}$. The three clusters in π_1 are labeled as "1," "2," and "3." The three clusters in π_2 are labeled as "a," "b," and "c." The four clusters in π_3 are labeled as "r," "s," "t," and "u." When extracting the fragments, we first construct the label sequences of all the eight points and their corresponding label sequences are: "1ar," "1ar," "1bs," "2bt," "2bt," "3cs," "3cu," and "3cu." As a result, x_1 and x_2 are put together as one fragment, x_4 and x_5 are put together as one fragment, x_7 and x_8 are put together as one fragment. x_3 and x_6 are taken as two fragments, respectively. The obtained fragment set is $\mathbb{F} = \{\{x_1, x_2\}, \{x_3\}, \{x_4, x_5\}, \{x_6\}, \{x_7, x_8\}\}$. The computational complexity of the set $\mathbb{F}$ is estimated. We obtain each point's label sequence by accessing each point once. Therefore, the time taken to label the points is $O(NH)(N = |X|, H = |\Pi|)$. The next step is to put together all the points with the same label sequences as one fragment. The time taken for this step is $O(N \log Z)(Z = |\mathbb{F}|)$. Thus, the total time complexity is $O(NH) + O(N \log Z)$.

TABLE II
DESCRIPTION OF THE EXPERIMENTAL DATA

Dataset	Data size	No. of features	No. of classes (clusters)
Syn1	2000	2	5
Syn2	2000	2	5
Syn3	2000	2	5
Yeast	1,484	8	10
Wave	5,000	21	3

B. Empirical View of the Number of Data Fragments Z

This subsection aims to provide an empirical observation of the number of fragments Z. The values of Z are observed on three synthetic data sets (called Syn1, Syn2, and Syn3) and two real data sets (called Yeast and Wave). The three synthetic data sets are shown in Fig. 2. The two real data sets are downloaded from UCI Repository [43]. The five data sets are briefly introduced in Table II.

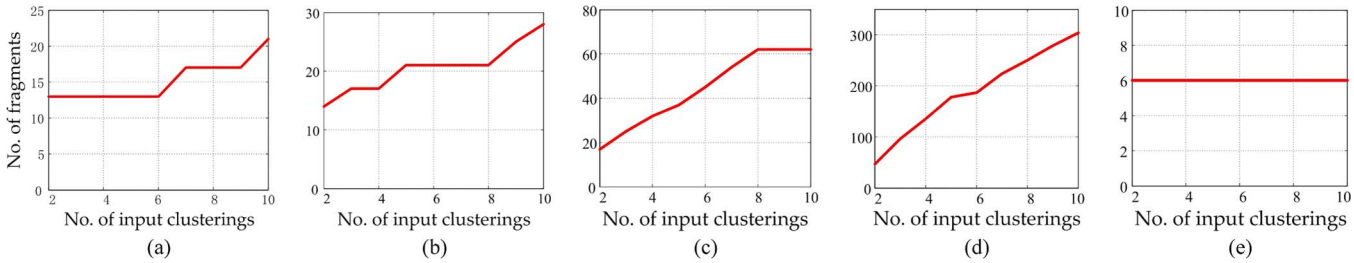


Fig. 3. Variations of the number of fragments (Z) with increasing input clusterings while on each data set the input clusterings have equal cluster counts. (a) Syn1; (b) Syn2; (c) Syn3; (d) yeast; (e) wave.

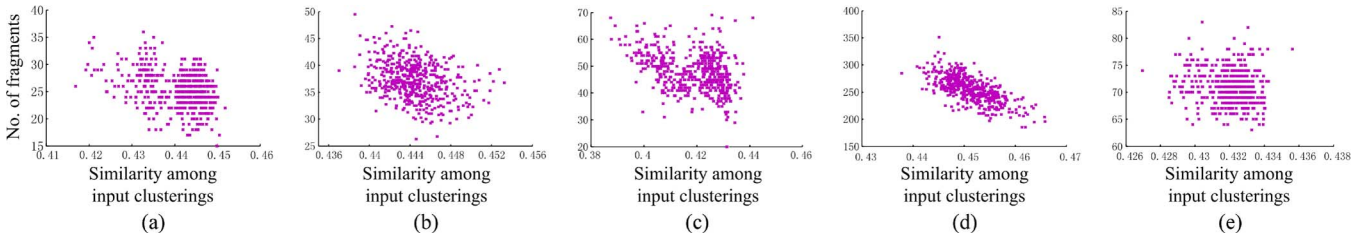


Fig. 4. Variations of the number of fragments (Z) with different similarities among input clusterings. (a) Syn1; (b) Syn2; (c) Syn3; (d) yeast; (e) wave.

Fig. 3 shows the variations of Z over the increasing of input clusterings on the five data sets. In Fig. 3, each input clustering is obtained by K-means with different initializations yet identical initial numbers of clusters (They equal the numbers of classes of the corresponding data sets). In each subfigure of Fig. 3 [e.g., Fig. 3(a)], the numbers of input clusterings are increased from left to right. Based on the experimental results in Fig. 3, two observations are obtained.

1) Z partially depends on the number of input clusterings. As shown in Fig. 3, the more the input clusterings, the larger the values of Z.
2) Z partially depends on the distribution of original data. For the three synthetic data sets Syn1, Syn2, and Syn3, the densities of the five clusters are increased, while the values of Z are also increased. Here, we further provide a quantitative analysis between the distribution and Z. In statistics, the dispersion degree is a normalized measure of the dispersion of a probability distribution [46], which can be estimated using the ratio of the standard deviation to the mean of the data [47]. The dispersion degrees of the five data sets are 3.00 (Syn1), 2.37 (Syn2), 2.27 (Syn3), 0.5088 (Yeast), 2.64 (Wave), respectively. The correlation coefficient between the dispersion degree and Z is -0.9724, indicating that there is a relatively strong correlation between them.

Another factor that affects the values of Z is the similarity among input clusterings. To calculate the similarity among input clusterings, the comparison measures [9] are used.[1] For each pair of clusterings, their disagreement is calculated and normalized. The normalized disagreement is denoted as d. Then, the similarity is $1-d$. Finally, the average value of the similarities of all the pairs of the input clusterings is taken as the similarity of the input clusterings. Fig. 4 shows the numbers of fragments in terms of the similarities among fragments. The initial numbers of clusters of the first K-means run for Syn1, Syn2, Syn3, Yeast, and Wave are 5, 5, 5, 9, and 3, respectively. Then, the initial numbers of clusters for all the five data sets are increased by two step-by-step for other six successive K-means runs. The values of the correlation coefficient between Z and input clusterings' similarities shown in Fig. 4 are -0.3745, -0.3313, -0.3734, -0.682, and -0.1376, respectively, which reveal a strong negative correlation between Z and input clustering similarities.

Based on the experiments used to obtain the results of Figs. 3 and 4, we find that Z also partially depends on the number of clusters in each input clustering. The numbers of clusters in the input clusterings used to obtain Fig. 3 are less than those used to obtain Fig. 4. Take the Syn1 set as an example. In the experiments used to obtain Fig. 3, the numbers of clusters in each input clustering are equal to 5. In the experiments used to obtain Fig. 4, the numbers of clusters in the input clusterings are 5, 7, 9, 11, 13, 15, and 17, respectively. It can be observed that the values of Z in Fig. 3 experiments are also less than those in Fig. 4 experiments. The larger the number of clusters in each input clustering, the larger of Z it is very likely to be.

The four factors above are not independent of each other. The distribution of the input data affects the number of clusters in each input clustering. The number of input clustering affects the diversity (dissimilarity) of the input clustering. The four factors can be reduced to two factors: the distribution of the input data and the properties of the partition algorithms, for example the strategies of the algorithms, the parameters, and the number of algorithms. The diversity of the input clustering is determined by both factors. In our future work, we intend to provide a quantitative evaluation of the number of fragments and its dependent factors based on several related theoretical studies [1], [24], [42] on clustering aggregation.

[1]The comparison measure will be detailed in the following section.

III. Main Theory and Its Proof

This section gives an answer to the question asked in Section I on whether clustering aggregation can be achieved directly on data fragments. As cluster aggregation is unsupervised, a goodness measure is usually required to be defined in order to evaluate a candidate aggregation result. We first introduce two well-known goodness measures.

A. Goodness Measures

Goodness measures are usually heuristically defined according to the intuition that the higher the consistency between a given partition and the input partitions, the better the given partition.

1) Comparison Measure: Ginois *et al.* [9] define a simple 0–1 comparison criterion to measure the disagreement of two partitions with regard to a pair of points. Let π_1 and π_2 be partitions of a set X and for $(x, y) \in X^2$. Let the function $d_{xy}(\pi_1, \pi_2)$ be defined by

$$d_{xy}(\pi_1,\pi_2)=\begin{cases}0 & \text{if } \pi_1(x)=\pi_1(y) \text{ and } \pi_2(x)=\pi_2(y)\\ & \textit{or } \pi_1(x)\neq\pi_1(y) \text{ and } \pi_2(x)\neq\pi_2(y)\\ 1 & \text{otherwise.}\end{cases} \quad (1)$$

The disagreement between two partitions is the sum of 0/1 comparison over all pairs of points

$$d_X(\pi_1,\pi_2)=\sum_{(x,y)\in X^2} d_{xy}(\pi_1,\pi_2). \quad (2)$$

Then, for a candidate partition π, a measure of its disagreement of all the input partitions is defined by

$$D(\pi)=\sum_{i=1}^{H} d_X(\pi,\pi_i). \quad (3)$$

Under this measure, the smaller the disagreement $D(\pi)$, the better the partition π is. Ginois *et al.* [9] find that under this measure, a clustering aggregation problem can be transformed into a correlation clustering problem. Then, Ginois *et al.* [9] propose three point-based cluster aggregation algorithms which are used in this work as shown in Section IV. Studies in [4], [11] are also based on this measure.

2) Normalized Mutual Information Measure: In contrast with the comparison measure, another widely accepted strategy is to measure the agreement between two partitions. Strehl and Ghosh [16] define the mutual information [2], [3], [7] between a pair of partitions by

$$\phi(\pi_i,\pi_j)=\frac{\sum_{h=1}^{|\pi_i|}\sum_{l=1}^{|\pi_j|}|\pi_{ih}\cap\pi_{jl}|\log\left(\frac{|X|\cdot|\pi_{i,h}\cap\pi_{jl}|}{|\pi_{ih}||\pi_{jl}|}\right)}{\log|\pi_i|+\log|\pi_j|} \quad (4)$$

Then, for a candidate partition π, its overall agreement with the input partitions is defined by

$$\Phi(\pi)=\sum_{i=1}^{H}\phi(\pi,\pi_i). \quad (5)$$

Variations on (5) can be found in [8], [17]. The larger the value of $\Phi(\pi)$, the better the partition π.

B. Main Proposition and Proofs

To perform clustering aggregation directly on fragments, the data points in the same fragment should not be split in the consensus partition. To this end, the following proposition is proposed.

Proposition: Let π be the optimal partition via clustering aggregation based on D or Φ. Each data fragment is contained in a single subset of π.

1) Proof for Clustering Aggregation Based on D:

Lemma 1: Given a fragment F of X and input partitions $\prod = \{\pi_1, \ldots, \pi_i, \ldots, \pi_{\mathrm{H}}\}$. For each π_i, there exists a cluster $O \in \pi_i$ such that $F \subseteq O$.

The proof is obvious.

Lemma 2: Let π be a partition of X such that the function D defined on partitions of X reaches its minimum value at π. Let F be any fragment of the set of partitions $\pi_i, 1 \leq i \leq H$. Then, there exists $O \in \pi$ such that $F \subseteq O$.

Proof: See the Appendix A. ■

2) Proof for Clustering Aggregation Based on Φ:

Lemma 3: Let π be a partition of X at which the function $\pi' \mapsto \Phi(\pi')$ reaches its global maximum and let A, B be distinct elements of π. If there exists a fragment F such that $A \cap F$ and $B \cap F$ are both nonempty, then A and B are both contained in F.

Proof: See the Appendix B. ■

Lemma 4: Let π be a partition of X at which the function $\pi' \mapsto \Phi(\pi'$ reaches its global maximum and let A, B be distinct elements of π. If there exists a fragment F such that $A \cap F$ and $B \cap F$ are both nonempty, then $\Phi(\pi)$ can be increased by merging A and B.

Proof: According to Lemma 3, A and B are both from the same fragment. With a simple deduction, we can obtain that the numerator of $\Phi(\pi)$ remains unchanged when merging A and B. For $\phi(\pi, \pi_i)$, its denominator is $\log|\pi| + \log|\pi_i|$. When merging A and B, $|\pi|$ is decreased and then the denominator is decreased. As a result, $\Phi(\pi)$ is increased ■

Lemma 4 conversely indicates that the data points located in the same data fragment are definitely in the same subset of the optimal partition when using the NMI measure.

IV. Fragment-Based Clustering Aggregation

Section III concludes that clustering aggregation can directly be achieved on data fragments when either the comparison measure (3) or the NMI measure (5) is used. The conclusion provides a theoretical basis to guarantee that the fragment-based cluster aggregation is feasible. This idea is summarized as the fragment-based clustering aggregation approach shown in Fig. 5. The approach can be divided into three major steps: 1) generate data fragments (S1); 2) perform clustering aggregation on fragments (S2); and 3) transform the obtained fragment partition into a point partition, which is known as the consensus clustering (S3). The main step, S2, can be carried out by adapting existing point-based clustering aggregation

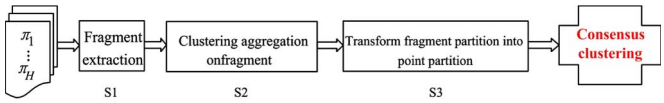


Fig. 5. Fragment-based clustering aggregation approach.

algorithms. Our proposition and lemmas in Section III do not provide a general method for adapting a point-based clustering aggregation algorithm into a fragment-based clustering aggregation algorithm. The adaptation depends on the specific point-based algorithm. This section will provide several adaptation examples.

Three new clustering aggregation algorithms are generated by modifying three existing point-based clustering aggregation algorithms [9], namely, Agglomerative, Furthest, and LocalSearch, respectively. To make clear the difference from the original point-based algorithms, the three new algorithms are called F-Agglomerative, F-Furthest, and F-LocalSearch, respectively.

According to [9], the underlying goodness measure of the above three point-based algorithms is the comparison measure. To simplify the descriptions, two distance matrices which are based on the (1)–(3) are introduced: the distance matrix between data points (DMP) and the distance matrix between fragments (DMF)

$$DMP(u,v) = \frac{1}{H} \cdot \left|\{i | 1 \le i \le H \text{ and } \pi_i(x_u) \ne \pi_i(x_v)\}\right|, \quad 1 \le u, v \le N$$

$$DMF(k,l) = DMP(u,v) \text{ where } x_u \in F_k,\ x_v \in F_l, \quad 1 \le k,\ l \le Z.$$

When a point v is considered, the cost of assigning a point v to a subset π_{ij} is computed as follows:

$$cost(v, \pi_{ij}) = \sum_{u \in \pi_{ij}} DMP(u,v) + \sum_{u \in (X \cap \bar{\pi}_{ij})} (1 - DMP(u,v)). \quad (6)$$

The cost of moving from a subset π_{ik} to π_{il} is

$$\hat{cost}(v, \pi_{ik} \to \pi_{il}) = cost(v, \pi_{il}) - cost(v, \pi_{ik}). \quad (7)$$

A. F-Agglomerative

The Agglomerative algorithm [9] is a classical bottom-up method. It initially places each data point in a single cluster. It then iteratively merges two clusters if the average distance between two clusters is less than 1/2. The algorithm stops when no two clusters have an average distance less than 1/2. The average distance between two clusters (π_{ik} and π_{jl}) is defined as

$$avg\_dist(\pi_{ik}, \pi_{jl}) = \sum_{u \in \pi_{ik}} \sum_{v \in \pi_{jl}} DMP(u,v) / \left(|\pi_{ik}||\pi_{jl}|\right). \quad (8)$$

Since the proposed approach is performed on fragments, the average distance (on fragments) between two clusters can be rewritten as

$$avg\_\partial(\pi_{ik}, \pi_{jl}) = \sum_{F_u | F_u \subset \pi_{ik}} \sum_{F_v | F_v \subset \pi_{jl}} DMF(u,v) \cdot \frac{|F_u||F_v|}{\pi_{ik}||\pi_{jl}|}. \quad (9)$$

Equation (9) is calculated on fragments. A fragment-based (F-Agglomerative) algorithm is shown in Algorithm 1.

Algorithm 1 Steps of F-Agglomerative

Input: $X=\{x_1, \ldots, x_i, \ldots, x_N\}$, $\prod=\{\pi_1, \ldots, \pi_i, \ldots, \pi_{\mathrm{H}}\}$.
Output: consensus clustering.
Steps:

1) Extract data fragments based on the input partitions.
2) Calculate the distance matrix for fragments (DMF).
3) Place each fragment in a single cluster.
4) Calculate the average distance between each pair of clusters using (9), and choose the smallest average distance.
5) Merge the corresponding clusters and return to (3), if the smallest average distance is below 0.5. Otherwise, go to the next step.
6) Transform the obtained fragment partition into a data point partition, and return the new data point partition.

B. F-Furthest

Furthest is a top-down algorithm [9] inspired by the furthest-first traversal algorithm [28]. The algorithm initially places all data points into a single cluster, searches for the pair of points whose distance (DMP distance) is the largest, and then makes each point the center of a new cluster. The remaining points are assigned to one or other of two clusters according to their distances to the two cluster centers. This following procedure is iterative. At each step, the furthest data point from the existing centers is chosen and taken as a new center of a singleton cluster; the nodes are assigned to the new center that leads to the least moving cost that defined by (7). If the total moving cost is not smaller than 0, the algorithm stops.

To adapt Furthest to our approach, we move fragments instead of data points at each step. Equations (8) and (9) can be rewritten for fragments as

$$cost(F_\zeta, \pi_{ij}) = \sum_{F_k | F_k \subset \pi_{ij}} DMF(\zeta, k) + \sum_{F_l | F_l \subset (X \cap \bar{\pi}_{ij})} (1 - DMF(\zeta, l)). \quad (10)$$

Then, the cost of moving F_ζ from cluster π_{ik} to π_{il} is

$$\hat{cost}(F_\zeta, \pi_{ik} \to \pi_{il}) = cost(F_\zeta, \pi_{il}) - cost(F_\zeta, \pi_{ik}). \quad (11)$$

The steps of F-Furthest are detailed in Algorithm 2.

Algorithm 2 Steps of F-Furthest

Input: $X=\{x_1,\ldots,x_i,\ldots,x_N\}$, $\prod=\{\pi_1,\ldots,\pi_i,\ldots,\pi_{\mathrm{H}}\}$.
Output: consensus clustering.
Steps:

1) Extract data fragments based on the input partitions.
2) Calculate the distance matrix for fragments (DMF).
3) Place all fragments in a single cluster, then find the pair of fragments whose distance (DMF distance) is furthest, and make each of them the center of a singleton cluster. Assign the rest of the fragments to one of the two clusters in order to achieve the minimum moving cost.
4) Choose the furthest fragment from the existing centers and make the fragment the center of a new singleton cluster. Calculate the cost of moving the fragment from the cluster it is located to the new cluster. If the moving cost is not smaller than 0, then go to (7).
5) Assign each of the remaining nodes to the new center if the moving cost is below 0.
6) If all moving costs are not smaller than 0, go to the next step. Otherwise, go to (4).
7) Transform the obtained fragment partition into a data point partition, and return the new data point partition.

C. F-LocalSearch

LocalSearch is an application of a local-search heuristics [25] to the correlation clustering [9] which aims to find a partition of a data set such that the data points with low DMP distances (less than 1/2) are kept together and the data points with high DMP distance (more than 1/2) are kept separate as much as possible. An initial clustering is given. Then, it goes through all the data points by considering a local movement of a point from a cluster to another one, or a new singleton cluster. For each local movement, we calculate the moving cost by (6). If one placement yields a negative moving cost, then the local movement improves the goodness of the current clustering. Consequently, the node is placed in the cluster that yields the minimum moving cost. The algorithm iterates until all moving costs are not smaller than 0. Based on LocalSearch, the steps of F-LocalSearch are detailed in Algorithm 3.

Algorithm 3 Steps of F-LocalSearch

Input: $X=\{x_1,\ldots,x_i,\ldots,x_N\}$, $\prod=\{\pi_1,\ldots,\pi_i,\ldots,\pi_{\mathrm{H}}\}$.
Output: consensus clustering.
Steps:

1) Extract data fragments based on the input partitions.
2) Calculate the distance matrix for fragments (DMF).
3) Generate a new partition that ensures data in a fragment kept together.
4) Calculate all the moving costs of placing one fragment from its current cluster to any other cluster. If the minimum moving cost is not smaller than 0, go to next step. Otherwise, move the fragment with the minimum moving cost and repeat (4).
5) Transform the obtained fragment partition into a data point partition, and return the new data point partition.

TABLE III
DESCRIPTION OF THE DATA SETS

Agglomerative	$O(HN^2+N^2\log N)$
F-Agglomerative	$O(NH+N\log Z+HZ^2+Z^2\log Z)$
Furthest	$O(HN^2+C^2N)$
F-Furthest	$O(NH+N\log Z+HZ^2+C^2Z)$
LocalSearch	$O(HN^2+IN^2)$
F-LocalSearch	$O(NH+N\log Z+HZ^2+IZ^2)$

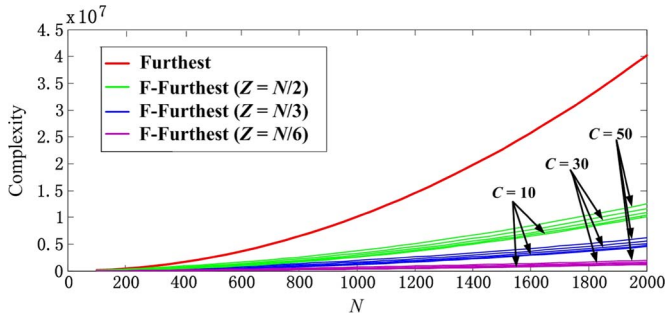


Fig. 6. Computational complexity of F-Furthest.

TABLE IV
DESCRIPTION OF THE DATA SETS

Dataset	Data size	No. of features	No. of classes
Hayes-Roth (#1)	160	5	3
Glass (#2)	214	10	6
Breast (#3)	569	32	2

D. Computational Complexity Analysis

Let C be the number of consensual clusters and I be the number of iterations of an algorithm. Table III lists the complexities of the point-based algorithms and the fragment-based algorithms. Fig. 6 shows the variation of complexity in terms of different values of N, Z, and C. Given a specific data set and input partitions, the complexity of fragment-based approach depends more on Z compared to C.

V. EXPERIMENTS

This section evaluates the performances of the proposed fragment-based clustering aggregation approach. We are mainly concerned with whether the computational complexity of an existing clustering aggregation algorithm can be reduced by being adapted to a fragment-based algorithm. In our experiments, we compare the computational efficiency of the three original clustering aggregation algorithms (i.e., Agglomerative, Furthest, and Local-Search) and their fragment-based versions (i.e., F-Agglomerative, F-Furthest, and F-LocalSearch), respectively. To check whether the fragment-based algorithms sacrifice the effectiveness, aggregation errors are also calculated.

A. Experimental Setup

Five experimental data sets, namely Hayes-Roth (#1), Glass (#2), Breast (#3), Yeast (#4), and Wave (#5), are obtained from the UCI Repository [43]. Table IV summarizes the details of the first three sets (#1–#3). The details of Yeast (#4) and Wave (#5) are given in Table II. We use running time, impurity index [9], [42], and average entropy [39] to evaluate the

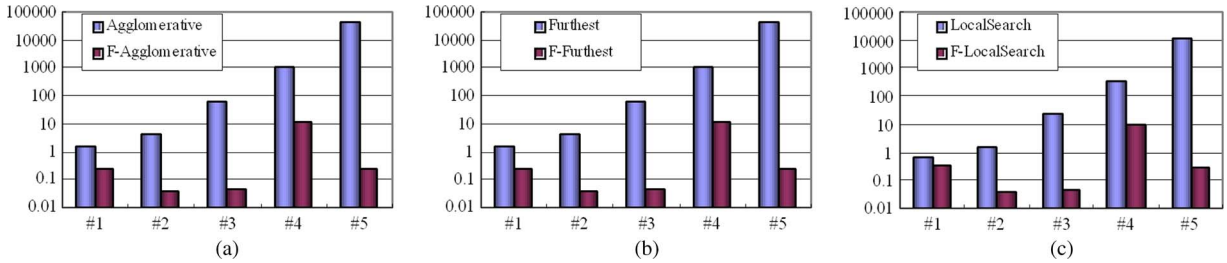


Fig. 7. Running time results on the five data sets with strategy 1).

TABLE V
Ec (%) ERRORS ON THE FIVE DATA SETS WITH STRATEGY 1)

Dataset	Agglomerative	F-Agglomerative	Furthest	F-Furthest	LocalSearch	F-LocalSearch
Hayes-Roth (#1)	33.75	36.25	48.31	36.40	37.50	36.25
Glass (#2)	44.99±2.54	45.06±1.63	52.64±1.93	45.06±2.05	44.79±1.78	44.45±1.23
Breast (#3)	41.40±2.27	41.40±2.27	41.40±3.87	41.40±1.45	40.90±1.29	40.90±1.29
Yeast (#4)	25.09±2.12	12.80±2.28	30.79±2.12	19.21±0.62	12.63±0.76	12.80±2.01
Wave (#5)	26.14±1.61	34.50±1.30	27.11±2.87	41.23±3.12	34.31±2.76	34.31±2.76

TABLE VI
AE VALUES ON THE FIVE DATA SETS WITH STRATEGY 1)

Dataset	Agglomerative	F-Agglomerative	Furthest	F-Furthest	LocalSearch	F-LocalSearch
Hayes-Roth (#1)	0.036	0.024	0.032	0.055	0.708	0.731
Glass (#2)	0.482±0.014	0.748±0.026	0.779±0.032	1.170±0.027	1.460±0.020	1.460±0.017
Breast (#3)	1.210±0.072	1.210±0.072	1.455±0.104	1.622±0.042	1.536±0.039	1.536±0.039
Yeast (#4)	1.692±0.062	1.737±0.095	3.280±0.063	3.100±0.084	2.865±0.118	2.874±0.113
Wave (#5)	0.771±0.023	0.757±0.057	1.540±0.081	1.052±0.042	0.795±0.043	0.795±0.043

performance of each algorithm, taking the data classes as groundtruth. Let m_{ij} denote the size of the majority class in cluster π_{ij}. The impurity index E_c is defined as

$$E_c(\pi_i) = \frac{\sum_{j=1}^{Pi} (|\pi_{ij}| - m_{ij})}{N}. \tag{12}$$

The average entropy (AE) of clusterings used in [39] is also applied to evaluate the results. The criteria measure the information needed to correctly classify all the points in the clustering. The smaller the value of this criterion, the better the performance of the clustering results. For the ith cluster (π_i), let m_{ij}^k denote the number of points which come from the kth class in π_{ij}. Let $\mathcal{C}$ be the number of classes of the original data. AE is defined as

$$AE(\pi_i) = \sum_{j=1}^{Pi} \frac{|\pi_{ij}|}{N} \times \left(\sum_{k=1}^{|\mathcal{C}|} -\frac{m_{ij}^k}{|\pi_{ij}|} \log_2 \frac{m_{ij}^k}{|\pi_{ij}|} \right). \tag{13}$$

All the experiments are performed on a 3.4 GHz PC with 1 GB memory. Before performing clustering aggregation, we produce the input partitions for each data set. There are different strategies to create input partitions. Two typical strategies are 1) to perform the same clustering algorithm with different parameters and 2) to perform different clustering algorithms. In this experiments, we perform these two strategies. 1) The input partitions are repeatedly obtained by K-means with different initial numbers of cluster centers. 2) The input partitions are obtained by four different clustering algorithms (e.g., K-means and SOM). Each algorithm generates two clusterings with different numbers of clusters: one equals the number of classes (of the corresponding data set) adding two; the other equals the number of classes adding four.

Considering that Hayes-Roth is a categorical data set and each of its attributes defines an input partition, the results are only shown in strategy 1). When using clustering aggregation on categorical data, each feature dimension provides a natural way of clustering the data (The data with the same values are taken as a cluster.). The simple clusterings obtained from the feature dimensions are aggregated to form the output clustering.

In each strategy, as K-means randomly selects initial cluster centers in each run, each experiment including the clustering aggregation runs 10 times, and then the results are reported.

B. Results With Strategy 1)

This subsection reports the results on the five data sets when input partitions are obtained by K-means with different initial numbers of clusters (except Hayes-Roth), which are as follows: 1) Glass set—4, 5, 6, 7, and 8; 2) Yeast set—8, 9, 10, 11, and 12; and 3) Breast set and Wave set—2, 3, 4, 5, and 6. Fig. 7 shows the experimental results with respect to the running time. Tables V and VI shows the means and standard deviations in terms of E_c (%) and AE, respectively.

It can be observed that the running time of fragment-based algorithms is much smaller than that of original ones. With the increasing of the data size, the time consumption of fragment extraction occupies little proportion. As a consequence, the running time of original algorithms increases sharply while that of the fragment-based algorithms remains very small.

We note that the E_c values and AE values are different in some cases. For example, on the Yeast set (#4), the E_c of

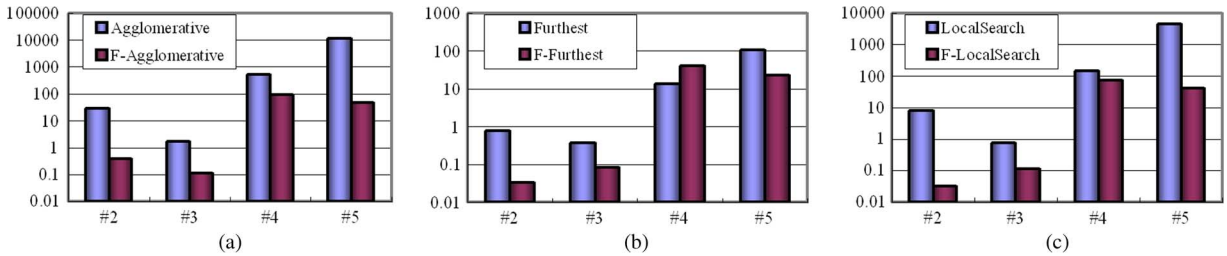


Fig. 8. Running time results on the #2–#5 data sets with strategy (2).

TABLE VII
E_c (%) ERRORS ON THE #2–#5 DATA SETS WITH STRATEGY (2)

Dataset	Agglomerative	F-Agglomerative	Furthest	F-Furthest	LocalSearch	F-LocalSearch
Glass (#2)	37.37±1.97	37.37±1.63	37.37±1.93	37.37±2.05	37.37±1.78	37.37±1.23
Breast (#3)	22.32±2.27	16.74±2.67	29.3±3.87	18.6±0.62	16.74±1.29	16.74±1.69
Yeast (#4)	48.62±2.12	47.68±3.28	61.41±2.12	60.6±2.44	47.54±0.76	47.32±2.01
Wave (#5)	32.81±2.61	32.81±1.30	39.07±2.87	51.55±3.12	35.17±2.06	35.17±3.75

TABLE VIII
AE VALUES ON THE #2–#5 DATA SETS WITH STRATEGY (2)

Dataset	Agglomerative	F-Agglomerative	Furthest	F-Furthest	LocalSearch	F-LocalSearch
Glass (#2)	0.238±0.011	0.604±0.057	0.758±0.095	0.635±0.077	0.497±0.069	0.418±0.081
Breast (#3)	1.153±0.106	1.073±0.075	1.458±0.073	1.455±0.051	1.456±0.047	1.458±0.062
Yeast (#4)	1.702±0.088	2.197±0.113	3.637±0.157	3.931±0.149	1.723±0.097	1.838±0.065
Wave (#5)	0.769±0.067	1.278±0.084	1.377±0.115	1.188±0.077	0.796± 0.068	1.318±0.049

Agglomerative is larger than that of F-Agglomerative, while the AE of Agglomerative is smaller than that of F-Agglomerative. The major reason is that the two measure criteria evaluate different aspects of the clustering results. E_c focuses on the majority-class points in each cluster, while AE focuses on the distribution of the points of all the classes in each cluster. In [39], the values of the two criteria also behave differently in some cases (Interested readers can refer to [39, Tables 2–3]).

C. Results With Strategy (2)

In this strategy, each algorithm still produces two input clusterings for each data set. Take the Glass set as an example, when applying K-means, each algorithm produces two input clusterings with the numbers of clusters equaling 8 and 10, respectively. Fig. 8, Tables VII, and VIII show the results with respect to running time, E_c, and AE, respectively. Similar observations to strategy (1) are obtained: most fragment-based algorithms are more efficient than the original algorithms. There is only one exception that the running time of F-Furthest is larger than Furthest on the Yeast set (#4) in Fig. 8(b). The cures in Fig. 6 and the complexities in Table III indicate that the complexity of F-Furthest is largely influenced by the number of final clusters (i.e., C). The larger the number of fragments and the larger the number of final clusters, the larger the complexity of F-Furthest is. Compared with other sets, the average number of final clusters on Yeast is 13, which is the largest. The average number of fragments of the Yeast set under Strategy (2) is 279, which is also the largest. Hence, it is reasonable that the running time of F-Furthest on yeast is higher than that of Furthest. In all the other cases, the running time of F-Furthest is smaller than that of Furthest. In terms of E_c and AE values, the two approaches change only slightly (Fig. 9).

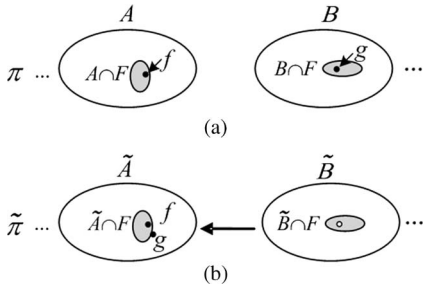


Fig. 9. Optimal partition π (a) and its variation $\tilde{\pi}$ (b).

TABLE IX
RESULTS ON THE MAGIC SET

Algorithms	Running time (s)	E_c (%)	AE
Agglomerative	129855	40.05	0.831
F-Agglomerative	1.172	36.01	0.879
Furthest	1206.325	46.88	1.413
F-Furthest	1.024	46.88	1.413
LocalSearch	1032451	41.61	0.968
F-LocalSearch	1.300	34.64	0.754

D. Results on Large Data Sets

A data set with 19020 points, namely Magic, is downloaded from UCI Repository [43] to test the performances of our proposed approach. This data set is larger than all the five data sets used in the aforementioned experiments. There are two classes in the data set and each sample has 11 features. Because the running time of the original algorithms is very large, we run only once for each algorithm. The input partitions are obtained using K-means with different initial numbers—2, 3, 4, 5, and 6. Table IX shows the experimental results. Although the data set is very large, the time consumption of fragment-based algorithms is still quite low.

TABLE X
RESULTS ON THE SYN4 SET

Algorithm	Running time(s)	Ec(%)	AE
F-Agglomerative	3632+5.787s	19.13	0.3861
F-Furthest	3632+4.915s	25.98	0.4224
F-LocalSearch	3632+7.572s	25.83	0.4391

Another synthetic data set consisting of 1.2 million samples is introduced to further test the scalable ability of our approach. This data set, called Syn4, is extended from Syn2 shown in Fig. 2 simply by increasing the number of data points. As the running time for existing algorithms is too large, only the results of the proposed algorithms are reported. The input partitions are obtained using K-means with different initial numbers—5, 6, 7, 8, and 9. The average number of fragments is 132. Table X shows the results. For running time values, 3632 is the runing time of fragment creation. The results show that the time complexity of the fragment-based methods is acceptable for large data.

E. Discussions

The experimental results show the satisfactory performance of the fragment-based clustering aggregation approach. The running time of the fragment-based algorithms is far less than the running time of the corresponding point-based ones and in addition appears to less sensitive to the number of data points. The reason is that for a specific data set and fixed input partitions, the number of data fragments is the main factor that affects the time complexity of a fragment-based algorithm. To further check the performances of the fragment-based algorithms, the bagging sampling approach [45] is performed on the five data sets (Hayes-Roth, Glass, Breast, Yeast, and Wave). Each bagging sampling generates a new data set containing 2/3 the number of data points of the original data set. For each data set, the sampling is repeated 10 times and 10 new data sets are generated. On each new set, the generation of input partitions is the same as the Strategy (1). The means and standard deviations of results on the 10 new data sets are recorded. The running time of fragment-based algorithms is still smaller than that of point-based ones. The average running-time values of F-Agglomerative and Agglomerative on Glass are 0.0502s and 0.4419s, respectively. The average running-time values of F-Furthest and Furthest on Glass are 0.0677s and 0.8910s, respectively. The average running-time values of F-LocalSearch and LocalSearch on Glass are 0.0371s and 0.5735s, respectively. The performances of both approaches change slightly in terms of E_c and AE.

In the experiments, it is observed that in several cases when the numbers of fragments are not much smaller than the numbers of data points, the computational complexity is increased [#4 in Fig. 8(b)]. Therefore, a two-layer approach may be more required. At first, we generate the fragments and calculate the ratio between the number of fragments and the data size. Second, if the ratio is smaller than a predefined value, the fragment-based clustering aggregation is utilized; otherwise, the point-based clustering aggregation is still utilized. This approach may work in real applications.

We conduct a statistical test to analyze the small differences between the performances between the fragment-based clustering aggregation approach and the point-based clustering aggregation approach in terms of E_c and AE. As we should compare point-based algorithms with fragment-based algorithms on multiple data sets, we introduce the idea in [41] that leverages the Wilcoxon rank sum test [40] to compare algorithms on multiple data sets. In statistics, the Wilcoxon rank-sum test is a nonparametric statistical hypothesis test for assessing whether two independent samples of observations have equal values. In our experiments, the results of point-based approach and fragment-based approach are taken as two independent samples of observations. On each data set, the comparisons between each point-based algorithm and its corresponding fragment-based algorithm are recorded. The test procedure is based on the comparisons of the results between each pair of point-based algorithm and its corresponding fragment-based algorithm listed in Tables V–VIII. For example, on the Hayes-Roth in Table V, the E_c error of Agglomerative is lower than that of F-Agglomerative, so the number of times that point-based outperforms fragment-based is increased by one. Similarly, the E_c error of Furthest is larger than that of F-Furthest on Glass, so the number of times that fragment-based outperforms point-based is increased by one. In terms of E_c, there are 11 times that fragment-based methods outperform point-based methods, 7 times that they are equal, and 9 times that the former is inferior to the latter. In terms of AE, there are 14 times that fragment-based methods outperform point-based methods, 5 times that they are equal, and 8 times that the former is inferior to the latter. On applying the test, we concluded that there are no significant difference between the overall performances of the two approaches in terms of both E_c and AE.

VI. CONCLUSION

This paper has defined data fragments and proposed a new fragment-based clustering aggregation approach. This approach is based on the proposition that in an optimal partition each fragment is a subset of a cluster. We have proved this proposition for two widely used goodness measures: comparison measure and NMI measure. Existing point-based algorithms can be brought into our proposed approach. As the number of data fragments is usually far smaller than the number of data points, clustering aggregation based on data fragments is very likely to have a much lower time complexity than directly on data points. To demonstrate the efficiency of the proposed approach, three new clustering aggregation algorithms are presented, namely, F-Agglomerative, F-Furthest, and F-LocalSearch (based on three existing point-based clustering aggregation algorithms ones). We have conducted experiments on six public data sets. The results show that the three new algorithms outperform the original clustering aggregation algorithms in terms of running time without sacrificing effectiveness.

APPENDIX A

The main idea of the proof for ***Lemma*** 2 can be illustrated by Fig. 9. π is an optimal partition of X such that the

comparison-measure function D defined on partitions of X reaches its minimum value at π. F is a fragment. A and B are two clusters and $A, B \in \pi$. $f \in A \cap F$ and $g \in B \cap F$ are two data points. If we can prove that either placing g into A [as shown in Fig. 9(b)] or placing f to B can reduce the value of the comparison-measure function, then π is not an optimal partition. The contradiction indicates that the data points in a fragment cannot be located in different clusters in an optimal partition as illustrated by ***Lemma*** 2.

Lemma 2: Let π be a partition of X such that the function D defined on partitions of X reaches its minimum value at π. Let F be any fragment of the set of partitions $\pi_i, i = 1, \ldots, H$. Then, there exists a cluster $O \in \pi$ such that $F \subseteq O$.

Proof: Suppose, on the contrary, that there exists a fragment F and sets $A \in \pi, B \in \pi$ such that $A \cap F$ is nonempty and $B \cap F$ is nonempty. Let f, g be elements of F such that f is an element of A and g is an element of B. Let $S \mapsto C(g, S)$ be the function defined on subsets S of X by

$$C(g, S) = \sum_{i=1}^{H} \sum_{x \in S} \delta\left(\pi_i(x) \neq \pi_i(g)\right) \tag{14}$$

where δ is an indication function. The elements f, g are in the same fragment F, thus $C(g, S) = C(f, S)$ for all subsets S of X. Let $\tilde{A}, \tilde{B}$ be the subsets of X defined by $\tilde{A} = A \cup \{g\}, \tilde{B} = B - \{g\}$ and let $\tilde{\pi}$ be the partition of X defined by $\tilde{\pi} = \{\tilde{A}, \tilde{B}\} \cup (\pi - \{A, B\})$.[2] Let V be the function defined by $V = D(\pi) - D(\tilde{\pi})$. Let x, y be any two elements of X. If at least one of x, y, is not contained in $A \cup B$, then $d_{xy}(\pi, \pi_i) = d_{xy}(\tilde{\pi}, \pi_i), 1 \leq i \leq H$. It follows that:

$$V = \sum_{i=1}^{H} \sum_{(x,y) \in A \cup B} \left(d_{xy}(\pi, \pi_i) - d_{xy}(\tilde{\pi}, \pi_i)\right). \tag{15}$$

Let V_1, V_2, V_3 be defined as follows:

$$V_1 = \sum_{i=1}^{H} \sum_{(x,y) \in A^2} \left(d_{xy}(\pi, \pi_i) - d_{xy}(\tilde{\pi}, \pi_i)\right)$$

$$V_2 = \sum_{i=1}^{H} \sum_{(x,y) \in A \times B} \left(d_{xy}(\pi, \pi_i) - d_{xy}(\tilde{\pi}, \pi_i)\right)$$

$$V_3 = \sum_{i=1}^{H} \sum_{(x,y) \in B^2} \left(d_{xy}(\pi, \pi_i) - d_{xy}(\tilde{\pi}, \pi_i)\right). \tag{16}$$

It follows from the above definitions that:

$$V = V_1 + 2V_2 + V_3. \tag{17}$$

If $(x, y) \in A^2$, then $(x, y) \in \tilde{A}^2$, from which it follows that $d_{xy}(\pi, \pi_i) = d_{xy}(\tilde{\pi}, \pi_i)$. The term V_1 is thus equal to zero. To evaluate V_2, the set $A \times B$ is divided into the subsets $A \times (B - \{g\})$ and $A \times \{g\}$. The contribution to V_2 from the elements (x, y) of $A \times (B - \{g\})$ is zero. It follows that:

$$\begin{aligned} V_2 &= \sum_{i=1}^{H} \sum_{x \in A} \left(d_{xg}(\pi, \pi_i) - d_{xg}(\tilde{\pi}, \pi_i)\right) \\ &= \sum_{i=1}^{H} \sum_{x \in A} \left(1 - 2\delta\left(\pi_i(x) \neq \pi_i(g)\right)\right) \\ &= H|A| - 2C(g, A). \end{aligned} \tag{18}$$

To evaluate V_3, the set $B \times B$ is divided into the three subsets: $(B - \{g\}) \times (B - \{g\})$, $B \times \{g\}$, and $\{g\} \times (B - \{g\})$. The contribution to V_3 from the elements of $(B - \{g\}) \times (B - \{g\})$ is zero. The contribution to V_3 from $B \times \{g\}$ is equal to the contribution to V_3 from $\{g\} \times (B - \{g\})$. It follows that:

$$\begin{aligned} V_3 &= 2\sum_{i=1}^{H} \sum_{x \in B - \{g\}} \left(d_{xg}(\pi, \pi_i) - d_{xg}(\tilde{\pi}, \pi_i)\right) \\ &= 2\sum_{i=1}^{H} \sum_{x \in B - \{g\}} 2\delta\left(\pi_i(x) \neq \pi_i(g)\right) - 2H\left(|B| - 1\right) \\ &= \sum_{i=1}^{H} \sum_{x \in B} 2\delta\left(\pi_i(x) \neq \pi_i(g)\right) - 2H\left(|B| - 1\right) \\ &= 4C(g, B) - 2H\left(|B| - 1\right). \end{aligned} \tag{19}$$

It follows from the above expressions for V_1, V_2, V_3, that:

$$V = 2H|A| - 4C(g, A) + 4C(g, B) - 2H\left(|B| - 1\right). \tag{20}$$

Now that the roles of A and B are interchanged, in that the element f is removed from A and added to B to yield a new partition $\overline{\pi}$. Let the new value of V be $\overline{V}$. It follows that:

$$\overline{V} = 2H|B| - 4C(f, B) + 4C(f, A) - 2H\left(|A| - 1\right). \tag{21}$$

If $V > 0$ or $\tilde{V} > 0$ denoting that $D(\pi) - D(\tilde{\pi}) > 0$ or $D(\pi) - D(\overline{\pi}) > 0$, then $\tilde{\pi}$ or $\overline{\pi}$ is better than π, which contradicts to the statement that π is an optimal partition. Hence, the lemma is established. Suppose, if possible, that $V \leq 0$ and $\overline{V} \leq 0$. On adding the two expressions for V and $\overline{V}$, it follows that:

$$-4C(g, A) + 4C(g, B) - 4C(f, B) + 4C(f, A) + 4H \leq 0 \tag{22}$$

which is impossible, because $C(f, A) = C(g, A)$ and $C(f, B) = C(g, B)$. As a result, the lemma is established.

APPENDIX B

Let w, q be the real valued functions defined by

$$\begin{aligned} w(x) &= x\log(x) - (x-1)\log(x-1), \quad 1 < x \\ q(x) &= w(x+1) - w(x), \quad 1 < x. \end{aligned}$$

The definitions of w and q are extended to the range $1 \leq x < \infty$ by setting $w(1) = 0$ and $q(1) = w(2)$. A short calculation shows that $x \mapsto q(x)$ is strictly monotonic decreasing and strictly convex for $1 \leq x < \infty$.

[2] $\tilde{\pi}$ and π are also sets of data subsets (clusters).

Lemma 3: Let π be a partition of X at which the function $\pi' \mapsto \Phi(\pi')$ reaches its global maximum and let A, B be distinct elements of π. If there exists a fragment F such that $A \cap F$ and $B \cap F$ are both nonempty, then A and B are both contained in F.

Proof: Let f be an element of $A \cap F$ and let g be an element of $B \cap F$. Define the subsets $\tilde{A}, \tilde{B}$ of X by

$$\tilde{A} = A \cup \{g\}, \quad \tilde{B} = B - \{g\}. \tag{23}$$

A detailed proof of the theorem is given for the case in which $\tilde{B}$ is nonempty. If $\tilde{B}$ is empty, then the theorem still holds, but minor modifications to the proof are required. When $\tilde{B}$ is nonempty, the denominator of $\Phi(\pi')$ is unchanged. Hence, the denominator is omitted in the following steps. Define the partition $\tilde{\pi}$ of X by

$$\tilde{\pi} = \{\tilde{A}, \tilde{B}\} \cup (\pi - \{A, B\}). \tag{24}$$

Let α, β, r_i, s_i be defined by $\alpha = |A|, \beta = |B|, r_i = |A \cap O_i|, s_i = |B \cap O_i|$. Let O_i be the element of π_i which contains $\{f, g\}$. The contribution of O_i to

$$Con(O_i) = \phi(\tilde{\pi}, \pi_i) - \phi(\pi, \pi_i) \tag{25}$$

is

$$|\tilde{A} \cap O_i| \log\left(\frac{|X||\tilde{A} \cap O_i|}{|\tilde{A}||O_i|}\right) + |\tilde{B} \cap O_i| \log\left(\frac{|X||\tilde{B} \cap O_i|}{|\tilde{B}||O_i|}\right) - r_i \log\left(\frac{|X|r_i}{\alpha|O_i|}\right) - s_i \log\left(\frac{|X|s_i}{\beta|O_i|}\right). \tag{26}$$

On observing that

$$|\tilde{A} \cap O_i| = r_i + 1, \quad |\tilde{B} \cap O_i| = s_i - 1. \tag{27}$$

Equation (26) can be reduced to

$$r_i \log\left(\frac{|\tilde{A} \cap O_i|\alpha}{|\tilde{A}|r_i}\right) + s_i \log\left(\frac{|\tilde{B} \cap O_i|\beta}{|\tilde{B}|s_i}\right) + \log\left(\frac{|\tilde{A} \cap O_i||\tilde{B}|}{|\tilde{A}||\tilde{B} \cap O_i|}\right). \tag{28}$$

Let π_{il} be any element of π_i distinct from O_i. It follows that:

$$\tilde{A} \cap \pi_{il} = A \cap \pi_{il}, \quad \tilde{B} \cap \pi_{il} = B \cap \pi_{il}. \tag{29}$$

Based on (28) and (29), the contribution of π_{il} to $Con(O_i)$ is thus

$$Con(\pi_{il}) = |A \cap \pi_{il}| \log\left(\frac{\alpha}{|\tilde{A}|}\right) + |B \cap \pi_{il}| \log\left(\frac{\beta}{|\tilde{B}|}\right). \tag{30}$$

On adding $Con(\pi_{il})$ over all the elements π_{il} of π_i distinct from O_i, the following expression is obtained:

$$|A \cap \overline{O_i}| \log\left(\frac{\alpha}{|\tilde{A}|}\right) + |B \cap \overline{O_i}| \log\left(\frac{\beta}{|\tilde{B}|}\right). \tag{31}$$

By adding (28) and (31), we have

$$\phi(\tilde{\pi}, \pi_i) - \phi(\pi, \pi_i) = w(\beta) - w(\alpha+1) + w(r_i+1) - w(s_i). \tag{32}$$

On adding the equations (32), it follows that:

$$\Phi(\tilde{\pi}) - \Phi(\pi) = H \cdot w(\beta) - H \cdot w(\alpha+1) + \sum_{i=1}^{H} w(r_i+1) - \sum_{i=1}^{H} w(s_i) \tag{33}$$

The function $\pi' \mapsto \Phi(\pi')$ reaches its global maximum at π. Thus, it follows from (33) that:

$$w(\alpha+1) - w(\beta) \geq \frac{1}{H}\sum_{i=1}^{H} w(r_i+1) - \frac{1}{H}\sum_{i=1}^{H} w(s_i). \tag{34}$$

Let $\tilde{\pi}'$ be the partition of X defined by

$$\tilde{\pi}' = \{A - \{f\}, B \cup \{f\}\} \cup (\pi - \{A, B\}). \tag{35}$$

On evaluating the expression $\Phi(\tilde{\pi}') - \Phi(\pi)$ it follows that:

$$w(\beta+1) - wt(\alpha) \geq \frac{1}{H}\sum_{i=1}^{H} w(s_i+1) - \frac{1}{H}\sum_{i=1}^{H} w(r_i). \tag{36}$$

On adding (34) and (36), it follows that:

$$q(\alpha) + q(\beta) \geq \frac{1}{H}\sum_{i=1}^{H} q(r_i) + \frac{1}{H}\sum_{i=1}^{H} q(s_i). \tag{37}$$

Let $\bar{r}, \bar{s}$ be defined by

$$\bar{r} = \frac{1}{H}\sum_{i=1}^{H} r_i. \tag{38}$$

It follows from (37) and the strict convexity of q that:

$$q(\alpha) + q(\beta) \geq q(\bar{r}) + q(\bar{s}). \tag{39}$$

It follows from the definitions of α, β, r_i and s_i that $\alpha \geq \bar{r}$ and $\beta \geq \bar{s}$. The function q is strict monotonic decreasing, thus, it follows from (39) that:

$$\alpha = \bar{r} \quad \left(|A| = \frac{1}{H}\sum_{i=1}^{H} |A \cap O_i|\right) \tag{40}$$

$$\beta = \bar{s} \quad \left(|B| = \frac{1}{H}\sum_{i=1}^{H} |B \cap O_i|\right) \tag{41}$$

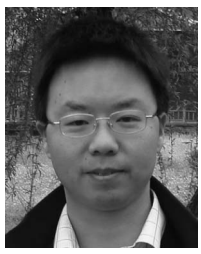
which yield $\alpha = r_i (|A| = |A \cap O_i|), 1 \leq i \leq H$ and $\beta = s_i (|B| = |B \cap O_i|), 1 \leq i \leq H$. It now follows from the definitions of α, β, r_i and s_i that $A \subseteq O_i, 1 \leq i \leq H$ and $B \subseteq O_i, 1 \leq i \leq H$. Thus A, B are in the same fragment.

ACKNOWLEDGMENT

The authors would like to thank Q. Wang and X. Zou for their helpful suggestions.

REFERENCES

[1] N. Ailon, M. Charikar, and A. Newman, "Aggregating Inconsistent Information: Ranking and Clustering," in *Proc. STOC*, New York, 2005, pp. 684–693.
[2] M. AlRazgan and C. Domeniconi, "Weighted Clustering Aggregation," in *Proc. SIAM SDM*, 2006, pp. 90–101.
[3] H. G. Ayad and M. S. Kamel, "Cumulative voting consensus method for partitions with variable number of clusters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 160–173, Jan. 2008.
[4] S. N. Bahloul, B. Rouba, and Y. Amghar, "Minimization of the disagreements in clustering aggregation," in *Proc. ICIC*, 2008, pp. 517–524.
[5] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Proc. ICML*, 2004, pp. 36–43.
[6] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *Proc. ICML*, 2003, pp. 186–193.
[7] A. L. N. Fred and A. K. Jain, "Robust data clustering," in *Proc. IEEE CVPR*, 2003, vol. 2, pp. II-128–II-133.
[8] D. Gondek and T. Hofmann, "Non-redundant clustering with conditional ensembles," in *Proc. ACM KDD*, 2005, pp. 70–77.
[9] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, pp. 1–30, Mar. 2007.
[10] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.
[11] T. Li and C. Ding, "Weighted consensus clustering," in *Proc. SIAM SDM*, 2008, pp. 798–809.
[12] T. Lange and J. Buhmann, "Combining partitions by probabilistic label aggregation," in *Proc. ACM KDD*, 2005, pp. 147–156.
[13] N. Nguyen and R. Caruana, "Consensus clustering," in *Proc. IEEE ICDM*, 2005, pp. 607–612.
[14] H. Shinnou and M. Sasaki, "Ensemble document dlustering using weighted hypergraph generated by NMF," in *Proc. ACL Demo Poster Sessions*, 2007, pp. 77–80.
[15] V. Singh, L. Mukherjee, J. Peng, and J. Xu, "Ensemble clustering using semidefinite programming," in *Proc. NIPS*, pp. 145–152, 2007.
[16] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining partitions," in *Proc. AAAI*, 2002, pp. 93–98.
[17] 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. 2003.
[18] A. Topchy, A. K. Jain, and W. F. Punch, "Clustering aggregation: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.
[19] K. Tumer and A. K. Agogino, "Ensemble clustering with voting active clusters," *Pattern Recognit. Lett.*, vol. 29, no. 14, pp. 1947–1953, Oct. 2008.
[20] X. Sevillano, F. Alłas, and J. C. Socoro, "Borda consensus: A new consensus function for soft cluster ensembles," in *Proc. ACM SIGIR*, 2007, pp. 743–744.
[21] H. Wang, Z. Li, and Y. Cheng, "Weighted latent dirichlet allocation for cluster ensemble," in *Proc. IEEE GEC*, 2008, pp. 437–441.
[22] Y. Yang and K. Chen, "Temporal data clustering via weighted clustering ensemble with different representations," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 2, pp. 307–320, Feb. 2011.
[23] Y. Hong, S. Kwong, Y. Chang, and Q. Ren, "Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm," *Pattern Recognit.*, vol. 41, no. 9, pp. 2742–2756, Sep. 2008.
[24] A. P. Topchy, M. H. C. Law, A. K. Jain, and A. L. Fred, "Analysis of consensus partition in cluster ensemble," in *Proc. IEEE ICDM*, 2004, pp. 225–232.
[25] T. Coleman, J. Saunderson, and A. Wirth, "A local-search 2-approximation for 2-correlation-clustering," *Lecture Notes in Computer Science (LNCS)*, pp. 308–319, 2008.
[26] O. R. Terrades, E. Valveny, and S. Tabbone, "Optimal classifier fusion in a non-bayesian probabilistic framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1630–1644, Sep. 2009.
[27] I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 2, pp. 146–156, Apr. 2002.
[28] D. Hochbaum and D. Shmoys, "A best possible heuristic for the K-center problem," *Math. Oper. Res.*, vol. 10, no. 2, pp. 180–184, May 1985.
[29] X. Zhang, L. Jiao, F. Liu, L. Bo, and M. Gong, "Spectral clustering ensemble applied to texture features for SAR image segmentation," *IEEE Trans. Geosci. Remote Ssens*, vol. 46, no. 7, pp. 2126–2136, Jul. 2008.
[30] R. Ghaemi, M. Sulaiman, H. Ibrahim, and N. Mutspha, "A survey: Clustering ensembles techniques," in *Proc. World Acad. Sci., Eng. Technol.*, 2009, vol. 50, pp. 636–645.
[31] F. Wang, X. Wang, and T. Li, "Generalized cluster aggregation," in *Proc. IJCAI*, 2009, pp. 1279–1284.
[32] X. Hu and I. Yoo, "Cluster ensemble and its applications in gene expression analysis," in *Proc. Asia-Pacific Bioinform.*, 2004, pp. 297–302.
[33] X. Sevillano, G. Cobo, F. A. , and J. C. Socoro, "Feature diversity in cluster ensembles for robust document clustering," in *Proc. ACM SIGIR*, 2006, pp. 697–698.
[34] Z. Zhou and W. Tang, "Clusterer ensemble," *Knowledge-Based Systems*, vol. 19, no. 1, pp. 77–83, Mar. 2006.
[35] S. T. Sarasamma, Q. A. Zhu, and J. Huff, "Hiearchical Kohonenen net for anomaly detection in network security," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 302–312, Apr. 2005.
[36] C. Gentile and M. Sznaier, "An improved voronoi diagram based neural net for pattern classification," *IEEE Trans. Neural Netw.*, vol. 12, no. 5, pp. 1227–1234, Sep. 2001.
[37] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 3, pp. 433–439, Jun. 1999.
[38] Y. Hadi, F. Essannouni, and R. O. H. Thami, "Video summarization by K-medoid clustering," in *Proc. ACM SAC*, 2006, pp. 1400–1401.
[39] M. Zhang and Z. Zhou, "Multi-instance clustering with applications to multi-instance prediction," *Appl. Intell.*, vol. 31, no. 1, pp. 47–68, Aug. 2009.
[40] B. Rosner, R. J. Glynn, and M.-L. T. Lee, "Incorporation of clustering effects for the Wilcoxon rank sum test: A large-sample approach," *Biometrics*, vol. 59, no. 4, pp. 1089–1098, Dec. 2003.
[41] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res. (JMLR)*, vol. 7, pp. 1–30, Dec. 2006.
[42] M. Meilă, "Comparing clusterings—An axiomatic view," in *Proc. ICML*, 2005, pp. 577–584.
[43] . [Online]. Available: http://archive.ics.uci.edu/ml/
[44] I. O. Kyrgyzov, H. Maître, and M. Campedel, "A method of clustering combination applied to satellite image analysis," in *Proc. ICIAP*, 2007, pp. 81–86.
[45] D. N. Politis and J. P. Romano, "The stationary bootstrap," *J. Amer. Stat. Assoc.*, vol. 89, no. 428, pp. 1303–1313, Dec. 1994.
[46] D. R. Cox and P. A. W. Lewis, *The Statistical Analysis of Series of Events*. London: Methuen, 1966.
[47] J. Wu, H. Xiong, and J. Chen, "A data distribution view of clustering algorithms," in *Encyclopedia of Data Warehousing and Mining*, vol. 1, J. Wang, Ed., 2nd ed. Hershey, PA: IGI Global, 2008, pp. 374–381.

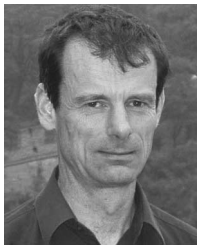


Ou Wu (M'10) received the B.Sc. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2003, and the M.Sc. degree and the Ph.D. degree in computer science from the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, China, in 2006 and 2011, respectively.

From April 2007, he joined NLPR as an Assistant Professor. His research interests include data mining and Web appearance analysis.

Weiming Hu (SM'09) received the Ph.D. degree from the Department of Computer Science and Engineering, Zhejiang University, Zhejiang, China.

From April 1998 to March 2000, he was a Postdoctoral Research Fellow with the Institute of Computer Science and Technology, Founder Research and Design Center, Peking University, Beijing, China. Since April 1998, he has been with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, where he is currently a Professor and a Ph.D. Student Supervisor with the laboratory. He has published more than 100 papers in national and international journals and international conferences. His research interests include visual surveillance, filtering of Internet objectionable information, retrieval of multimedia, and understanding of Internet behaviors.



Stephen J. Maybank (SM'06) received the B.A. degree in mathematics from King's College, Cambridge, U.K., in 1976 and the Ph.D. degree in computer science from Birkbeck College, University of London, London, U.K., in 1988.

In 1980, he joined the Pattern Recognition Group at Marconi Command and Control Systems, Frimley, U.K. In 1989, he moved to the GEC Hirst Research Centre, Wembley, U.K. During 1993–1995, he was a Royal Society/Engineering and Physical Sciences Research Council Industrial Fellow with the Department of Engineering Science, University of Oxford, Oxford, U.K. In 1995, he was a Lecturer at the Department of Computer Science, University of Reading, Reading, U.K. In 2004, he became a Professor with the School of Computer Science and Information Systems, Birkbeck College. His research interests include the geometry of multiple images, camera calibration, visual surveillance, information geometry, and the applications of statistics to computer vision.

Mingliang Zhu, photograph and biography not available at the time of publication.

Bing Li, photograph and biography not available at the time of publication.