ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/319219167

Fuzzy Consensus Clustering With Applications on Big Data

Article *in* IEEE Transactions on Fuzzy Systems · August 2017
DOI: 10.1109/TFUZZ.2017.2742463

CITATION
1

READS
124

6 authors, including:

Zhiang Wu
Nanjing University of Finance and Economics
92 PUBLICATIONS **520** CITATIONS
SEE PROFILE

Hongfu Liu
Beihang University (BUAA)
17 PUBLICATIONS **122** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

FIW Dataset and Evaluations View project

All content following this page was uploaded by Zhiang Wu on 19 December 2017.

The user has requested enhancement of the downloaded file.

Fuzzy Consensus Clustering With Applications on Big Data

Junjie Wu, Zhiang Wu, *Member, IEEE*, Jie Cao, Hongfu Liu, Guoqing Chen, and Yanchun Zhang

***Abstract*—Consensus clustering aims to find a single partition of data that agrees as much as possible with existing basic partitions. Given its robustness and generalizability, consensus clustering has emerged as a promising solution to find cluster structures inside heterogeneous big data rising from various application domains. In the area of fuzzy systems, however, research along this line is still in its initial stage with some unsystematic algorithmic studies. Finding a fuzzy consensus partition from multiple fuzzy basic partitions in an efficient, flexible, and robust way is still an exciting open problem calling for further investigation. In light of this, this paper provides a systematic study of fuzzy consensus clustering (FCC) from a utility perspective. Specifically, we first define the objective function of FCC clearly using the novel fuzzified contingency matrix. We then derive a family of FCC Utility functions termed as FCCU that can transform FCC to a weighted piecewise fuzzy *c*-means clustering (piFCM) problem. This helps us to establish an algorithmic framework for FCC with flexible choice of utility functions, and speeds FCC significantly with a FCM-like iterative process of piFCM. To meet the big data challenge, we further parallelize FCC on the Spark platform with both vertical and horizontal segmentation schemes. Extensive experiments on various real-world datasets demonstrate the excellent performance of FCC, even with a majority of poor basic partitions. In particular, our method exhibits interesting potential for big data clustering in two real-life applications concerned with online event detection and overlapping community detection, respectively.**

***Index Terms*—Apache spark, big data, fuzzy *c*-means, fuzzy consensus clustering (FCC), utility function.**

Manuscript received April 19, 2017; revised July 10, 2017; accepted August 14, 2017. Date of publication August 21, 2017; date of current version November 29, 2017. The work of J. Wu was supported in part by the National Natural Science Foundation of China under Grant 71531001, Grant U1636210, Grant 71471009, Grant 71490723, Grant 71322104, Grant 71171007, and Fundamental Research Funds for Central Universities. The work of Z. Wu and J. Cao were supported in part by NSFC under Grant 71571093, Grant 91646204, Grant 71372188, Grant 71672053, in part by National Center for International Joint Research on E-Business Information Processing (2013B01035), and in part by National Key Technologies Research and Development Program of China under Grant 2017YFD0401001. The work of Y. Zhang was supported in part by Australia Research Council (ARC) Project (DP140100841). *(Corresponding author: Zhiang Wu.)*

J. Wu is with School of Economics and Management, Beihang University, Beijing 100191, China (e-mail: wujj@buaa.edu.cn).

Z. Wu and J. Cao are with Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing 210003, China (e-mail: zawuster@gmail.com; caojie690929@163.com).

H. Liu is with College of Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: liu.hongf@husky.neu.edu).

G. Chen is with School of Economics and Management, Tsinghua University, Beijing 100084, China (e-mail: Chengq@sem.tsinghua.edu.cn).

Y. Zhang is with the Centre for Applied Informatics, College of Engineering and ScienceVictoria University, Footscray, Vic 3011, Australia (e-mail: Yanchun.Zhang@vu.edu.au).

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/TFUZZ.2017.2742463

I. Introduction

CONSENSUS clustering, also known as cluster ensemble or clustering aggregation, aims to find a single partition of data that agrees as much as possible with multiple existing basic partitions (BPs). It has been widely recognized that consensus clustering can generate robust clustering results; find usual clusters, handle noise, outliers and sample variations; and integrate solutions from multiple distributed sources of data or attributes [1]–[3]. A recent study also gives rigorous proofs to the robustness and generalizability of consensus clustering, which lays theoretical foundation for the success of consensus clustering [4].

Due to the rich information conveyed by the membership grade matrix, fuzzy clustering has long been widely used in many real-world application domains where well-separated clusters are typically not available [5]. Applications requiring fuzzy clustering abound, ranging from deriving taxonomy for a collection of ambiguous news articles [6] to analyze gigantic Facebook logs [7] and patient stratification [8]. However, with the explosive growth of online big data in recent years, researchers and practitioners come to realize that a single fuzzy clustering might fail with complex data, such as high-dimensional text corpora. Hence, the introduction of consensus clustering to fuzzy clustering becomes natural, and *fuzzy consensus clustering* (FCC) thus emerges as a new research frontier.

FCC aims to identify a soft consensus partition with overlapping clusters from a set of crisp or fuzzy BPs. In the literature, tremendous efforts have been taken in designing effective and efficient algorithms for consensus clustering. In particular, much attention has been paid to crisp consensus clustering (CCC) [4], [9]–[11], which requires crisp BPs as inputs and outputs crisp consensus partitions. However, little attention has been paid to FCC, where only a few fragmental algorithmic studies have been addressed. For instance, Dimitriadou *et al.* developed a voting-based method and minimized the average square distance between the consensus one and a set of BPs [12]. With a similar approach, Punera and Ghosh [13] extended the work of [2], where each instance was represented by the concatenated cluster membership probability distributions derived by fuzzy BPs. While these pilot studies offer some algorithmic techniques in the context of FCC, the underlying mechanism of consensus learning, i.e., the objective function, has not been clearly interpreted in theory. Since the fuzzy clustering algorithms usually have high computational cost, scaling the FCC framework to handle real-world big data is another challenging issue.

1063-6706 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

In light of this, we propose the systematic framework of FCC to fuse several fuzzy BPs efficiently from a utility perspective, which is further enhanced and parallelized to handle the big data clustering. Both theoretical and empirical studies indicate that FCC is clear, efficient, flexible, and robust in consensus clustering. Our main contributions are summarized as follows:

First, we propose the novel *Fuzzified Contingency Matrix*, upon which the objective function of FCC is clearly defined in a utility perspective. To our best knowledge, our study is among the earliest work that makes use of utility functions as a guide to the consensus learning of soft cluster structures, which is of great help in revealing the underlying mechanism of consensus clustering.

Second, we derive a special family of utility functions called FCCU that successfully transforms FCC into a weighted piFCM. This helps us to establish an algorithmic framework for FCC with *flexible* choice of utility functions, and the *efficiency* of FCC is boosted by an FCM-like iterative process, piFCM.

Third, we propose both the *vertical and horizontal segmentation* schemes and establish a simple yet feasible framework for FCC on big data. This framework is further accelerated by parallelization on the Spark platform. To the best of our knowledge, this is among the earliest attempts to leverage distributed platforms for FCC.

Finally, we conduct extensive experiments on various real-world datasets to demonstrate the excellent performance of FCC, especially its *robustness* against a majority of poor BPs. More importantly, FCC exhibits great potentials in big data clustering in two interesting applications for both online event detection and overlapping community detection.

The remainder of this paper is organized as follows. In Section II, we present the research mostly related to our study. In Section III, we give some preliminary knowledge, and then define FCC in Section IV. In Section V, we derive FCC utility function (FCCU) and transform FCC to piFCM. In Section VI, we propose the framework for big data clustering. Experiments are given in Section VII, followed by the two case studies in Section VIII. We finally conclude our work in Section IX.

II. Related Work

Here, we introduce related work on fuzzy clustering and consensus clustering, and highlight the differences between existing work and ours.

A. Fuzzy Clustering

In fuzzy clustering, a data object may belong to more than one cluster with different degrees of membership. The Fuzzy c-Means algorithm was first developed by Dunn [14] and later improved by Bezdek [5]. Dave and Krishnapuram analyzed several popular robust clustering methods and established a connection between fuzzy set theory and robust statistics [15]. Hathaway and Bezdek extended the rough-fuzzy c-means algorithm to non-Euclidean dissimilarity data [16]. In recent years, Wu *et al.* have proposed the point-to-centroid distance, which is a generalization of distance functions for Fuzzy c-Means [17]. Dave and Sen proposed the fuzzy relational data clustering algorithm that can handle datasets containing outliers and can deal with all kinds of relational data [18]. Hall *et al.* presented a single-pass fuzzy c-means algorithm for scaling the fuzzy c-means to very large numbers of examples [19], [20]. Many other fuzzy clustering related problems have also received increased research attention, such as feature selection [21], kernel clustering [22], optimization [23], etc.

B. Consensus Clustering

Due to its excellent robustness, tremendous efforts have been devoted to CCC. The existing work can be roughly generalized into two categories as follows. The first category employs the utility function to measure the similarity between the consensus clustering and multiple BPs, which usually finds the final clustering result by maximizing the aggregated utility function value. Different utility functions measure the similarity of two partitions in different aspects, rendering different objective functions for consensus clustering. For instance, Topchy *et al.* proposed a Quadratic Mutual Information based objective function for consensus clustering, and used K-means clustering to find a solution [24]. They further extended their work to using the EM algorithm with a finite mixture of multinomial distributions for consensus clustering [25]. Along this line, Wu *et al.* transferred the consensus clustering into a K-means clustering problem with K-means consensus clustering (KCC) utility functions, of which the necessary and sufficient conditions are also given [10]. Then, Liu *et al.* applied KCC on high-dimensional text clustering with Entropy-based utility function [26]. In addition, there are some other interesting objective functions for the consensus clustering, such as the ones introduced by [27]–[29]. The second category summarizes the information of BPs into a coassociation matrix, which counts how many times two instances occur in the same cluster. Based on the coassociation matrix, any graph partitioning algorithms can be conducted for the final clustering. For example, Fred *et al.* utilized the agglomerative hierarchical clustering [9], yet Strehl *et al.* designed three graph-based algorithms for consensus clustering [2]. Recently, Liu *et al.* have proposed a spectral ensemble clustering method, which ran spectral clustering on the coassociation matrix and transformed it as a weighted K-means problem [4], [30]. Other methods include relabeling and voting [31], locally adaptive cluster-based methods [32], genetic algorithm-based methods [33], and still many more.

Compared with the above CCC, relatively few research efforts have been conducted to fuse several fuzzy BPs into a consensus one. Dimitriadou *et al.* extended a voting-based method and minimized the average square distance between consensus one and a set of basic partitions [12]. However, the mapping relations should be known in advance and it fails to handle BPs with differing cluster numbers. Similarly, Punera and Ghosh [13] extended the work of [2], where each instance was represented by the concatenated cluster membership probability distributions derived by fuzzy BPs and K-means clustering with KL-divergence was invoked to obtain the final result. Sevillano *et al.* proposed to rank the instances according to their membership probability with respect to each cluster [34], and conducted the thorough experimental comparisons on voting-based

TABLE I
MAIN MATH SYMBOLS

Notation	Description
π	the crisp consensus partition for all data instances
π_i	the ith crisp BP for the ith data sample
C_k	the kth cluster in a crisp or fuzzy consensus partition
$C_j^{(i)}$	the jth cluster in the ith crisp or fuzzy BP
r	the number of BPs
K	the number of clusters for a consensus partition
K_i	the number of clusters for the ith BP
U (U_f)	the utility function measuring the similarity of crisp (fuzzy) partitions
w_i	a user-specific weight for the ith BP
$n_{kj}^{(i)}$	the number of data objects shared by crisp C_k and $C_j^{(i)}$
$\boldsymbol{\mu}$	the fuzzy membership-degree matrix for consensus partition
$\boldsymbol{\mu}^{(i)}$	the fuzzy membership-degree matrix for the ith BP
$s_{kj}^{(i)}$	the similarity of all data objects to C_k and $C_j^{(i)}$, respectively
m	the fuzzifier for $\boldsymbol{\mu}$
$v_k^{(i)}$	the kth centroid for the ith BP in a piecewise FCC

methods [6]. Moreover, Collaborative Fuzzy Clustering worked with a finite family of partial data residing at multiple sites and fused them as a global structure [3], [35], [36]. While it shares some common ground with FCC from a combinatorial perspective given a distributed environment, it is essentially a local optimization heuristic that attempts to conduct fuzzy clustering on original distributed data. In contrast, FCC focuses on fusing various existing clustering results and has a global objective function that guides the consensus clustering, which can be solved naturally via a simple FCM-like iterative process.

In this paper, we propose a framework of FCC to fuse several fuzzy BPs in a utility way. Based on FCCUs, theoretic analysis are provided to equivalently transform this problem into a Fuzzy c-means problem, which can be formulated in a neat mathematical way and be solved efficiently. Both input and output partitions of FCC could be fuzzy, which implies its flexibility in handling both soft and CCC.

III. PRELIMINARIES

In this section, we introduce the concept of consensus clustering. As a key component, the utility function for consensus clustering will then be introduced in more details.

Following the convention, we use $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{R}_{++}$, $\mathbb{R}^d$, and $\mathbb{R}^{n\times d}$ to denote the sets of reals, nonnegative reals, positive reals, d-dimensional real vectors, and $n \times d$ real matrix, respectively. $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{Z}_{++}$, $\mathbb{Z}^d$, and $\mathbb{Z}^{n\times d}$ are defined analogously for integers. For a d-dimensional real vector x, $\|x\|_p$ denotes the l_p-norm of x, i.e., $\|x\|_p = (\sum_{i=1}^d |x_i|^p)^{1/p}$, $p \geq 1$. $x^\top$ denotes the transposition of x. The gradient of a multivariate function f is denoted as ∇f. The logarithm of base 2 is denoted as $\log$. Table I lists the main math symbols we would use throughout this paper for the purpose of quick reference. More details about these symbols are given when first used in the text.

A. Consensus Clustering

In the literature, most of the relevant studies on consensus clustering are concerned with *crisp* partitioning. Hence, we begin with introducing CCC.

TABLE II
CONTINGENCY MATRIX

		π_i				
		$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	$\sum$
π	C_1	$n_{11}^{(i)}$	$n_{12}^{(i)}$	$\cdots$	$n_{1K_i}^{(i)}$	n_{1+}
	C_2	$n_{21}^{(i)}$	$n_{22}^{(i)}$	$\cdots$	$n_{2K_i}^{(i)}$	n_{2+}
	$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
	C_K	$n_{K1}^{(i)}$	$n_{K2}^{(i)}$	$\cdots$	$n_{KK_i}^{(i)}$	n_{K+}
	$\sum$	$n_{+1}^{(i)}$	$n_{+2}^{(i)}$	$\cdots$	$n_{+K_i}^{(i)}$	n

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of data objects (a.k.a. points, instances, samples). A partitioning of $\mathcal{X}$ into K crisp clusters is represented as a collection of K subsets of objects in $\mathcal{C} = \{C_k | k = 1, \ldots, K\}$, with $C_k \bigcap C_{k'} = \emptyset \, \forall k \neq k'$, and $\bigcup_{k=1}^K C_k = \mathcal{X}$, or as a label vector $\pi = (L_\pi(x_1), \ldots, L_\pi(x_n))^T$, where L_π maps x_l to some label in $\{1, 2, \ldots, K\}$, $1 \leq l \leq n$.

The existing CCC methods can be generally classified into two categories, i.e., the methods with and without global objective functions, respectively [37]. In this paper, we mainly focus on former methods, which are typically formulated as combinatorial optimization problems as follows. Given r *BPs* of $\mathcal{X}$ in $\Pi = \{\pi_1, \pi_2, \ldots, \pi_r\}$ (a BP is the result of a partitioning on $\mathcal{X}$ by some clustering algorithm), the goal is to solve the following optimization problem:

$$\max_{\pi} \sum_{i=1}^{r} w_i U(\pi, \pi_i) \tag{1}$$

where π is called the *consensus partition*, $U : \mathbb{Z}_{++}^n \times \mathbb{Z}_{++}^n \mapsto \mathbb{R}$ is a *utility function* that measures the similarity between π and any π_i, and $w_i \in [0, 1]$ is a user-specified *weight* for BP π_i, $\sum_{i=1}^r w_i = 1$. Sometimes a dissimilarity function, e.g., the well-known Mirkin distance [38], rather than a utility function is used for consensus clustering. In that case, we can simply turn the maximization problem into a minimization problem without changing the nature of the problem.

B. Utility Function

Equation (1) is often solved by some heuristics or metaheuristics. Therefore, the choice of the utility function is crucial for the success of a consensus clustering, since it largely determines the heuristics to employ.

In the literature, some external measures originally proposed for cluster validity have been adopted as utility functions for consensus clustering, such as Normalized Mutual Information [2], Category Utility Function [39], Quadratic Mutual Information [24], and Rand Index [29]. Although these functions are in different forms, they can be derived uniformly from the so-called *contingency matrix*, as shown in Table II.

Suppose we have two partitions: π and π_i, containing K and K_i clusters, respectively. In Table II, $n_{kj}^{(i)}$ denotes the number of data objects shared by cluster $C_j^{(i)}$ (in π_i) and cluster C_k (in π).

As a result, $n_{k+} = \sum_{j=1}^{K_i} n_{kj}^{(i)}$, $n_{+j}^{(i)} = \sum_{k=1}^{K} n_{kj}^{(i)}$ represent the numbers of data objects in C_k and $C_j^{(i)}$, respectively. Nearly all commonly used utility functions can be defined on this matrix. For instance, let $P_k^{(i)} = (n_{k1}^{(i)}/n_{k+}, \ldots, n_{kK_i}^{(i)}/n_{k+})^T$, then the well-known Category Utility Function [39] can be formulated as:

$$U_c(\pi, \pi_i) \propto \sum_{k=1}^{K} n_{k+} \|P_k^{(i)}\|_2^2. \tag{2}$$

Though defined on the same contingency matrix, the utility functions yet hold different mathematical properties, and also pose computational challenges of varying degrees to consensus clustering. Existing CCC methods are usually specially designed to fit one or several utility functions, which may inevitably lose flexibility when dealing with real-world big data gathered from various application domains. One exception is from [10], in which a framework for K-means-based consensus clustering is proposed, with a general form for various (in fact, infinite) utility functions. However, whether CCC can be extended to FCC and how to deal with big data, remain very interesting open problems.

IV. Problem Definition

In this section, we define the problem of FCC to be studied in this paper. We begin with the definition of the *fuzzified contingency matrix*, upon which utility functions for FCC can be defined.

A. Fuzzified Contingency Matrix

Let us first recall $n_{kj}^{(i)}$ in Table II, which is the number of data objects shared by cluster C_k and cluster $C_j^{(i)}$. Let v denote a n-dimensional binary vector, with $v(l) = 1$ if $x_l \in C_k$, or 0 otherwise. In this way, v becomes the binary membership vector for all data objects to C_k. Similarly, we can define v', the binary membership vector for all data objects to $C_j^{(i)}$, with $v'(l) = 1$ if $x_l \in C_j^{(i)}$, or 0 otherwise. We therefore have

$$n_{kj}^{(i)} = v^\top v' \tag{3}$$

which implies that the "similarity" between two clusters can be measured by the inner product of two membership vectors corresponding to the two clusters, respectively. This simple idea can be extended to the fuzzy case, which turns Table II into the so-called *fuzzified contingency matrix* as follows.

Let μ_{lk} and $\mu_{lj}^{(i)}$ denote the membership degrees of x_l to C_k and $C_j^{(i)}$, respectively. We then have the membership degree matrices $\boldsymbol{\mu} = [\mu_{lk}]_{n\times K}$ and $\boldsymbol{\mu}^{(i)} = [\mu_{lj}^{(i)}]_{n\times K_i}$. Let

$$\mu_{\cdot k} = (\mu_{1k}, \ldots, \mu_{lk}, \ldots, \mu_{nk})^\top \tag{4}$$

$$\mu_{\cdot j}^{(i)} = \left(\mu_{1j}^{(i)}, \ldots, \mu_{lj}^{(i)}, \ldots, \mu_{nj}^{(i)}\right)^\top \tag{5}$$

TABLE III
Fuzzified Contingency Matrix

		$\boldsymbol{\mu}^{(i)}$				
		$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	$\sum$
$\boldsymbol{\mu}$	C_1	$s_{11}^{(i)}$	$s_{12}^{(i)}$	$\cdots$	$s_{1K_i}^{(i)}$	s_{1+}
	C_2	$s_{21}^{(i)}$	$s_{22}^{(i)}$	$\cdots$	$s_{2K_i}^{(i)}$	s_{2+}
	$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
	C_K	$s_{K1}^{(i)}$	$s_{K2}^{(i)}$	$\cdots$	$s_{KK_i}^{(i)}$	s_{K+}
	$\sum$	$s_{+1}^{(i)}$	$s_{+2}^{(i)}$	$\cdots$	$s_{+K_i}^{(i)}$	s

then the similarity between C_k and $C_j^{(i)}$ in the fuzzy case is given by

$$s_{kj}^{(i)} = (\mu_{\cdot k})^\top \mu_{\cdot j}^{(i)} \quad \forall\, k, j. \tag{6}$$

We further fuzzify $s_{kj}^{(i)}$ by introducing a fuzzifier $m \in (1, +\infty)$ to μ_{lk}. That is, we let

$$\mu_{\cdot k}^m = ((\mu_{1k})^m, \ldots, (\mu_{lk})^m, \ldots, (\mu_{nk})^m)^\top \tag{7}$$

but keep $\mu_{\cdot j}^{(i)}$ unchanged in (6), which leads to

$$s_{kj}^{(i)} = (\mu_{\cdot k}^m)^\top \mu_{\cdot j}^{(i)} \quad \forall\, k, j. \tag{8}$$

Along this way, we can have the fuzzified contingency matrix in Table III, where

$$s_{k+} = \sum_{j=1}^{K_i} s_{kj}^{(i)} = (\mu_{\cdot k}^m)^\top (1, \ldots, 1)^\top = \|\mu_{\cdot k}^m\|_1 \tag{9}$$

$$s_{+j}^{(i)} = \sum_{k=1}^{K} s_{kj}^{(i)} = \left(\sum_{k=1}^{K} \mu_{\cdot k}^m\right)^\top \mu_{\cdot j}^{(i)} \tag{10}$$

$$s = \sum_{k=1}^{K} s_{k+} = \sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 = \|\sum_{k=1}^{K} \mu_{\cdot k}^m\|_1. \tag{11}$$

Based on the fuzzified contingency matrix, utility functions for FCC can be defined, with details given in Section V.

Note that the following vectors related to the fuzzified contingency matrix will be also frequently used:

$$\mu_{l\cdot} = (\mu_{l1}, \ldots, \mu_{lK})^\top \quad \forall\, l \tag{12}$$

$$\mu_{l\cdot}^m = ((\mu_{l1})^m, \ldots, (\mu_{lK})^m)^\top \quad \forall\, l \tag{13}$$

$$\mu_{l\cdot}^{(i)} = \left(\mu_{l1}^{(i)}, \ldots, \mu_{lK_i}^{(i)}\right)^\top \quad \forall\, l \tag{14}$$

$$P_k^{(i)} = \left(\frac{s_{k1}^{(i)}}{s_{k+}}, \ldots, \frac{s_{kK_i}^{(i)}}{s_{k+}}\right)^\top = \left(\frac{(\mu_{\cdot k}^m)^\top \boldsymbol{\mu}^{(i)}}{\|\mu_{\cdot k}^m\|_1}\right)^\top = \sum_{l=1}^{n} \frac{(\mu_{lk})^m}{\|\mu_{\cdot k}^m\|_1} \mu_{l\cdot}^{(i)} \quad \forall\, k. \tag{15}$$

Apparently, $\|\mu_{l\cdot}\|_1 = \|\mu_{l\cdot}^{(i)}\|_1 = \|P_k^{(i)}\|_1 = 1$, indicating all discrete distributions. It is noteworthy that the fuzzy factor m

introduced here is to provide intrinsic adjustment to $\mu_{l\cdot}$, during consensus clustering. That is, a larger m tends to generate more uniform membership degrees for x_l to all clusters.

B. Fuzzy Consensus Clustering

We now define the FCC formally. We still have (1) as the objective, but the utility functions are defined on the fuzzified contingency matrix with a fuzzifier m. Also note that the consensus partition is no longer an integer vector π now, but rather an $n \times K$ membership degree matrix $\boldsymbol{\mu}$, where K is the number of clusters for consensus clustering. Similarly, the BP π_i should be replaced by a $n \times K_i$ membership degree matrix $\boldsymbol{\mu}^{(i)}$, where K_i is the number of clusters in that BP.

Specifically, we aim to find a *fuzzy consensus partition*, $\boldsymbol{\mu}$, for the following optimization problem:

$$\max_{\boldsymbol{\mu}} \sum_{i=1}^{r} w_i U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m) \tag{16}$$

where U_f is the *fuzzy utility function*, $m \in (1, +\infty)$ is a user-specified fuzzifier, $\Pi = \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(r)}\}$ is the set of BPs, and $w = (w_1, \ldots, w_r)^\top$ is the set of user-specified weights for BPs, with $\|w\|_1 = 1$.

Our task is to design an appropriate heuristic to solve (16) such that FCC is

1) *Accurate:* It is able to find a high-quality fuzzy consensus partition.
2) *Efficient:* It is able to find the consensus partition efficiently with robustness.
3) *Flexible:* It can choose utility functions to suit different application domains.

To achieve the above goals simultaneously is far from trivial. For instance, we might use some meta-heuristics, e.g., simulated annealing [29] or genetic algorithms [40], to solve (16), which however can hardly be efficient for just normal size data, much less the everrising online big data. For another, we might just pick up a best BP from Π as the consensus partitioning, which is highly efficient but may well sacrifice clustering accuracy seriously.

The discussions above imply that to ignore the very properties of specific utility functions is not a good idea for FCC. In other words, the ideal heuristic we seek for FCC should be "somehow" related to the properties of the utility functions. If the family of utility functions holding these properties is fairly large, then the flexibility of FCC could be ensured.

V. From FCC to Piecewise Fuzzy c-Means Clustering

In this section, we propose a fuzzy c-means-like approach for FCC defined in (16). We begin with the analysis of utility functions for FCC.

A. Sum-of-Cluster Utility Function (SOCU)

Recall the Category Utility Function in (2). It conveys two points critically important for designing an appropriate fuzzy utility function U_f:

1) U_f could be formulated in a cluster-based manner; that is, the overall utility obtained from the fuzzified contingency matrix could be decomposed into the utility coming from each cluster of the consensus partition (denoted as $U_f(k)$ for short), with the cluster size as the weight.
2) $U_f(k)$ could be formulated as the function of $P_k^{(i)}$, i.e., the discrete distribution derived from the kth row of the fuzzified contingency matrix. For instance, the Category Utility Function takes the squared l_2-norm of $P_k^{(i)}$, i.e., $\|P_k^{(i)}\|_2^2$, as the utility obtained from the kth cluster. Intuitively, a cluster with higher quality should lead to a larger $U_f(k)$.

The above discussion indeed gives us a clue for designing utility functions for FCC. We put it into the following definition:

Definition 1: A fuzzy utility function U_f is called a SOCU, if

$$U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m) = \sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \, \phi(P_k^{(i)}) \tag{17}$$

where $P_k^{(i)}$ is the discrete distribution defined by (15), and ϕ is a function defined on discrete distributions.

The above definition suggests a cluster-based view of utility functions. It reduces the complex problem to find a function ϕ defined on discrete distributions. For instance, if we let $\phi(x) = \|x\|_2^2$ and $m \to 1$ for the crisp clustering case, U_f then reduces to the Category Utility Function U_c defined by (2). Note that a SOCU is not necessarily a metric, since $\boldsymbol{\mu}$ and $\boldsymbol{\mu}^{(i)}$ are treated asymmetrically in Definition 1.

We further explore the ϕ function in a SOCU that measures the utility gained from every cluster of the consensus partition. Intuitively, we would expect ϕ not to be "weird," e.g., like a piecewise function, which is hard to handle and lacks interpretability, but the candidate set for ϕ is still too large. A practical way is to empower ϕ with some special property, like the convexity. For instance, ϕ for the Category Utility Function is the squared l_2-norm, which is an obvious convex function *w.r.t.* each component of a discrete distribution.

It could be argued that why a convex function is appropriate for ϕ. To understand this, we should first ascertain what is a high quality $P_k^{(i)}$ for a consensus partition. Being a discrete distribution, it is not difficult to infer that a highly biased $P_k^{(i)}$ implies a heavy overlap between the kth cluster of the consensus partition and some cluster in the ith BP, which is considered positive for giving clue to clear clusters [41]. As a result, a convex ϕ would be more appropriate than a concave one in favor of a bias $P_k^{(i)}$. For instance, as a well-known concave function, the Shannon entropy will reach the upper limit when $P_k^{(i)}$ is in a strict uniform distribution.

B. FCC Utility Function

Given SOCU defined on convex ϕ, we now formulate the utility functions particularly suitable for FCC problem defined by (16). Let us first recall the formulation of $P_k^{(i)}$ in (15). Let $a_{lk} = (\mu_{lk})^m / \|\mu_{\cdot k}^m\|_1$, we have $P_k^{(i)} = \sum_{l=1}^{n} a_{lk} \mu_{l\cdot}^{(i)}$, with

$\sum_{l=1}^{n} a_{lk} = 1$. This implies that $P_k^{(i)}$ is essentially the linear combination of all $\mu_{l\cdot}^{(i)}$. From this angle, we can define the *utility increment* for the kth cluster as $\Delta_k^{(i)} = \phi(P_k^{(i)}) - \sum_{l=1}^{n} a_{lk}\phi(\mu_{l\cdot}^{(i)})$, which measures the utility gain from the BP to the consensus partition. By Jensen's Inequality [42], $\Delta_k^{(i)} \leq 0$ for the convexity of ϕ.

Through this, we can have the total utility increment by simply making summation of $\Delta_k^{(i)}$ over all clusters, which defines the so-called "FCC Utility Function" as follows.

Definition 2: A fuzzy utility function U_f is called a FCCU for the FCC problem defined by (16), if

$$\begin{aligned} U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m) &= \sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \Delta_k^{(i)} \\ &= \sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \phi(P_k^{(i)}) - \sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 \phi(\mu_{l\cdot}^{(i)}) \end{aligned} \tag{18}$$

where ϕ is a convex function defined on discrete distributions.

Compared to SOCU, FCCU in (18) has an additional item: $-\sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 \phi(\mu_{l\cdot}^{(i)})$, which conceptually measures the utility obtained from the ith BP, and therefore serves as the baseline relative to SOCU. From this perspective, if we consider SOCU as an "absolute utility," FCCU is better viewed as the "relative utility." Note that when $m \to 1$ for the crisp clustering case, this item becomes a constant, and FCCU reduces to SOCU accordingly. This implies that FCCU is essentially more general than SOCU for FCC. In what follows, we show how to conduct FCC with FCCU.

C. Problem Transformation

Here, we transform the FCC problem into a fuzzy c-means-like problem. We first have the following theorem:

Theorem 1: Suppose U_f is a FCCU derived from a convex function ϕ. Let

$$v_k^{(i)} = \sum_{l=1}^{n} \frac{(\mu_{lk})^m}{\|\mu_{\cdot k}^m\|_1} \mu_{l\cdot}^{(i)} \tag{19}$$

and

$$f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) = \phi(\mu_{l\cdot}^{(i)}) - \phi(v_k^{(i)}) - (\mu_{l\cdot}^{(i)} - v_k^{(i)})^T \nabla\phi(v_k^{(i)}). \tag{20}$$

Then,

$$\sum_{k=1}^{K}\sum_{l=1}^{n} (\mu_{lk})^m f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) = -U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m). \tag{21}$$

Proof: If we substitute f in the left-hand side of (21) by f in (20), we have

$$\sum_{k=1}^{K}\sum_{l=1}^{n} (\mu_{lk})^m f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) = A - B \tag{22}$$

where

$$A = \sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 \phi(\mu_{l\cdot}^{(i)}) - \sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \phi(v_k^{(i)}) \tag{23}$$

and

$$B = \sum_{k=1}^{K} \left(\sum_{l=1}^{n} (\mu_{lk})^m \left(\mu_{l\cdot}^{(i)} - v_k^{(i)}\right)^\top \right) \nabla\phi(v_k^{(i)}). \tag{24}$$

From (19), $\sum_{l=1}^{n} (\mu_{lk})^m v_k^{(i)} = \sum_{l=1}^{n} (\mu_{lk})^m \mu_{l\cdot}^{(i)}$, which indicates $B = 0$. Furthermore, it is easy to verify that $v_k^{(i)} = P_k^{(i)}$ by (15), then we have $A = -U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m)$ by (18), which finally gives (21). ■

Furthermore, if we let

$$J_{mf}(\boldsymbol{\mu}, \boldsymbol{v}; \Pi) = \sum_{k=1}^{K}\sum_{l=1}^{n} (\mu_{lk})^m \left(\sum_{i=1}^{r} w_i f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) \right) \tag{25}$$

it is easy to infer from Theorem 1 that:

Corollary 1: The problem of FCC with FCCU is equivalent to the problem of minimizing J_{mf}, i.e.,

$$\max_{\boldsymbol{\mu}} \sum_{i=1}^{r} w_i U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m) \Longleftrightarrow \min_{\boldsymbol{\mu}, \boldsymbol{v}} J_{mf}(\boldsymbol{\mu}, \boldsymbol{v}; \Pi). \tag{26}$$

Corollary 1 shows us that the minimization of J_{mf} seems very similar to the well-known fuzzy c-means problem (FCM) [5]. This implies that the FCC with FCCU might be equivalently transformed into a fuzzy c-means-like problem, which can then be solved quickly by the iterative algorithm of fuzzy c-means. We verify this point in the section to follow.

D. Algorithmic Design

In this section, our attention shifts to the problem of $\min J_{mf}$. In fact, if we take a closer look at J_{mf} in (25), we can easily find that it is very similar to the objective function of a fuzzy c-means clustering, except that the "data-centroid" distance $\sum_{i=1}^{r} w_i f(\mu_{l\cdot}^{(i)}, v_k^{(i)})$ seems a bit "weird"—we cannot easily figure out the data objects and the centroids from the distance formulation.

To better understand this, we introduce the concept: *multi-membership data*. Let $\mathcal{Y} = \{y_1, \ldots, y_n\}$ be a multimembership dataset derived from the set of BPs $\Pi = \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(r)}\}$, with

$$y_l = (\mu_{l\cdot}^{(1)}, \ldots, \mu_{l\cdot}^{(i)}, \ldots, \mu_{l\cdot}^{(r)})^\top \quad \forall\, l. \tag{27}$$

Apparently, y_l corresponds to x_l, the lth data object in $\mathcal{X}$, in that it aligns the membership degrees of x_l to all the clusters of every BP. As a result, y_l is a $(\sum_{i=1}^{r} K_i)$-dimensional vector with $\|y_l\|_1 = r\,\forall\, l$.

Next, we define the *piecewise centroid* for the fuzzy clustering of $\mathcal{Y}$. Let $\boldsymbol{v} = \{v_1, \ldots, v_K\}$, with

$$v_k = (v_k^{(1)}, \ldots, v_k^{(i)}, \ldots, v_k^{(r)})^\top \tag{28}$$

and $v_k^{(i)}$ given in (19) $\forall\, k, i$. Apparently, v_k is also a $(\sum_{i=1}^{r} K_i)$-dimensional vector constituted by r pieces. More importantly, according to (19), $v_k^{(i)}$ can be viewed as the kth centroid of $\mathcal{Y}$ in

Algorithm 1: **piFCM**

1. $l \leftarrow 0$;
2. Initialize $\boldsymbol{\mu}_{[l]}$, and let $\boldsymbol{v}_{[l]} = G(\boldsymbol{\mu}_{[l]})$; // see Eq. (31)
3. **repeat**
4. $l \leftarrow l + 1$;
5. Let $\boldsymbol{\mu}_{[l]} = F(\boldsymbol{v}_{[l-1]})$; // see Eq. (30)
6. Let $\boldsymbol{v}_{[l]} = G(\boldsymbol{\mu}_{[l]})$; // see Eq. (31)
7. **until** some stopping criterion is met;

Fig. 1. Pseudocodes of piFCM.

the ith piece. That is why we call v_k the piecewise centroid for $\mathcal{Y}$. Note that $v_k^{(i)}$ is also a discrete distribution with $\|v_k^{(i)}\|_1 = \|P_k^{(i)}\|_1 = 1; \forall\, k, i$. As a result, we have $\|v_k\|_1 = r, \forall\, k$.

Given the multimembership data and the piecewise centroid, we can now understand J_{mf} more deeply from a fuzzy c-means clustering perspective. From (25), to minimize J_{mf} is to take a fuzzy c-means clustering on the multimembership data $\mathcal{Y}$, with the kth centroid being the piecewise centroid $v_k\ \forall\, k$, and the distance from y_l to v_k being

$$d(y_l, v_k) = \sum_{i=1}^{r} w_i f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) \quad \forall\, l, k. \tag{29}$$

Based on the above reasoning, we can formulate a FCM-like algorithm to minimize J_{mf}. Without loss of generality, we consider the nondegeneration case of $\boldsymbol{\mu}$. Let

$$\mu_{lk} = \frac{d(y_l, v_k)^{-1/(m-1)}}{\sum_{k'=1}^{K} d(y_l, v_{k'})^{-1/(m-1)}} \quad \forall\, l, k \tag{30}$$

and

$$v_k^{(i)} = \sum_{l=1}^{n} \frac{(\mu_{lk})^m}{\|\mu_{\cdot k}^m\|_1} \mu_{l\cdot}^{(i)} \quad \forall\, k, i. \tag{31}$$

We then have $\boldsymbol{\mu} = [\mu_{lk}]_{n\times K} = F(\boldsymbol{v})$ and $\boldsymbol{v} = \{v_k\}_{k=1}^{K} = G(\boldsymbol{\mu})$, where F and G are two mapping functions defined by (30) and (31), respectively. A FCM-like algorithm called *piecewise FCM* (piFCM) based on F and G can then be given in Fig. 1.

It is easy to see that piFCM is very similar to FCM, by iteratively optimizing $\boldsymbol{\mu}$ and $\boldsymbol{v}$ in Lines 5 and 6 of the pseudocodes. Moreover, it is noteworthy that when we set equal weights to all w_i and let f be the squared l_2-norm, then $d(y_l, v_k)$ in (29) reduces to the well-known squared Euclidean distance, and $\boldsymbol{\mu}$ in (30) and $\boldsymbol{v}$ in (31) thus become the same solution for FCM (in [5, , pp. 67–69]). As to the degeneration case, i.e., where there exists $d(y_l, v_k) = 0$ for some l and k, the treatment for piFCM can be exactly the same as for FCM. That is, let $I_k = \{l | d(y_l, v_k) = 0\}$, then if $I_k \neq \emptyset$, we have $\mu_{lk} = 0\ \forall\, l \in \tilde{I}_k$, and $\mu_{lk} = 1/|I_k|\ \forall\, l \in I_k$.

Now the only problem remaining is whether piFCM can have a global convergence using the two-phase iterative process in Fig. 1. The answer is certainly positive, but the proof can be cumbersome if we begin with the Zangwill's theorem on judging the global convergence of an algorithm [43]. We therefore take an indirect way for proof here. Specifically, we have the following theorem:

Algorithm 2: **FCC**

1. Generate the multi-membership data $\mathcal{Y}$ from $\Pi = \{\boldsymbol{\mu}^{(1)}, \cdots, \boldsymbol{\mu}^{(r)}\}$;
2. Set the weights $\{w_i\}_{i=1}^{r}$ and derive f from some convex function ϕ by Eq. (20);
3. Call the piFCM algorithm;

Fig. 2. Pseudocodes of FCC.

Theorem 2: PFCM has a global convergence; that is, either the iteration sequence $((\boldsymbol{\mu}_{[l]}, \boldsymbol{v}_{[l]}))_{l=0}^{\infty}$ or one of its subsequences converges to a point in the solution set.

Proof: Recall $d(y_l, v_k)$ in (29). If we substitute $f \times f$ in (20), and define a new function on $x = (x^{(1)}, \ldots, x^{(r)})^\top$:

$$\Phi(x) = \sum_{i=1}^{r} w_i \phi(x^{(i)}) \tag{32}$$

we have for all l, k,

$$d(y_l, v_k) = \Phi(y_l) - \Phi(v_k) - (y_l - v_k)^\top \nabla\Phi(v_k). \tag{33}$$

Since ϕ is a convex function, it is obvious that Φ is also a convex function. Thus, (33) indicates that the data-centroid distance of piFCM is actually a *point-to-centroid distance*, and hence piFCM belongs to the family of so-called *Generalized Distance-based FCM* (GD-FCM) algorithms, whose global convergence has been given by [17, Th. 8]. Therefore, the global convergence of piFCM is guaranteed. ■

Until now, we can formulate the complete algorithm for FCC with FCCU. Fig. 2 shows the pseudocodes of the FCC algorithm. By generating the multimembership data $\mathcal{Y}$, we successfully transform the problem of FCC with FCCU into the problem of piFCM clustering, which essentially a FCM-like algorithm employing a two-phase iterative process to obtain the final fuzzy consensus partition. Since the time cost for generating $\mathcal{Y}$ can be ignored, FCC has a relatively low computational complexity similar to FCM.

Note that, although the inputs and output of FCC are all membership matrices, it can easily adapt to crisp partitions by: 1) transforming crisp partitions into membership matrices with binary values, and 2) let the fuzzy factor $m = 1$. In this sense, FCC is actually a very general algorithm that can handle both crisp and soft consensus clustering.

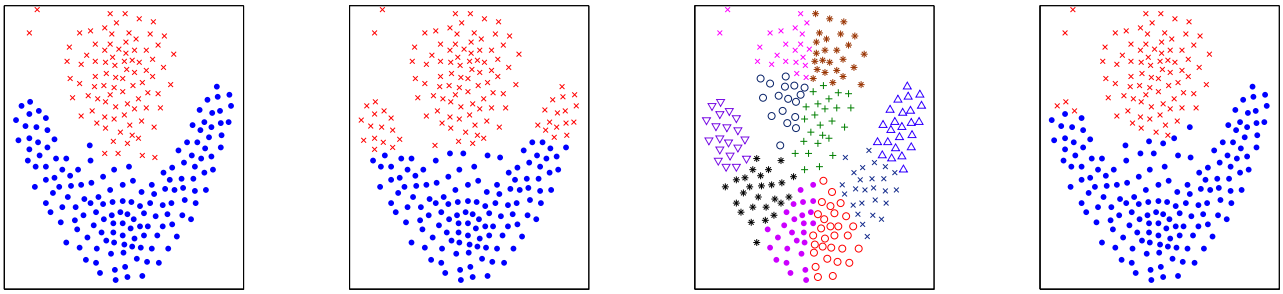
E. Examples of FCCU

Table IV shows some examples of FCCU (see the column of U_f) derived from some commonly used convex function ϕ's, and their corresponding point-to-centroid distances (see the column of d). It is obvious that FCCU is a big family of utility functions specially suitable for FCC. In other words, the employment of FCCU enables the flexibility of FCC defined by (16).

TABLE IV
EXAMPLES OF FCCU WITH CORRESPONDING DISTANCES

Symbol	$\phi(x)$	$U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m)$	$d(y_l, v_k)$
U_{fE}	$\|x\|_2^2$	$\sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \|P_k^{(i)}\|_2^2 - \sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 \|\mu_{l\cdot}^{(i)}\|_2^2$	$\sum_{i=1}^{r} w_i \|\mu_{l\cdot}^{(i)} - v_k^{(i)}\|_2^2$
U_{fH}	$-H(x)$	$\sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 (-H(P_k^{(i)})) - \sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 (-H(\mu_{l\cdot}^{(i)}))$	$\sum_{i=1}^{r} w_i D(\mu_{l\cdot}^{(i)} \| v_k^{(i)})$
U_{fCOS}	$\|x\|_2$	$\sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \|P_k^{(i)}\|_2 - \sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 \|\mu_{l\cdot}^{(i)}\|_2$	$\sum_{i=1}^{r} w_i (1 - \cos(\mu_{l\cdot}^{(i)}, v_k^{(i)}))$
U_{fLP}	$\|x\|_p$	$\sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \|P_k^{(i)}\|_p - \sum_{l=1}^{n} \|\mu_{l\cdot}^m\|_1 \|\mu_{l\cdot}^{(i)}\|_p$	$\sum_{i=1}^{r} w_i \left(1 - \frac{\sum_{j=1}^{K_i} \mu_{lj}^{(i)} (v_{kj}^{(i)})^{p-1}}{\|v_k^{(i)}\|_p^{p-1}}\right)$

Note: 1) x must be a discrete distribution; 2) $\{w_i\}_{i=1}^r$ are the weights of BPs; 3) $p \geq 2$, $p \in \mathbb{Z}_{++}$; H denotes the Shannon entropy [44]; 5) D denotes the KL-divergence [45].



(a) (b) (c) (d)

Fig. 3. Illustrative example: *flame*. (a) The *flame* Data. (b) FCM Result: $c = 2$. (c) FCC Result. (d) FCM Result: $c = 10$.

Note that the point-to-centroid distance for U_{fE} is the (weighted summation of) squared Euclidean distance widely adopted in the fuzzy clustering area, but the other three members of the FCCU family have not been mentioned as much. Specifically, U_{fH} is derived from the Shannon entropy H [44], and the corresponding distance is the well-known KL-divergence [45] widely used for information-theoretic text clustering [46]. Likewise, the cosine similarity for $U_{f\text{COS}}$ is another widely used distance in spherical K-means clustering for texts [47]. If we generalize the cosine similarity to l_p-norm, we get $U_{f\text{LP}}$ with $U_{f\text{COS}}$ being its special case at $p = 2$.

F. Illustrative Example of FCC

Here, we give a small example to illustrate how to conduct FCC to find a better consensus partition. A synthetic dataset *flame* is used, which consists of two-dimensional data points of two categories, as shown in Fig. 3(a). Note that these two classes get very close to each other and linearly inseparable, bringing challenges to clustering.

We first employ fuzzy c-means directly on *flame* with $c = 2$, and obtain the results shown in Fig. 3(b) after hardening the membership degree matrix (i.e., assigning data points to the cluster with the highest membership degree). As expected, quite a few *blue dots* are misclassified as *red crosses* due to the unusual shape of the data and the high closeness of the two classes. To remedy this, one simple yet effective idea is to increase c to a certain level so that the whole dataset can be partitioned clearly. Fig. 3(c) shows the result after setting $c = 10$. With a high c, the data points misclassified in Fig. 3(b) are now separated from the upper class successfully. If we take this as a BP and run FCC to fuse the fragmented yet clear structures, we can expect a much better clustering performance.

Along this line, we now employ FCC on *flame*. We first select squared Euclidean distance as the distance function for FCM, and then run FCM ten times with the randomly selected cluster number $c \in [3, 12]$ to obtain ten fuzzy BPs. We then call piFCM with the utility function U_{fE} and $c = 2$ to obtain the final consensus partition, with equal weights for all the BPs. The result is shown in Fig. 3(d) after hardening the membership degree matrix. As can be seen, FCC performs nearly perfectly on *flame*, which well demonstrates the advantage of consensus clustering to single clustering.

VI. HANDLING BIG DATA

Big data clustering is often very challenging. For FCC, it is difficult to obtain BPs on large high-dimensional data, which also brings great trouble to the subsequent consensus clustering. In this section, we propose a big data computing framework for FCC based on the ability of FCC in handling *incomplete basic partitions* (IBP). Finally, the parallelized implementation of FCC on the Spark platform is given.

A. Framework

The difficulty of big data clustering usually comes from two sources: the high dimensionality and the huge data volume. Benefiting from the consensus learning scheme of FCC, we can employ the so-called *vertical segmentation* treatment for original data so as to ease the dimensionality anxiety. In detail, we do

uniform random sampling with or without replacement on the whole feature set to generate data subsets for basic partitioning, which are then input into FCC for consensus partition. In this way, the high dimensionality of original data reduces to sM for each BP (where s is the sampling ratio and M is the original dimensionality), and to $r\bar{K} \ll M$ roughly for consensus partitioning (where r is the number of BPs, and $\bar{K}$ is the average number of clusters in the BPs). Moreover, weakly related or even noise features might be avoided in some data subsets after sampling, which could foster high-quality BPs and hence benefit the consensus partition.

To relieve the volume anxiety, one straightforward solution is to employ a distributed computing platform such as Spark [48]. Moreover, for the extremely huge data, we propose the so-called *horizontal segmentation* treatment, which conducts uniform random sampling with replacement on data objects so as to form data subsets for basic partitioning. The benefit is also twofold: 1) a large dataset is decomposed into several smaller ones, which is of great help to speed up the generation of each BP; 2) like the vertical segmentation, the high dimensionality of the original data is decreased greatly to $r\bar{K}$ for final consensus clustering. Nevertheless, it is worth noting that horizontal segmentation is more "dangerous" than vertical segmentation; that is, it might result in the missing of true cluster structures for instance shortage after sampling, which could seriously degrade FCC (we find this in experiments). Therefore, we suggest using horizontal segmentation only if the data are too large to be fed into the computing platform.

The framework of FCC of big data is thus established as follows: 1) Vertical segmentation and/or horizontal segmentation are first called to form data subsets in the master node of a Spark cluster; 2) Basic partitioning is invoked sequentially in parallel on Spark to obtain multiple BPs; 3) FCC is finally run on Spark in parallel for the final consensus partition. Two points remain unsolved regarding this simple framework. The first one is about the *incomplete basic partition* (IBP), which means a data subset for basic partitioning may contain incomplete data objects due to the vertical or horizontal sampling. Accordingly, FCC should have the ability to handle IBPs during clustering, which will be carefully addressed in Section VI-B. The other one is about how to run basic partitioning and FCC in parallel on *Spark*. We will depict this with details in Section VI-C.

B. Handling IBPs

Here, we study how to handle IBP in FCC. A BP π_i is obtained by clustering a data subset $\mathcal{X}_i \subseteq \mathcal{X}$, $1 \leq i \leq r$, $\bigcup_{i=1}^{r} \mathcal{X}_i = \mathcal{X}$. Given r IBPs in $\Pi = \{\pi_1, \ldots, \pi_r\}$, we need to find a soft K-way consensus partition via FCC. To this end, we first extend the concept of FCCU in Definition 2 as follows:

Definition 3: A fuzzy utility function U_f is called an *Incomplete FCC Utility Function* (IFCCU) for FCC on IBP, if

$$U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m) = \sum_{k=1}^{K} \|\mu_{\cdot k}^m\|_1 \phi(P_k^{(i)}) - \sum_{x_l \in \mathcal{X}_i} \|\mu_{l\cdot}^m\|_1 \phi(\mu_{l\cdot}^{(i)}) \tag{34}$$

with $P_k^{(i)} = (\sum_{x_l \in \mathcal{X}_i} \frac{(\mu_{lk})^m}{\|\mu_{\cdot k}^m\|_1} \mu_{l\cdot}^{(i)})^\top$, and ϕ being a convex function defined on discrete distributions.

IFCCU is certainly more general than FCCU defined in (18). That is, IFCCU can reduce to FCCU when $\mathcal{X}_i = \mathcal{X}\ \forall i$. We then have the following theorem:

Theorem 3: Suppose U_f is an IFCCU. Let

$$J_{mf}(\boldsymbol{\mu}, \boldsymbol{v}) = \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_l \in \mathcal{X}_i} w_i (\mu_{lk})^m f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) \tag{35}$$

where w_i is the weight of the ith BP, and

$$v_k^{(i)} = \sum_{x_l \in \mathcal{X}_i} \frac{(\mu_{lk})^m}{\|\mu_{\cdot k}^m\|_1} \mu_{l\cdot}^{(i)}. \tag{36}$$

Then, we have

$$J_{mf}(\boldsymbol{\mu}, \boldsymbol{v}) = -\sum_{i=1}^{r} w_i U_f(\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}; m). \tag{37}$$

The proof is similar to Corollary 1 and thus can be omitted here. Given Theorem 3, FCC on IBPs can also transform to piFCM, by simply modifying the distance to $d(y_l, v_k) = \sum_{i=1}^{r} w_i I(x_l \in \mathcal{X}_i) f(\mu_{l\cdot}^{(i)}, v_k^{(i)})$, where $I(x_l \in \mathcal{X}_i) = 1$ if $x_l \in \mathcal{X}_i$, and 0 otherwise. The convergence of FCC on IBPs is also guaranteed by observing the following fact.

Let $z = (z_1, \ldots, z_r)^\top$ be any possible centroid vector, we have

$$\begin{aligned}
\Delta &= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_l \in \mathcal{X}_i} w_i (\mu_{lk})^m f(\mu_{l\cdot}^{(i)}, z_i) - w_i (\mu_{lk})^m f(\mu_{l\cdot}^{(i)}, v_k^{(i)}) \\
&= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_l \in \mathcal{X}_i} (\mu_{lk})^m (\phi(v_k^{(i)}) - \phi(z_i) + (\mu_{l\cdot}^{(i)} - z_i)^\top \nabla \phi(z_i)) \\
&= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_l \in \mathcal{X}_i} (\mu_{lk})^m f(v_k^{(i)}, z_i) \geq 0
\end{aligned} \tag{38}$$

which indicates that the objective function of FCC on IBPs will decrease in the updating phase of piFCM. This is the key factor that drives the convergence of piFCM in Theorem 2. In practice, some stopping criteria, like the maximum iteration number or the difference between two successive iterations below certain threshold, can also help to prematurely finish the clustering process.

C. Parallelization on Spark

Now we show how to parallelize FCC on Spark, a very popular in-memory cluster computing framework [48]. As indicated by Algorithm 1, FCC is essentially a two-phase iterative heuristic similar to the well-known K-means algorithm with data assignment and centroid updating successively. Accordingly, we implement FCC on Spark by designing the MapReduce procedure as follows.

In detail, a dataset is first horizontally split and distributed to every computational node. As the centroids are broadcast to all nodes, each node needs to complete the data assignment only for a part of instances using (30), which is encapsulated in the

TABLE V
EXPERIMENTAL DATASETS

ID	Data Sets	Source	#Instance	#Feature	#Class	CV
1	*wine*	UCI	178	13	3	0.194
2	*dermatology*	UCI	358	33	6	0.509
3	*libras*	UCI	360	90	15	0.000
4	*breast_w*	UCI	699	9	2	0.439
5	*satimage*	UCI	4435	36	6	0.430
6	*pendigits*	UCI	10992	16	10	0.042
7	*tr12*	TREC	313	5804	8	0.638

map function. The *reduce* phase collects averaged vectors for each cluster from all map nodes, updates centroid vectors using (31), and finally broadcasts to all nodes again. A new iteration can then be started.

Note that on Spark, all data including the input matrix, centroid vectors, and the membership-degree matrix are fully loaded into the *resilient distributed datasets* (RDDs). By virtue of RDDs residing in memory, the I/O, and communication costs can be significantly saved during two-phase iterations, which is a big advantage provided by the Spark. In addition, to implement FCC on Spark, the horizontal split is also done for the fuzzified contingency matrix, which does not conflict with both vertical and horizontal samplings for generating BPs. Finally, it is similar to implement FCM or FCC on Spark, with varying distance functions given in Table IV. The major difference is the input data: FCM is called on original data to generate BPs, while FCC is called on the matrix $\Pi = (\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(r)})$ to generate the consensus partition.

VII. EXPERIMENTAL VALIDATION

In this section, we provide experimental results on seven real-world datasets. Specifically, we demonstrate: 1) the clustering quality of FCC with different utility functions, 2) the impact of BPs on FCC, and 3) the effectiveness of FCC in handling IBPs.

A. Experimental Setup

Data sets: We use seven real-world testbed datasets publicly available in our experiments. Some characteristics of these datasets are shown in Table V, where "CV" is the coefficient of variation [49] used to characterize the class imbalance of the datasets. A larger CV indicates a severer class imbalance in data.

Default settings: To generate multiple BPs, we use the FCM algorithm and adopt a *random parameter selection* (RPS) strategy. That is, in each basic clustering, we randomize the number of clusters within $[K', \sqrt{n}]$ for FCM, where K' is the number of true clusters and n is the data size. The number of clusters for FCC is just the number of true clusters K', and the fuzzifier m is set to 2 by default for both FCM and FCC. We then employ three kinds of utility functions for FCC including U_{fH}, $U_{f\cos}$ and U_{fE} (see Table IV).

Validation measure: Since we have crisp class labels for all the experimental data, we use the external measure for cluster validity [50]. In the literature, it has been proven that the normalized Variation of Information (*NVI*) measure shows merits in evaluating the clustering performance of K-means [41]. Hence in our study, we adopt the fuzzy version of *NVI* suggested by [17] as follows. Suppose we have the membership degree matrix $\boldsymbol{\mu} = [\boldsymbol{\mu}_{\cdot k}], 1 \le k \le K$, as the result of K-way fuzzy clustering on a dataset with n instances and K' classes. We then compute the summarized matrix $\boldsymbol{Z}$ as follows: $z_{kj} = \sum_{x_l \in C'_j} \mu_{lk}$, $1 \le k \le K, 1 \le j \le K', 1 \le l \le n,$, where C'_j denotes the jth class in the ground truth. Let $z_{k\cdot} = \sum_{j=1}^{K'} z_{kj}$, $z_{\cdot j} = \sum_{k=1}^{K} z_{kj}$, $p_{k\cdot} = z_{k\cdot}/n$, $p_{\cdot j} = z_{\cdot j}/n$ and $p_{kj} = z_{kj}/n$. We then have

$$NVI = \frac{H(C|C') + H(C'|C)}{H(C) + H(C')} = 1 + \frac{2\sum_{k=1}^{K}\sum_{j=1}^{K'} p_{kj}\log(p_{kj}/p_{k\cdot}p_{\cdot j})}{\sum_{k=1}^{K} p_{k\cdot}\log p_{k\cdot} + \sum_{j=1}^{K'} p_{\cdot j}\log p_{\cdot j}}. \quad (39)$$

It is easy to show $NVI \in [0, 1]$, and a larger NVI value indicates a poorer clustering performance. Note that NVI differentiates itself from the traditional fuzzy measures such as Partition Coefficient and Classification Entropy [51] by using the external class information, which is considered more objective in recent studies [52], [53].

B. Overview of Clustering Performances

We first watch the clustering performance of FCC on the testbed data. To this end, we leverage RPS to generate $r = 40$ BPs as the input of FCC. We run both FCC and FCM 100 times, respectively, and return the average clustering results for each dataset, as listed in Table VI. Note that FCM is taken here as both the generator of BPs and the baseline of FCC, whose distance function is selected from Table IV and corresponds to the utility function used by FCC for the comparison purpose. The significance levels of the improvements of FCC against FCM are also given in Table VI by means of the Independent Sample T-Test [54].

Table VI conveys some really interesting information. The first observation is that FCC beats FCM significantly on nearly all datasets regardless of what utility function is used. For datasets *dermatology*, *breast_w* and *pendigits*, the maximal improvement of clustering quality by FCC goes even beyond 50%, which well demonstrates the advantage of consensus clustering over traditional single clustering. As an extreme case, FCM fails entirely on the text dataset *tr12*, with $NVI = 1$ for all utility functions. This is not unusual since the iterative process of FCM easily gets stuck for high-dimensional data. In contrast, FCC with $U_{f\cos}$ has a *NVI* value of 0.56 on *tr12*, indicating a much better partition and a great benefit from consensus learning.

The second observation is that the performance of FCC is subject to the choice of utility functions. As can be seen, FCC with U_{fH} or $U_{f\cos}$ outperforms FCC consistently with U_{fE} in almost all scenarios except for the dataset *satimage*. This might be surprising to some researchers, since in the literature the corresponding distance function of U_{fE}, i.e., the squared Euclidean distance, is the most widely used distance function for FCM. However, our result implies that for FCC, using KL-divergence (U_{fH}) or cosine similarity $U_{f\cos}$ as approximation function might be a better choice.

TABLE VI
CLUSTERING PERFORMANCES OF FCC ON TESTBED DATA (BY *NVI*)

Data sets	Utility Functions								
	U_{fH}			$U_{f\cos}$			U_{fE}		
	FCM	FCC	Improv.	FCM	FCC	Improv.	FCM	FCC	Improv.
wine	0.7456	0.5898	20.9%***	0.7350	0.6226	15.3%**	0.6886	0.8306	−17.0%***
dermatology	0.8649	0.2752	68.2%***	0.8837	0.2598	70.6%***	0.8805	0.4150	52.9%***
libras	0.9583	0.6033	37.0%***	0.9520	0.5971	37.3%***	0.9708	0.8240	15.1%***
breast_w	0.9750	0.2649	72.8%***	0.9729	0.3087	68.3%***	0.5546	0.5426	2.2%***
satimage	0.7999	0.4853	39.3%***	0.8091	0.6204	23.3%***	0.7612	0.4755	37.5%***
pendigits	0.9978	0.4404	55.9%***	0.8713	0.4870	44.1%***	0.8944	0.7196	19.5%***
tr12	1	0.7245	27.5%***	1	0.5568	44.3%***	1	0.7351	26.5%***
Avg.	0.9452	0.5839	38.3%	0.9354	0.5197	43.75%	0.8959	0.6834	22.5%

Note: p-value $\leq$0.001 "$***$", p-value $\leq$0.01 "$**$", p-value $\leq$0.05 "$*$".

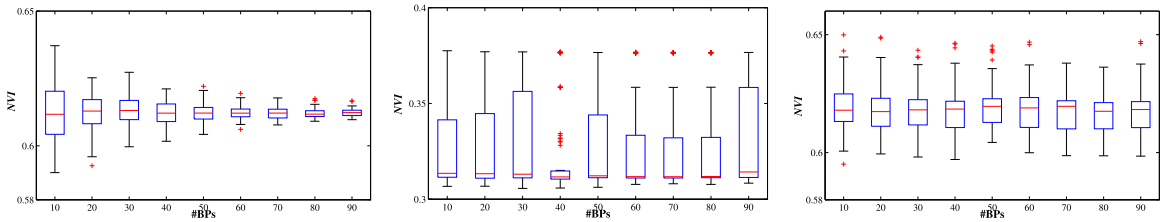


Fig. 4. Impact of the number of BPs to FCC. (a) wine, (b) dermatology, (c) libras.

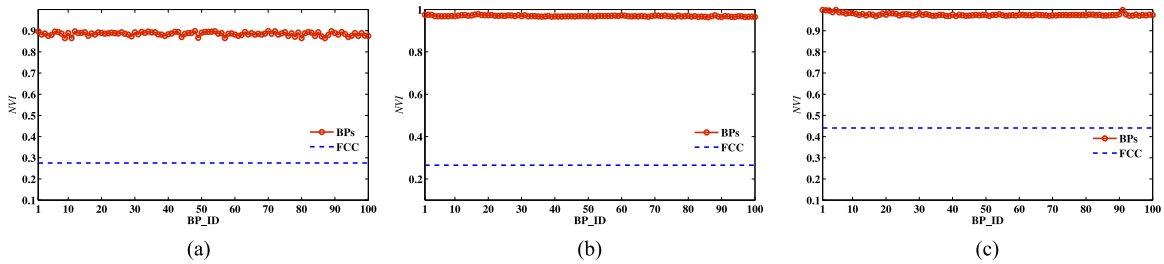


Fig. 5. Quality of BPs in terms of *NVI*. (a) dermatology, (b) breast_*w*, (c) pendigits.

To sum up, FCC exhibits striking advantages over single fuzzy clustering, and the choice of utility functions indeed has great impact to its success. Since FCC with U_{fH} leads to the lowest *NVI* values on four out of seven datasets, we have it as the default choice in the following experiments unless otherwise stated.

C. Effect of BPs

Besides utility functions, the BP is another key factor to the success of FCC. Here, we examine three ingredients of BP that might affect the performance of FCC: 1) the number of BPs, 2) the quality of BPs, and (3) the generation strategies of BPs.

1) Number of BPs: We first utilize the RPS strategy to generate 100 BPs from original data, among which r BPs are randomly selected and input into FCC for the final consensus clustering. We increase r gradually from 10 to 90 with 10 as the interval. For each r, we repeat BP sampling and running FCC 100 times to view the fluctuation of clustering quality.

The box plots in Fig. 4 show the variations of *NVI* under different r on three datasets: *wine*, *dermatology*, and *libras*. As can be seen, FCC seems not very sensitive to the number of BPs in terms of both the average and variation of *NVI* values. This implies that we can set a relatively small r, e.g., 40 in many experimental cases, for FCC to gain higher efficiency yet without sacrificing its accuracy.

2) Quality of BPs: It isintuitive that the quality of BPs could have great impact to the performance of FCC. At the extreme case, if all BPs capture the true cluster structure of original data successfully, we could expect a perfect consensus partition from FCC. The real-world situation is often very tough—in many cases, we can hardly have one okay BP! We here explore whether FCC will fail with the majority of poor BPs. Three datasets, i.e., *dermatology*, *breast_w*, and *pendigits*, are selected for this experiment, and the number of BPs is set to 100 for robustness.

Fig. 5 shows the clustering quality of both BPs and consensus parition on each dataset. As can be seen, for all the three

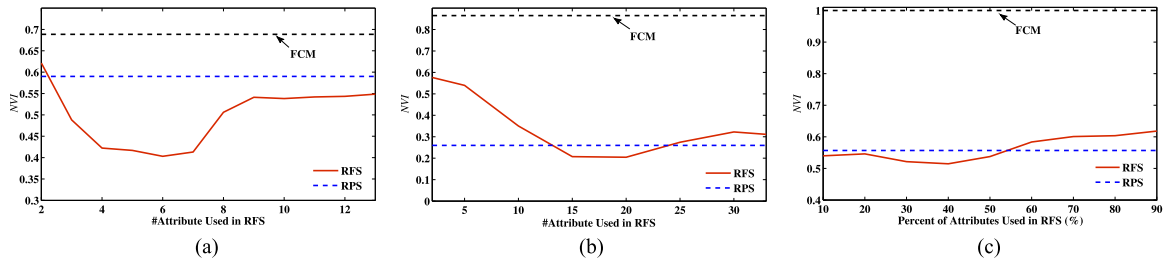


Fig. 6. Impact of RFS strategy to FCC. (a) wine, (b) dermatology, (c) *tr12*.

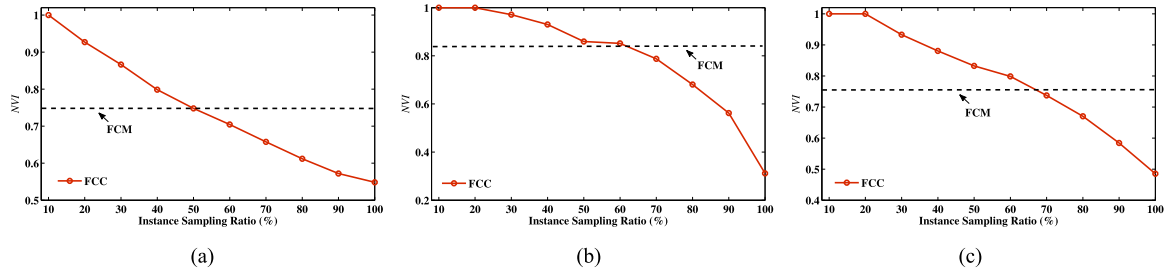


Fig. 7. Performances of FCC on IBPs. (a) wine, (b) dermatology, (c) satimage.

datasets, the BPs are extremely poor—none of them can have a *NVI* value below 0.85. However, as indicated by the dash horizontal lines, the consensus partitions achieved by FCC are far more better than the BPs, each with a substantial *NVI* improvement over 50%. This result implies that the quality of BPs is not the sole deterministic factor to the success of FCC. Rather, the complementarity of BPs capturing some local true cluster information of data, or we may better say the *diversity* of BPs, also contributes greatly to the excellent performance of FCC, although usually in a latent manner.

3) Generation strategy of BPs: The above experiments all make use of RPS to generate BPs. As an alternative, we can also employ the *random feature selection* (RFS) scheme as the generation strategy. RFS is essentially the vertical segmentation scheme proposed in Section VI, which forms a data subset by taking all data objects but a certain number of features randomly sampled from original data. We generate 40 BPs for each sampling ratio of features, with the number of clusters set to the true one.

Fig. 6 shows the performance of the RFS strategy by varying the number of selected features on three datasets *wine*, *dermatology*, and *tr12*. Two observations are noteworthy from Fig. 6. First, compared with FCM on full features and FCC using RPS, FCC using RFS exhibits surprisingly good performances even with a very small set of features. For instance, FCC achieves a much better consensus partition on *wine* when only four features are selected for basic partitioning. Also, for text data *tr12*, FCC with only 10% sampled features outperforms FCC running on all features, and achieves the best clustering quality when sampling 40% features. These results indeed highlight the special potentials of FCC with RFS in handling datasets with noisy features (like *wine*) or sparse features (like text data *tr12*). The second observation is more interesting: the U-shape performances of FCC with RFS indicate that there exists an optimal interval for the feature sampling ratio for each dataset. According to the empirical results, FCC with RFS can obtain the best or almost the best performances on all datasets when only 40% to 50% features are sampled. Fewer features might result in information loss, whereas more features might bring in extra noises.

In summary, the above experiments clearly demonstrate that the generation strategy of BPs exerts great impact to the clustering performance of FCC. RPS with default settings can serve as the primary choice for FCC. However, when facing the dataset containing noisy features or consistently sparse features (e.g., text data), RFS becomes a good alternative.

D. Performance on IBPs

Here, we observe the performance of FCC in dealing with IBPs. To this end, we employ the horizontal segmentation proposed in Section VI; that is, we first randomly sample a certain percentage of data objects (denoted as $s\%$) from a whole dataset, and then invoke FCC with RPS on this incomplete data subset to generate an IBP. For each given s, we repeat sampling and clustering 40 times to obtain multiple IBPs for the final consensus clustering. We increase s gradually from 10 to 100 with 10 as the interval, and watch the fluctuation of performances of FCC.

Fig. 7 reports the results on *wine*, *dermatology*, and *satimage*. Generally speaking, the sampling ratio s shows great impact to the performance of FCC. As can be seen, with the increase of s, *NVI* drops rapidly and indicates better clustering quality; when s goes beyond 60, the performance of FCC begins to be comparable to the performance of FCM; as s rises further, the clustering quality of FCC improves even faster, getting closer to the performance of FCC on complete BPs.

In summary, the above results demonstrate the ability of FCC in handling IBPs, which indicates the feasibility of horizontal segmentation when leveraging FCC for big data. However, the results also suggest setting a relatively large sampling ratio to ensure a good performance.

TABLE VII
IMPACT OF RFS ON EFFICIENCY

Method	RDD_Size	Time/Iter.	#Iteration	Total Time
K-means wtih RFS*	1.96 GB	≈1.2 min.	≈12	≈18 min.
K-means without RFS*	3.71 GB	≈14 min.	>30	>7 h
FCC on BPs	5.20 GB	≈50 s	>30	≈28 min.
FCM on Original Data	10.57 GB	≈20 min.	>30	>9 h

*This is the time consumption for one basic partitioning.

VIII. REAL-LIFE APPLICATIONS

In this section, we apply FCC on Spark to big data clustering applications. Two large-scale datasets are adopted for finding meaningful topics from one Weibo corpus and detecting overlapping communities from one massive network, respectively.

A. Topic Detection From Large-Scale Weibo Short Texts

1) Experimental Settings: The first application utilizes a large-scale text corpus collected from Sina Weibo (http://weibo.com/), a Twitter-like platform in China. In 2015, a female journalist released a documentary named "Under the Dome" to Weibo, which stimulated a nation-wide discussions of environment protection in recent China. This text dataset contains tweets that are concerned with this hot event from 28 February to 6 March, 2015. We filter tweets with less than eight words and produce a dataset with 1 873 184 instances and 213 159 terms. Since a tweet is often very short, this dataset is highly sparse in dimensionality.

To conduct big data clustering, we follow the three-step procedure of the framework given in Section VI-A, except that: 1) K-means with cosine similarity is used instead of FCM to generate crisp BPs, and 2) only vertical segmentation is used in advance of basic partitioning to lower the risk of missing true cluster structures. Note that we use K-means here for the purpose of accelerating basic partitioning, and FCC is capable of handling both crisp and soft inputs. For both basic and consensus clustering, the number of clusters is set to 50, and the number of iterations is bounded to 30. The feature sampling ratio is set to 20%, which retains about 42 000 features for each data subset. We repeat basic clustering to obtain ten BPs, which is then fed to FCC with $U_{f\cos}$ for the final consensus clustering.

We implement K-means and FCC by ourselves on Spark that is built upon a cluster with eight nodes connected by a Gigabit Ethernet. Each node comes with four quad-core E5-2650v2 processors (2.6 GHZ), 128 GB of RAM, 240 GB of SSD disk, and 600 GB of SAS disk. We divide RDDs into eight splits and start eight parallel jobs for instance assignment.

2) The results: We first observe the efficiency of FCC on big data. Table VII shows the time consumption using different methods. By comparing the first and second rows, we can see that RFS plays a critically import role in reducing the time costs for BPs generation in big data scenario. As can be seen, if we run K-means on full data (i.e., without RFS) for a BP, it takes about 14 min for one iteration and cannot converge within 30 iterations, and finally consumes approximately 7 h in total. In contrast, K-means with RFS saves nearly half of memory space, achieves a $10\times$ speedup in each iteration, and converges within 15 iterations. As a result, we can generate a BP in about 20 min, which is rather efficient for such a big dataset.

Rows #3 and #4 compare the runtime of FCC on BPs and FCM on the original text data. As can be seen, since the input matrix of FCC is relatively low dimensional (i.e., $rK = 500$) and very sparse with binary values, it takes less than 50 s for one iteration and totally spends about half an hour for 30 iterations. This is in sharp contrast to running FCM directly on the original data, which consumes about 20 min for one iteration and can hardly converge within 30 iterations. In a nutshell, the above results demonstrate the special importance of vertical segmentation to the high efficiency of FCC on big data.

We then observe the effectiveness of FCC in finding high-quality topics from the tweets. Table VIII shows the representative keywords of seven clusters. As can be seen, the overlapped clusters #8, #27, and #45 contain a large number of government news. In particular, cluster# 8 talks about some positive measures have been taken to deal with the haze (e.g., artificial rainfall and haze dispersal), cluster #27 is about the forecasts of air quality in different cities (e.g., the yellow warning for bad weather), and cluster #45 discusses about more strong countermeasures should be taken (e.g., new energy, the relocation of heavily polluting enterprises, etc). Cluster #22 mainly contains comments for Chai Jing's documentary, which further causes people to worry about some social ills. This cluster has some overlaps with cluster #37, which extensively discusses about the national welfare and the people's livelihood. Clusters #19 and #44 are related to the diseases that might be caused by the haze, various health tips to avoid the haze, and even lots of advertisements for equipments (e.g., humidifier and air-purifier).

In summary, with the flexible data segmentation techniques and the parallel computing framework, FCC has exhibited great potentials and exciting advantages in big data clustering.

TABLE VIII
SEMANTICS OF SAMPLE CLUSTERS FOUND BY FCC

No.	Keywords
Clu. 8	Beijing, rainfall, haze dispersal, artificial, vehicles, spokesman, budget
Clu. 27	yellow warning, qualified air, emergency, strong wind, EPA
Clu. 45	regulate, new energy, yellow-label cars, removal, building, petroleum
Clu. 22	dispute, criticism, video, official, report, corruption, society, PO
Clu. 37	livelihood, government, proposals, hatred, justice, responsibility
Clu. 19	respiratory tract, PM, tumour, cancer, lung, children, pregnancy, diet
Clu. 44	humidifier, air-purifier, respirator, dust, PM, pollutant, sales volume

Note: 1) *EPA* means "Environmental Protection Agency"; 2) *PO* means "Procuratorial Organ;" 3) *PM* means "Particulate Matter".

TABLE IX
EFFICIENCY OF PARALLELIZED FCC ON *Pokec* NETWORK

Method	RDD_Size	Time/Iter.	#Iteration	Total Time
FCC on BPs	6.80 GB	≈7 min.	>30	≈2 h
FCM on Original Data	13.36 GB	≈90 min.	>30	>45 h

TABLE X
SEMANTICS OF SAMPLE COMMUNITIES IDENTIFIED BY FCC

No.	#User	Similar Profiles
Com. 138	48	like_comedies (58%), watch_movies_at_home (70%), profession_school (38%)
Com. 2737	53	like_comedies (65%), watch_movies_at_home (67%), profession_school (43%)
Com. 5046	75	disko_party (61%), looking_for_friends (92%), city_Prievidza (46%), relation_to_alcohol (63%)
Com. 8932	85	disko_party (56%), looking_for_friends (96%), city_Prievidza (70%), relation_to_alcohol (52%)

B. Community Detection From Large-Scale Social Network

1) Experimental settings: The second application is carried out on a massive network derived from *Pokec* [55], the most popular online social networking site in Slovakia (http://www.pokec.sk/). Pokec has operated for more than 10 years and maintains its popularity even after the coming of Facebook. The *Pokec* dataset consists of two parts, i.e., a network with 1 632 803 nodes and 30 622 564 edges, and the profiles of all users including gender, age, hobbies, interest, education, etc.

We now employ FCC for discovering overlapping communities [56] from the *Pokec* network. To this end, we first reorganize the network into a huge and highly sparse adjacent matrix, and utilize a scalable community detection tool named *Louvain* [57] for generating crisp BPs. We adjust the different granularity to maximize the modularity inside Louvain so as to obtain ten BPs, each consuming only dozens of seconds. We then use FCC with U_{fCOS} to combine crisp BPs and identify overlapping communities finally. The number of communities is set to 10 000 so that each could be small yet semantically meaningful.

2) Results: The runtime of FCC on BPs and FCM on original data is listed in Table IX. Similar observation can be found: FCC running on BPs is much faster than FCM on the original network. Specifically, FCM takes averagely 1.5 h for each iteration, while FCC only needs about 7 min, achieving a $13\times$ speedup in each iteration.

Next, we validate the quality of identified communities by extracting semantic information from user profiles. We take two pairs of overlapping communities as examples in Table X. As can be seen, Communities #138 and #2737 have 16 users in overlap, accounting for about 30% of total users. By extracting user preferences from the profiles, we can find that a majority of users in both communities have claimed they like watching movies, and some like comedies and watching movies at home as well. Meanwhile, quite a few users are working at school, and lots of users would not like to share their profession information on social networking sites. Communities #5046 and #8932 have 18 users in overlap. They are mainly "noisy" persons who like disko, party, alcohol, and are looking for friends in Pokec. From the *region* attribute, we find that they mainly live in city of Prievidza and most of them are even from the same town *Handlove*.

In summary, FCC on Spark again shows its ability in delivering efficient and effective big-data clustering for the task of detecting overlapping communities from massive networks.

IX. CONCLUSION

This paper contributes to providing a comprehensive study of fuzzy consensus study from a utility perspective, which eventually delivers a clear, efficient, flexible, and robust framework titled FCC. Specifically, based on the novel fuzzified contingency matrix, we derive a family of utility functions for FCC, and transform FCC to a weighted piecewise FCM, which gains high efficiency via an FCM-like iterative process. Moreover, we present both vertical and horizontal segmentation schemes for establishing an elastic framework for FCC on big data clustering, which is further parallelized on the Spark platform. Experiments on various testbed datasets and two real-life large-scale datasets demonstrate the effectiveness and scalability of our FCC framework. To make FCC more practical, in the future, we will investigate the automated estimation of the number of clusters, and employ some internal metrics for cluster validity.

REFERENCES

[1] N. Nguyen and R. Caruana, "Consensus clusterings," in *Proc. 7th IEEE Int. Conf. Data Min.*, Washington, DC, USA, 2007, pp. 607–612.
[2] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2002.
[3] W. Pedrycz and P. Rai, "Collaborative clustering with the use of fuzzy c-means and its quantification," *Fuzzy Sets Syst.*, vol. 159, no. 18, pp. 2399–2427, 2008.
[4] H. Liu, T. Liu, J. Wu, D. Tao, and Y. Fu, "Spectral ensemble clustering," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 715–724.
[5] J. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algoritms*. New York, NY, USA: Plenum Press, 1981.
[6] X. Sevillano, A. F, and J. Socor, "Positional and confidence voting-based consensus functions for fuzzy cluster ensembles," *Fuzzy Sets Syst.*, vol. 193, pp. 1–32, 2012.
[7] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, "Fuzzy c-means algorithms for very large data," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 6, pp. 1130–1146, Dec. 2012.
[8] H. Liu, R. Zhao, H. Feng, F. Cheng, Y. Fu, and Y. Liu, "Entropy-based consensus clustering for patient stratification," *Bioinformatics*, Mar. 2017, doi: 10.1093/bioinformatics/btx167.
[9] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.
[10] J. Wu, H. Liu, H. Xiong, J. Cao, and J. Chen, "K-means-based consensus clustering: A unified view," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 155–169, Jan. 2015.
[11] H. Liu, M. Shao, S. Li, and Y. Fu, "Infinite ensemble for image clustering," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1745–1754.
[12] E. Dimitriadou, A. Weingessel, and K. Hornik, "A combination scheme for fuzzy clustering," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 16, no. 7, pp. 901–912, 2002.
[13] K. Punera and J. Ghosh, "Consensus-based ensembles of soft clusterings," *Appl. Artif. Intell.*, vol. 22, no. 7–8, pp. 780–810, 2008.
[14] J. Dunn, "A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, pp. 32–57, 1973.
[15] R. Dav and R. Krishnapuram, "Robust clustering methods: A unified view," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 270–293, May 1997.

[16] R. Hathaway and J. Bezdek, "Nerf c-means: Non-euclidean relational fuzzy clustering," *Pattern Recognit.*, vol. 27, no. 3, pp. 429–437, Mar. 1994.
[17] J. Wu, H. Xiong, C. Liu, and J. Chen, "A generalization of distance functions for fuzzy c-means clustering with centroids of arithmetic means," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 3, pp. 557–571, Jun. 2012.
[18] R. Dave and S. Sen, "Robust fuzzy clustering of relational data," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, pp. 713–727, Dec. 2002.
[19] P. Hore, L. O. Hall, and D. B. Goldgof, "Single pass fuzzy C means," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2007, pp. 1–7.
[20] L. O. Hall and D. B. Goldgof, "Convergence of the single-pass and online fuzzy c-means algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 792–794, Aug. 2011.
[21] H. Shen, J. Yang, S. Wang, and X. Liu, "Attribute weighted mercer kernel based fuzzy clustering algorithm for general non-spherical datasets," *Soft Comput.*, vol. 10, no. 11, pp. 1061–1073, 2006.
[22] H. Huang, Y. Chuang, and C. Chen, "Multiple kernel fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 1, pp. 120–134, Feb. 2012.
[23] A. Saha and S. Das, "Axiomatic generalization of the membership degree weighting function for fuzzy c means clustering: Theoretical development and convergence analysis," *Inf. Sci.*, vol. 408, pp. 129–145, 2017.
[24] A. Topchy, A. Jain, and W. Punch, "Combining multiple weak clusterings," in *Proc. 3th IEEE Int. Conf. Data Min.*, Melbourne, FL, USA, Nov. 2003, pp. 331–338.
[25] A. Topchy, A. Jain, and W. Punch, "A mixture model for clustering ensembles," in *Proc. 4th SIAM Int. Conf. Data Min.*, FL, USA, 2004, pp. 379–390.
[26] H. Liu, J. Wu, D. Tao, Y. Zhang, and Y. Fu, "Dias: A disassemble-assemble framework for highly sparse text clustering," in *16th Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 766–774.
[27] T. Li, D. Chris, and I. J. Michael, "Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization," *Proc. 7th IEEE Int. Conf. Data Mining*, 2007, pp. 577–582.
[28] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper, "Weighted partition consensus via kernels," *Pattern Recognit.*, vol. 43, no. 8, pp. 2712–2724, 2010.
[29] Z. Lu, Y. Peng, and J. Xiao, "From comparing clusterings to combining clusterings," in *Proc. 23rd AAAI Conf. Artif. Intell.*, Jul. 2008, pp. 361–370.
[30] H. Liu, J. Wu, T. Liu, D. Tao, and Y. Fu, "Spectral ensemble clustering via weighted k-means: Theoretical and practical evidence," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1129–1143, May 2017.
[31] 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.
[32] C. Domeniconi and M. Al-Razgan, "Weighted cluster ensembles: Methods and analysis," *ACM Trans. Knowl. Discovery Data*, vol. 2, no. 4, 2009, Art. no. 17.
[33] H. S. Yoon, S. Y. Ahn, S. H. Lee, S. B. Cho, and J. Kim, "Heterogeneous clustering ensemble method for combining different cluster results," *Data Min. Biomed. Appl.*, vol. 3916, pp. 82–92, 2006.
[34] X. Sevillano, F. Alas, and J. Socor, "Bordaconsensus: A new consensus function for soft cluster ensembles," in *Proc. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2007, pp. 743–744.
[35] F. Crespo and R. Weber, "A methodology for dynamic data mining based on fuzzy clustering," *Fuzzy Sets Syst.*, vol. 150, no. 2, pp. 267–284, 2005.
[36] W. Pedrycz, "A dynamic data granulation through adjustable fuzzy clustering," *Pattern Recognit. Lett.*, vol. 29, no. 16, pp. 2059–2066, 2008.
[37] S. Vega-Pons and J. Ruiz-shulcloper, "A survey of clustering ensemble algorithms," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 25, no. 3, pp. 337–372, 2011.
[38] B. Mirkin, "The problems of approximation in spaces of relationship and qualitative data analysis," *Inf. Remote Control*, vol. 35, pp. 1424–1431, 1974.
[39] B. Mirkin, "Reinterpreting the category utility function," *Mach. Learn.*, vol. 45, no. 2, pp. 219–228, Nov. 2001.
[40] H. Luo, F. Jing, and X. Xie, "Combining multiple clusterings using information theory based genetic algorithm," in *Proc. Int. Conf. Comput. Intell. Security*, 2006, pp. 84–89.
[41] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Paris, France, 2009, pp. 877–886.
[42] J. Jensen, "Sur les fonctions convexes et les inégalités entre les valeurs moyennes," *Acta Math.*, vol. 30, no. 1, pp. 175–193, 1906.
[43] W. Zangwill, *No-Linear Programming: A Unified Approach*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1969.
[44] C. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[45] S. Kullback and R. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[46] J. Cao, Z. Wu, J. Wu, and H. Xiong, "Sail: Summation-based incremental learning for information-theoretic text clustering," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 570–584, Apr. 2013.
[47] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," *Mach. Learn.*, vol. 55, no. 3, pp. 311–331, 2004.
[48] M. Zaharia *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, 2012, pp. 2–2.
[49] M. DeGroot and M. Schervish, *Probability and Statistics*, 3rd ed. Reading, MA, USA: Addison-Wesley, 2001.
[50] I. Sledge, J. Bezdek, T. Havens, and J. Keller, "Relational generalizations of cluster validity indices," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 771–786, Aug. 2010.
[51] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering validity checking methods: Part II," *SIGMOD Rec.*, vol. 31, no. 3, pp. 19–27, 2002.
[52] J. C. Bezdek, M. Moshtaghi, T. Runkler, and C. Leckie, "The generalized c index for internal fuzzy cluster validity," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 6, pp. 1500–1512, Dec. 2016.
[53] A. Suleman, "Assessing a fuzzy extension of rand index and related measures," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 1, pp. 237–244, Feb. 2017.
[54] J. Rice, *Mathematical Statistics and Data Analysis*. 3rd ed. Boston, MA, USA: Cengage Learning, 2006.
[55] L. Takac and M. Zabovsky, "Data analysis in public social networks," in *Proc. Int. Sci. Conf. Int. Workshop Present Day Trends Innov.*, 2012, pp. 1–6.
[56] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using neighborhood-inflated seed expansion," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1272–1284, May 2016.
[57] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

Junjie Wu received the Ph.D. degree in management science and engineering from Tsinghua University in 2008, from where he also received a B.E. degree in civil engineering in 2002. He is currently a Full Professor in Information Systems Department, School of Economics and Management, Beihang University, Beijing, China. He is also the Director of the Research Center for Social Computing, and the Vice-Director of Beijing Key Laboratory of Emergency Support Simulation Technologies for City Operations. His general area of research is data mining, with a special interest in social computing, urban computing, and financial computing. He has been the PI of more than 30 research projects and has published more than 100 papers in refereed conference proceedings and journals such as KDD and TKDE.

Prof. Wu received three nation-wide academic awards in China: the NSFC Distinguished Young Scientist, the MOE Changjiang Young Scholars, and the National Excellent Doctoral Dissertation.

Zhiang Wu (S'14–M'17) received the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2009.

He is currently a Full Professor of Jiangsu Provincial Key Laboratory of E-Business and School of Information Engineering, Nanjing University of Finance and Economics. His recent research focuses on distributed computing, data mining, e-commerce intelligence, and social network analysis. He has published more than 30 refereed journal and conference papers in these areas.

Dr. Wu is the member of the ACM and a senior member of CCF.

Jie Cao received the Ph.D. degree in mechanical engineering from Southeast University, Nanjing, China, in 2002.

He is currently a Chief Professor and the Dean of School of Information Engineering, Nanjing University of Finance and Economics. His main research interests include data mining, big data, and e-commerce intelligence. He has been selected in the Program for New Century Excellent Talents in University and awarded with Young and Mid-aged Expert with Outstanding Contribution in Jiangsu province.

Dr. Cao is a member of the ACM, CCF, and IEEE Computer Society.

Hongfu Liu received the Bachelor's and Master's degree in management information systems from the School of Economics and Management, Beihang University, Beijing, China, in 2011 and 2014, respectively. He is currently working toward the Ph.D. degree in computer science from the Northeastern University, Boston, MA, USA.

His research interests generally focus on data mining and machine learning, with special interests in ensemble learning.

Guoqing Chen received the Ph.D. degree from Catholic University of Leuven, Leuven, Belgium.

He is currently the EMC Chair Professor of Information Systems, School of Economics and Management, Tsinghua University, Beijing, China. He was appointed China National Chang-Jiang Scholars Professor in 2005, and became IFSA Fellow in 2009. His research interests include big data and business analytics, fuzzy logic applications, managerial informatics, and decision support.

Yanchun Zhang received the Ph.D. degree in computer science from The University of Queensland, St Lucia, Australia, in 1991.

He is a Full Professor and the Director of Centre for Applied Informatics, Victoria University, Victoria, Australia, since 2004. His research interests include databases, data mining, web services and e-health. He has published more than 300 research papers in international journals and conference proceedings including TOCHI, TKDE, VLDB, SIGMOD, and ICDE conferences, and a dozen of books and journal special issues in the related areas.

Dr. Zhang is a Founding Editor and the Editor-in-Chief of the *World Wide Web Journal* (Springer), and *Health Information Science and Systems Journal* (BioMed Central), and also the Founding Editor of Web Information Systems Engineering Book Series and Health Information Science Book Series. He is the Chairman of International Web information Systems Engineering Society.

View publication stats