Distributed *k*-Means Algorithm and Fuzzy *c*-Means Algorithm for Sensor Networks Based on Multiagent Consensus Theory

Jiahu Qin, *Member, IEEE*, Weiming Fu, Huijun Gao, *Fellow, IEEE*, and Wei Xing Zheng, *Fellow, IEEE*

***Abstract*—This paper is concerned with developing a distributed *k*-means algorithm and a distributed fuzzy *c*-means algorithm for wireless sensor networks (WSNs) where each node is equipped with sensors. The underlying topology of the WSN is supposed to be strongly connected. The consensus algorithm in multiagent consensus theory is utilized to exchange the measurement information of the sensors in WSN. To obtain a faster convergence speed as well as a higher possibility of having the global optimum, a distributed *k*-means++ algorithm is first proposed to find the initial centroids before executing the distributed *k*-means algorithm and the distributed fuzzy *c*-means algorithm. The proposed distributed *k*-means algorithm is capable of partitioning the data observed by the nodes into measure-dependent groups which have small in-group and large out-group distances, while the proposed distributed fuzzy *c*-means algorithm is capable of partitioning the data observed by the nodes into different measure-dependent groups with degrees of membership values ranging from 0 to 1. Simulation results show that the proposed distributed algorithms can achieve almost the same results as that given by the centralized clustering algorithms.**

***Index Terms*—Consensus, distributed algorithm, fuzzy *c*-means, hard clustering, *k*-means, soft clustering, wireless sensor network (WSN).**

I. Introduction

A WIRELESS sensor network (WSN) consists of spatially distributed autonomous sensors capable of limited computing power, storage space, and communication range to monitor physical or environmental conditions, such as temperature, sound and pressure, and to cooperatively pass on their data through the network to a main location. The development of WSNs was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial applications, for instance, industrial and manufacturing automation, machine health monitoring, and so on [1]–[6].

When using a WSN as an exploratory infrastructure, it is often desired to infer hidden structures in distributed data collected by the sensors. For a scene segmentation and monitoring application, nodes in the WSN need to collaborate with each other in order to segment the scene, identify the object of interest, and thereafter to classify it. Outlier detection is also a typical task for WSN with applications in monitoring chemical spillage and intrusion detection [14].

Many centralized clustering algorithms have been proposed to solve the data clustering problem in order to grasp the fundamental aspects of the ongoing situation [7]–[11]. However, in WSN, large amounts of data are dispersedly collected in geographically distributed nodes over networks. Due to the limited energy, communication, computation and storage resources, centralizing the whole distributed data to one fusion node to perform centralized clustering may be impractical. Thus, there is a great demand for distributed data clustering algorithms in which the global clustering problem can be solved at each individual node based on local data and limited information exchanges among nodes. In addition, compared with the centralized clustering, distributed clustering is more flexible and robust to node and/or link failures.

Clustering can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find a cluster. One of the most popular notions of the clusters is groups with small in-group and large out-group differences [7]–[11]. Different measures of difference (or similarity) may be used to place items into clusters, including distance and intensity. A great deal of algorithms [7]–[10] have been proposed to deal with the clustering problem based on distance. Among them, the *k*-means algorithm is the most popular hard clustering algorithm, which features simple and fast-convergent iterations [7].

Parallel and distributed implementations of the *k*-means algorithm have risen most often because of its high efficiency in dealing with large data sets. The algorithm proposed in [12] is the parallel version of the *k*-means algorithm in multiprocessors of distributed memory, which has no communication constraints. In [13], observations are assigned to

Manuscript received June 24, 2015; revised October 25, 2015 and January 24, 2016; accepted February 2, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61333012 and Grant 61473269, in part by the Fundamental Research Funds for the Central Universities under Grant WK2100100023, in part by the Anhui Provincial Natural Science Foundation under Grant 1408085QF105, and in part by the Australian Research Council under Grant DP120104986. This paper was recommended by Associate Editor R. Selmic.

J. Qin and W. Fu are with the Department of Automation, University of Science and Technology of China, Hefei 230027, China (e-mail: jhqin@ustc.edu.cn; fwm1993@mail.ustc.edu.cn).

H. Gao is with the Research Institute of Intelligent Control and Systems, Harbin Institute of Technology, Harbin 150001, China (e-mail: hjgao@hit.edu.cn).

W. X. Zheng is with the School of Computing, Engineering and Mathematics, Western Sydney University, Sydney, NSW 2751, Australia (e-mail: w.zheng@westernsydney.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/TCYB.2016.2526683

distributed stations until all observations in a station belong to a cluster based on the partition method of the k-means algorithm. It is apparent that a large number of observations need to be exchanged, which would cause the network burden to be heavy. A kind of distributed k-means algorithm is proposed in [14] for peer-to-peer networks, in which each node gathers basic information of itself and then disseminates it to all the others. It can be observed that the algorithm proposed in [14] is not scaling well, since a very large amount of data needs to be stored when the number of nodes is huge. Forero *et al.* [15], [16] treated the clustering problem performed by distributed k-means algorithms as a distributed optimization problem. The duality theory and decomposition methods, which play a major role in the field of distributed optimization [17], are utilized to derive the distributed k-means algorithms. However, an additional parameter is introduced, and furthermore, the clustering results may not converge when an unsuitable parameter is selected. In [18], the distributed consensus algorithm in the multiagent systems community [19] is applied to help develop a distributed k-means algorithm. Although the algorithm proposed in [18] is feasible to accomplish the clustering work in a distributed manner without introducing any parameters, there are still some issues that need to be noted. The first issue is the high computational complexity. The second issue is that it is impossible for the algorithm to work when the network is a directed graph which does happen in practice since different sensors may have different sensing radii. Finally, the hard clustering algorithm is not always suitable because in some situations such as soil science [20] and image processing [21], overlapping classes may be useful in dealing with the observations for which the soft clustering is required. Certainly, there are also some other techniques to tackle the distributed clustering problems, which are not based on the k-means algorithm. For example, [22] presents a type of distributed information theoretic clustering algorithms based on the centralized one. In [23], distributed clustering and learning schemes over networks are considered, in which different learning tasks are completed by different clusters.

We consider in this paper, the framework that the nodes in the WSN hold a (possibly high-dimensional) measure of information and the network is a strongly connected directed graph. To be specific, we aim to extend the algorithms proposed in [18] from the following perspectives.

1) In order to obtain a faster convergence speed as well as a higher possibility of the global optimum, the k-means++ algorithm [24], a kind of centralized algorithm to find the initial centroid of the k-means algorithm, is extended to the distributed one.
2) The distributed k-means algorithm proposed in this paper has a relatively lower computational complexity than the one in [18] and can be used to deal with the condition that the network topology is a directed graph, irrespective of whether the underlying topology of each cluster is connected or not.
3) A distributed fuzzy c-means clustering algorithm is provided for distributed soft clustering.

The remainder of this paper is organized as follows. Some graph theory and the consensus algorithm are introduced in Section II. The k-means clustering algorithm is reviewed and a distributed k-means algorithm is proposed in Section III, while a distributed fuzzy c-means algorithm is developed in Section IV. Section V provides simulation results to demonstrate the usefulness of the proposed algorithms. Finally, some conclusions are made in Section VI.

II. Preliminaries

In order to provide a distributed implementation of the k-means algorithm, we need a tool to diffuse information among the nodes. The average-consensus, max-consensus, and min-consensus algorithms are proved with their effectiveness in composing local observations by means of one-hop communication. In this section, we first introduce some notations of graph theory and then the consensus algorithm.

A. Graph Theory

Let $G = (V, E, A)$ be a weighted directed graph of order n with a set of nodes $V = \{1, \ldots, n\}$, a set of edges $E \subset V \times V$ and a weighted adjacency matrix $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ with nonnegative adjacency element a_{ij}. If there is a directed edge from j to i in the directed graph G, then the edge is denoted by (i, j). The adjacency elements associated with the edges of the graph are positive, i.e., $(i, j) \in E \Leftrightarrow a_{ij} > 0$. The set of neighbors of node i is denoted by $N_i = \{j \in V : (i, j) \in E\}$.

A directed path from node i to node j of G is a sequence of edges like $(i, i_1), (i_1, i_2), \ldots, (i_{l-1}, i_l), (i_l, j)$ whose length is $l + 1$. The distance between two nodes is the length of the shortest path connecting them. A directed graph is strongly connected if there is a directed path from every node to every other node. The diameter D of a strongly connected graph is the maximum value of the distances between all the nodes.

For a directed graph, the in-degree and out-degree of i are, respectively, the sum of the weights of incoming and outgoing edges. A directed graph is weight balanced if for all $i \in V$, $\sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} a_{ji}$, i.e., the in-degree and out-degree coincide.

An undirected graph is a special kind of directed graph in which $a_{ij} = a_{ji}$ for all $i, j \in V$. Apparently, an undirected graph is always weight balanced.

B. Consensus Algorithm

Based on the notations given above, we review the consensus algorithms in this section. A large number of consensus algorithms have been proposed (see [19], [25]–[29] and the references therein), but we only introduce some simple algorithms in this section.

For a set of n nodes associated with a graph G, the discrete dynamics is described by

$$x_i(t + 1) = u_i(N_i \cup \{i\}, t),\ \ x_i(0) = x_{i0}, i = 1, \ldots, n \quad (1)$$

where $x_i(t)$, $x_{i0} \in \mathbb{R}^d$, and $u_i(N_i \cup \{i\}, t)$ is the function of the states of the nodes in the set $N_i \cup \{i\}$. Here the dimensions of the states of nodes are supposed to be 1 for simplification. In fact,

if the state is high-dimensional, then the same operation can be conducted on every component of the state individually.

Let $\mathcal{X}(x_{10}, \ldots, x_{n0}) \in \mathbb{R}$ be any function of the initial states of all the nodes. The $\mathcal{X}$-problem is to seek a proper $u_i(N_i \cup \{i\}, t)$ such that $\lim_{t\to\infty} x_i(t) = \mathcal{X}(x_{10}, \ldots, x_{n0})$ in a distributed way.

In the max-consensus problem, the nodes are required to converge to the maximum of the initial states, i.e., $\mathcal{X}(\cdot)$ is the maximum of its arguments. When the directed graph is strongly connected, the max-consensus can be reached [19] if the following control law is chosen:

$$u_i(N_i \cup \{i\}, t) = \max_{j\in N_i\cup\{i\}} x_j(t), i = 1, \ldots, n. \tag{2}$$

Apparently, the maximum values can be spread through the network within D steps, the diameter of the network, which can be upper-bounded by n, the number of the nodes. Regarding the computational complexity, for node i we have $|N_i| \propto n$, where $|\cdot|$ denotes the cardinality of a set, thus the computational complexity is $O(n^2)$ for a node.

Similarly, when the directed graph is strongly connected, the min-consensus can be reached [19] if the following control law is chosen:

$$u_i(N_i \cup \{i\}, t) = \min_{j\in N_i\cup\{i\}} x_j(t), i = 1, \ldots, n \tag{3}$$

and the computational complexity for one node is also $O(n^2)$.

Remark 1: In order to perform the max-consensus and min-consensus algorithms, every node needs to know the number of nodes n in the network. This is a strong assumption that lacks robustness (e.g., some nodes could run out of battery) and prevents scaling (i.e., if a new node is added into the network, every other node should update the number of nodes n). This problem can be alleviated by using a semi-distributed algorithm for estimating n (see the work in [30]). On the other hand, if an upper-bound of n is provided, which is a much milder condition, then the max-consensus and min-consensus algorithms can also perform well.

In the average-consensus problem, the nodes are required to converge to the average of their initial states, i.e., $\mathcal{X}(\cdot)$ is the mean of its arguments. When the directed graph is strongly connected and weight balanced, the average-consensus can be reached [19] if the following control law is chosen:

$$u_i(N_i \cup \{i\}, t) = x_i(t) + \tau \sum_{j\in N_i} a_{ij}\big(x_j(t) - x_i(t)\big) \tag{4}$$

where the parameter τ is assumed to be $\tau \leq (1/\max_i(\sum_{j\neq i} a_{ij}))$. Suppose that the times of iteration is $t_{\max}$, then the computational complexity is $O(nt_{\max})$ for a node since $|N_i| \propto n$.

Remark 2: Note that the result of the average-consensus algorithm is only asymptotically correct. This is different from the max-consensus/min-consensus algorithm, which can provide an exact result after a finite number of iterations. In the following, we will introduce the finite-time average-consensus algorithm, which can get the exact average in finite-time steps.

Introduce $W = [w_{ij}] \in \mathbb{R}^{N\times N}$ with

$$w_{ij} = \begin{cases} \tau a_{ij}, & i \neq j \\ 1 - \tau \sum_{k=1}^{N} a_{ik}, & i = j. \end{cases}$$

Denote by $q(t)$ the minimal polynomial of the matrix W, which is a unique nomic polynomial of smallest degree such that $q(W) = 0$. Suppose that the minimal polynomial of W has degree $\sigma + 1$ ($\sigma + 1 \leq N$) and takes the form $q(t) = t^{\sigma+1} + \alpha_\sigma t^\sigma + \cdots + \alpha_1 t + \alpha_0$.

In the finite-time average-consensus problem, the nodes are required to get the exact average of their initial states after finite-time steps. When the directed graph is strongly connected and weight balanced, the finite-time average-consensus can be reached within σ steps [31] by letting

$$x_i(\sigma + 1) = \frac{[x_i(\sigma)\ x_i(\sigma - 1)\ \cdots\ x_i(0)]S}{[1\ 1\ \cdots\ 1]S} \tag{5}$$

with

$$S = \begin{bmatrix} 1 \\ 1 + \alpha_\sigma \\ 1 + \alpha_\sigma + \alpha_{\sigma-1} \\ \vdots \\ 1 + \sum_{j=1}^{\sigma} \alpha_i \end{bmatrix}$$

if the control law (4) is chosen since

$$\lim_{t\to\infty} x_i(t) = \frac{[x_i(\sigma)\ x_i(\sigma - 1)\ \cdots\ x_i(0)]S}{[1\ 1\ \cdots\ 1]S}.$$

The distributed method to get $\alpha_0, \ldots, \alpha_\sigma$ is also provided in [31]. Similarly, the corresponding computational complexity is $O(n^2)$ for a node.

In the following, we will denote by

$$\mathbf{x} = \text{max-consensus}(\mathbf{x}_i, N_i, n)$$

and

$$\mathbf{x} = \text{min-consensus}(\mathbf{x}_i, N_i, n)$$

the execution of n iterations of the max-consensus procedure described by (2) and min-consensus procedure described by (3) by the ith node, respectively, and also denote by

$$\mathbf{x} = \text{average-consensus}(\mathbf{x}_i, N_i, a_i, n)$$

the execution of n iterations of the finite-time average-consensus procedure described by (4) by the ith node, where $\mathbf{x}_i$ is the initial state for node i, N_i is the set of neighbors of node i, a_i is the stack of the weights of node i's ingoing edges, and n is the number of the nodes in the network. The returned value $\mathbf{x}$ of each of the first two algorithms is the final state of node i and the returned value $\mathbf{x}$ of the last algorithm is obtained by (5) for node i, which is the same for every node after running each algorithm.

It is a strong assumption that a directed graph is assumed to be weight balanced. However, some weight balancing techniques [32], [33] can be utilized. In this paper, we employ the mirror imbalance-correcting algorithm presented in [33] to make the network be weight balanced, for which the basic idea is that each node changes its weights of the outgoing edges according to certain strategy. When considering the worst-case scenario, the computational complexity is $O(n^4)$ for each node for each node for each node.

III. Distributed k-Means Algorithm

In this section, the distributed k-means algorithm will be proposed. For this purpose, we first briefly introduce the centralized k-means algorithm which has been very well developed in the existing literature.

A. Introduction to the Centralized k-Means Algorithm

Given a set of observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each observation is a d-dimensional real-valued vector, the k-means algorithm [7] aims to partition the n observations into $k(\leq n)$ sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) function. In other words, its objective is to find

$$\underset{\mathbf{S}}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in S_i} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2 \tag{6}$$

where $\mathbf{c}_j$ is the presentation of cluster j, generally the centroid of points in S_j.

The algorithm uses an iterative refinement technique. Given an initial set of k centroids $\mathbf{c}_1(1), \ldots, \mathbf{c}_k(1)$, the algorithm proceeds by alternating between an assignment step and an update step as follows.

During the assignment step, assign each observation to the cluster characterized by the nearest centroid, that is

$$S_i(T) = \left\{ \mathbf{x}_p : \left\| \mathbf{x}_p - \mathbf{c}_i(T) \right\|^2 \leq \left\| \mathbf{x}_p - \mathbf{c}_j(T) \right\|^2, 1 \leq j \leq k \right\}, \quad i = 1, \ldots, k \tag{7}$$

where each $\mathbf{x}_p$ is assigned to exactly one cluster, even if it could be assigned to two or more of them. Apparently, this step can minimize the WCSS function.

During the update step, the centroid, say $\mathbf{c}_i(T+1)$, of the observations in the new cluster is computed as follows:

$$\mathbf{c}_i(T+1) = \frac{1}{|S_i(T)|} \sum_{\mathbf{x}_j \in S_i(T)} \mathbf{x}_j, \quad i = 1, \ldots, k. \tag{8}$$

Since the arithmetic mean is a least-squares estimator, this step also minimizes the WCSS function.

The algorithm converges if the centroids no longer change.

The k-means algorithm can converge to a (local) optimum, while there is no guarantee for it to converge to the global optimum [7]. Since for a given k, the result of the above clustering algorithm depends solely on the initial centroids, a common practice in clustering the sensor observations is to execute the algorithm several times for different initial centroids and then select the best solution.

The computational complexity of the k-means algorithm is $O(nkdM)$ [7], where n is the number of the d-dimensional vectors, k is the number of clusters, and M is the number of iterations before reaching convergence.

B. Choosing Initial Centroids Using the Distributed k-Means++ Algorithm

The choice of the initial centroids is the key to making the k-means algorithm work well. In the k-means algorithm, the partition result is dependent only on the initial centroids for a given k, so does the distributed one. Thus, an effective distributed initial centroids choosing method is important. Here we provide a distributed implementation of the k-means++ algorithm [24], a centralized algorithm to find the initial centroids for the k-means algorithm. It is noted that k-means++ generally outperforms k-means in terms of both accuracy and speed [24].

Algorithm 1: Distributed k-Means++ Algorithm: ith Node

Data: $\mathbf{x}_i, n, k, N_i$
Result: $\mathbf{c}(1)$

* Propagation of the first centroid *\
$temp_i = \text{random}(0, 1)$;
$temp = \text{max-consensus}(temp_i, N_i, n)$;
if $temp_i == temp$ **then**
  $\mathbf{x}_{ic} = \mathbf{x}_i$;
else
  $\mathbf{x}_{ic} = [-\infty, \ldots, -\infty]'$;
end
$\mathbf{c}_1(1) = \text{max-consensus}(\mathbf{x}_{ic}, N_i, n)$;
for $m = 2; m \leq k; m++$ **do**
  * Propagation of the maximum value of all the nodes' distance to their nearest centroids *\
  $d_i = \min_{1 \leq j \leq m-1} \|\mathbf{x}_i - \mathbf{c}_j(1)\|$;
  $rand_i = d_i^2 \times \text{random}(0, 1)$;
  $rand_{\max} = \text{max-consensus}(rand_i, N_i, n)$;
  * Propagation of the tth center *\
  if $rand_i == rand_{\max}$ **then**
    $\mathbf{x}_{ic} = \mathbf{x}_i$;
  else
    $\mathbf{x}_{ic} = [-\infty, \ldots, -\infty]'$;
  end
  $\mathbf{c}_m(1) = \text{max-consensus}(\mathbf{x}_{ic}, N_i, n)$;
end
$\mathbf{c}(1) = [\mathbf{c}_1(1)', \ldots, \mathbf{c}_k(1)']'$;

Let $D(\mathbf{x})$ denote the distance from observation $\mathbf{x}$ to the closest centroids that have been chosen. The k-means++ algorithm [24] is executed as follows.

1) Choose randomly an observation from $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ as the first centroid $\mathbf{c}_1$.
2) Take a new center $\mathbf{c}_j$, choosing $\mathbf{x}$ from the observations with probability $D(\mathbf{x})^2 / (\sum_{\mathbf{x} \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}} D(\mathbf{x})^2)$.
3) Repeat step 2) until we have taken k centers altogether.

Consider that the n nodes are deployed in an Euclidean space of any dimension and suppose that each node has a limited sensing radius which may be different for different nodes, therefore leading to the fact that the underlying topology of the WSN is directed. Each node is endowed with a real vector $\mathbf{x}_i \in \mathbb{R}^d$ representing the observation. Here we assume that every node has its own unique identification (ID), and further the underlying topology of the WSN is strongly connected. The detailed realization of the distributed k-means++ algorithm is as shown in Algorithm 1.

For node i, it knows its observation $\mathbf{x}_i$, n (the number of the nodes in the network), k (the number of the clusters), as well as N_i (the set of node i's neighbors). Executing the distributed k-means++ algorithm on all the nodes synchronously yields initial centroids $\mathbf{c}(1)$ which can be propagated to all the nodes. The detailed procedures are listed as follows.

1) First, every node produces a random number between 0 and 1, and the maximum number can be figured out by all nodes through the max-consensus algorithm. The node with maximum number chooses its observation while others choose $[-\infty, \ldots, -\infty]' \in \mathbb{R}^d$ to participate in the max-consensus algorithm so that the observation of the node with the largest random number, designated as the first centroid $\mathbf{c}_1(1)$, can be known by every node.
2) Next, all the nodes perform the following steps iteratively until all the k centroids are fixed. In the mth iteration, calculate the distance from each node to the existing centroids and obtain $d_i = \min_{1\le j\le m-1} \|\mathbf{x}_i - \mathbf{c}_j(1)\|$, $i = 1, \ldots, n$. Then, every node produces a random number $rand_i$ between 0 and d_i^2, followed by using the max-consensus algorithm to spread $rand_{\max} = \max_{1\le i\le n} rand_i$ throughout the network. If a certain node, say i, satisfies $rand_i = rand_{\max}$, then its observation is designated as the mth centroid and further is obtained by other nodes in the same manner as that for the first centroid.

Since the computational complexity of the max-consensus algorithm is $O(dn^2)$, and the max-consensus algorithm needs to be executed for $2k$ times, the distributed k-means++ algorithm's computational complexity is $O(kdn^2)$ for each node.

Remark 3: A contradiction will appear if there exist at least two nodes, say node i and node j, such that $\text{rand}_i = \text{rand}_j = \text{rand}_{\max}$ in Algorithm 1, since these two nodes both choose their own observations as the new centroid. A solution to this contradiction is that for any node, say node ℓ, with $\text{rand}_\ell = \text{rand}_{\max}$, it provides its ID while others provide 0 to participate in the max-consensus algorithm. Therefore, the maximum ID of the node with $\text{rand}_\ell = \text{rand}_{\max}$ can be obtained by every node, and further such a node's observation can be chosen as the new centroid. In fact, the same operation needs to be performed for obtaining the first centroid, although the probability for two nodes to produce the same random number is normally very small.

C. Distributed k-Means Algorithm

Based on the above work, in this section we will introduce the proposed distributed k-means algorithm in detail.

The objective of the distributed k-means algorithm is to partition the data observed by the nodes into k clusters, which minimizes the WCSS function (6) via a distributed approach. Similarly to the work in [18], we follow the same step as that for the centralized algorithm except that every step is performed in a decentralized manner. During the assignment step, if all the centroids are spread through the network via the max-consensus algorithm, then each node knows which cluster it belongs to. During the update step, the finite-time average-consensus algorithm can be utilized to choose the mean of each cluster as the new centroid. Note that some pretreatments such as weight balancing and normalization [see Fig. 1(a)] need to be done before the distributed algorithm is being executed. The flowchart for the distributed k-means algorithm is shown in Fig. 1(a) and the concrete realization of the algorithm is shown in Algorithm 2.

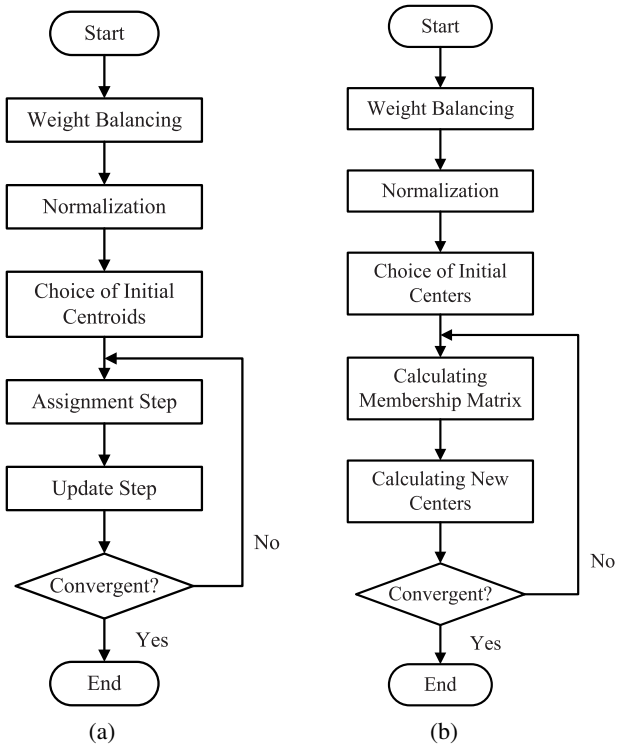


Fig. 1. Flowchart for proposed algorithms. Flowchart for distributed (a) k-means algorithm and (b) c-means algorithm.

Note that in Algorithm 2,

mirror-imbalance-correcting$(\cdot)$

is the mirror imbalance-correcting algorithm in [33] and

distributed-k-means++$(\cdot)$

is the distributed k-means++ algorithm proposed in Section III-B.

For node i, it knows its observation $\mathbf{x}_i \in \mathbb{R}^d$, n (the number of the nodes), k (the number of the clusters), as well as N_i (the set of node i's neighbors). All the nodes execute the distributed k-means algorithm synchronously, and the final centroids $\mathbf{c}$ and the cluster k_i which it belongs to, are obtained by every node. The whole algorithm is executed as follows.

1) Weight Balancing: To compute the average of a number of observations by the finite-time average-consensus algorithm, the strongly connected directed graph needs to be weight balanced. The mirror imbalance-correcting algorithm [33] is executed to assign weights to node i's outgoing edges to make the network be weight balanced.

2) Normalization: A min–max value standardization method is used to normalize the components in the observations. The max-consensus and min-consensus algorithms [19] are used first to obtain the maximum and minimum values of

Algorithm 2: Distributed k-Means Algorithm: ith Node

Data: $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]', n, k, N_i$
Result: $\mathbf{c}, k_i$

* Weight balancing *\
$a_i =$ mirror-imbalance-correcting(N_i);
* Normalization *\
$[\max_i, \ldots, \max_d]' =$ max-consensus($\mathbf{x}_i, N_i, n$);
$[\min_i, \ldots, \min_d]' =$ min-consensus($\mathbf{x}_i, N_i, n$);
$\hat{x}_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j}$, for $j = 1, \ldots, d$;
$\hat{\mathbf{x}}_i = [\hat{x}_{i1}, \ldots, \hat{x}_{id}]'$;
* Choice of the initial centroids *\
$\mathbf{c}(1) \triangleq [\mathbf{c}_1(1)', \ldots, \mathbf{c}_k(1)']' =$
distributed-k-means++($\mathbf{x}_i, n, k, N_i$);
T = 1;
while *true* **do**
  * Assignment step *\
  $k_i = \text{argmin}_j \|\hat{\mathbf{x}}_i - \mathbf{c}_j(T)\|$;
  * Update step *\
  $T = T + 1$;
  for $j = 1; j \le k; j++$ **do**
    if $k_i == j$ **then**
      $n_{ic} = 1$;
    else
      $n_{ic} = 0$;
    end
    $\mathbf{c}_j(T) =$ average-consensus($n_{ic}\hat{\mathbf{x}}_{ic}, N_i, a_i, n$);
    $n_j =$ average-consensus(n_{ic}, N_i, a_i, n);
    $\mathbf{c}_j(T) = \mathbf{c}_j(T)/n_j$;
  end
  $\mathbf{c}(T) = [\mathbf{c}_1(T)', \ldots, \mathbf{c}_k(T)']'$;
  * Check convergence *\
  if $\mathbf{c}(T) == \mathbf{c}(T-1)$ **then**
    break;
  end
end
$\mathbf{c} = \mathbf{c}(T)$;

every component of the observation. For the jth component of node i's observation x_{ij}, if the maximum and minimum value are $\max_j$ and $\min_j$, respectively, then invoking the min–max value standardization x_{ij} is normalized as follows:

$$\hat{x}_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j}. \tag{9}$$

3) Choice of the Initial Centroids: Similar to the centralized k-means++, this step is the key to yield a faster convergence speed as well as a higher possibility of obtaining the global optimum. The normalized observations are applied to the distributed k-means++ algorithm to produce the initial centroids. Let $T = 1$, and the initial centroids are $\mathbf{c}(\mathbf{1})$.

4) Assignment Step: Since the centroids have been provided in step 3), each node needs to figure out which cluster, represented by the centroid, it belongs to. More specific, node i belongs to the cluster k_i which satisfies the condition that the distance from node i to the centroid of cluster k_i is the shortest among all the distances from the centroids to such a node, i.e., it belongs to the k_ith cluster with $k_i = \text{argmin}_j \|\hat{\mathbf{x}}_i - \mathbf{c}_j(T)\|$.

5) Update Step: After the assignment step, the update step is executed. Let $T = T + 1$. To compute the jth center, each node provides its normalized observations if it is assigned to the jth cluster and $[0, \ldots, 0] \in \mathbb{R}^d$ otherwise, to participate in the finite-time average-consensus algorithm. If the quantity of the nodes in the jth cluster is q_j and the centroid is $\mathbf{c}_j(T)$, then it is apparent that $q_j\mathbf{c}_j(T) = \sum_{i=1}^n n_{ic}\hat{\mathbf{x}}_{ic}$. Thus, the result of the finite-time average-consensus algorithm is $(q_j\mathbf{c}_j(T)/n)$.

Then every node chooses 1 if it is assigned to the jth cluster and 0 otherwise to participate in the finite-time average-consensus algorithm. The finite-time average-consensus algorithm's result is (q_j/n) since $q_j = \sum_{i=1}^n n_{ic}$. Apparently, the jth cluster's centroid can be obtained by $\mathbf{c}_j(T) = (q_j\mathbf{c}_j(T)/n)/(q_j/n)$.

6) Check Convergence: After the new centroids are produced, we need to judge whether the algorithm has converged. Obviously, if $\mathbf{c}(T) = \mathbf{c}(T-1)$, then $\mathbf{c}(T)$ is the final cluster center $\mathbf{c}$. Otherwise, the assignment step and update step are repeated until the algorithm converges.

It is known that the computational complexity of the distributed k-means++ algorithm is $O(kdn^2)$ and the complexity of the mirror imbalance-correcting algorithm is $O(n^4)$ for a node. During the normalization step, every node executes the max-consensus and min-consensus for one time and the dimension of the observations is d, so the computational complexity is $O(dn^2)$. As for the assignment step and update step, if the algorithm has converged within T steps, then the computational complexity is $O(kdn^2T)$, since the complexity of the finite-time average-consensus algorithm is $O(dn)$ for a node. In conclusion, the computational complexity of the distributed k-means algorithm is $O(n^2 \max\{n^2, dkT\})$ for a node, and when the network is undirected, the computational complexity is $O(n^2dkT)$ for a node since an undirected graph is always weight balanced.

D. Advantages of the Distributed k-Means Algorithm

Compared with the algorithms proposed in [18], the main difference occurs in the update step regarding how to calculate the averages of the observations in each cluster. For the algorithm proposed in this paper, when calculating the average of the nodes' observations in each cluster, each node provides valid information if it belongs to a cluster otherwise invalid information (i.e., 0) through the whole network. Thus, the proposed algorithm is valid as long as the whole network is strongly connected. However, for the algorithms given in [18], the centroid for each cluster of observations is calculated by employing the finite-time average-consensus algorithm on the underlying network of such a cluster, thus leading to two shortcomings of the algorithm proposed in [18].

The first shortcoming is that it cannot be extended directly to the case when the network topology is a directed graph, because there is no guarantee that the subnetwork (i.e., the underlying topology of each cluster) is strongly connected as well if the whole network itself is strongly connected. The second shortcoming is that for the algorithm provided in [18] to cope with disconnected clusters, it involves a high computational complexity of $O(n^2 k \max\{d, n\}|\sigma(S_j(t))|T)$, where $\sigma(S_j(t))$ means the set of subclusters of cluster $S_j(t)$ at step t, while the computational complexity of the algorithm proposed herein is only $O(n^2 dkT)$. In other words, the advantages of the algorithm proposed in this paper over that in [18] is its capability of dealing with the condition that the network topology is a directed graph and its lower computational complexity.

Remark 4: The distributed k-means algorithm prefers clusters of approximately similar size, as it will always assign an observation to the nearest centroid. This often leads to incorrect cut borders in between clusters. Moreover, it is necessary for an observation to be assigned to more than one cluster [20], [21]. For example, when one data point is located in the overlapping area of many different clusters, it may not be appropriate to assign the data point to exactly one cluster since it has the properties of those overlapping clusters. Hence, a distributed fuzzy partition method, which will be introduced in the next section, is needed.

IV. Distributed Fuzzy c-Means Algorithm

In this section, we introduce the distributed fuzzy c-means algorithm. To this end, the fuzzy c-means algorithm is first briefly presented here.

A. Introduction to the Centralized Fuzzy c-Means Algorithm

Consider a given set of observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each observation is a d-dimensional real vector. Similarly to the k-means algorithm, the fuzzy c-means algorithm [34], [35] aims to partition the n observations into $k(\leq n)$ sets $S = \{S_1, S_2, \ldots, S_k\}$ with degrees called the membership matrix described by a row stochastic matrix $U = [u_{ij}] \in \mathbb{R}^{n\times k}$, where the membership value u_{ij} means that node i belongs to cluster j with the degree of u_{ij}, so as to minimize an objective function. The objective function is given by

$$\sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}^m \left\|\mathbf{x}_i - \mathbf{c}_j\right\|^2 \tag{10}$$

where $\mathbf{c}_j$ is the presentation of the cluster j, which is called the center later, and m is the fuzzifier determining the level of cluster fuzziness. In the absence of experimentation or domain knowledge, m is commonly set to 2 [35].

The Lagrangian multiplier method [34] can be used to solve this problem and the necessary condition to minimize the objective function is

$$\mathbf{c}_j = \frac{\sum_{i=1}^{n} u_{ij}^m x_i}{\sum_{i=1}^{n} u_{ij}^m}, j = 1, \ldots, k \tag{11}$$

and

$$u_{ij} = \frac{1}{\sum_{t=1}^{k}\left(\frac{\mathbf{x}_i - \mathbf{c}_j}{\mathbf{x}_i - \mathbf{c}_t}\right)^{2/(m-1)}}, i = 1, \ldots, n, j = 1, \ldots, k. \tag{12}$$

Similar to the k-means algorithm, the fuzzy c-means algorithm also uses an iterative refinement technique. Given an initial set of k centers $\mathbf{c}_1(1), \ldots, \mathbf{c}_k(1)$ or an initial membership matrix, the algorithm proceeds by alternating between updating $\mathbf{c}_j$ and U according to (11) and (12).

Remark 5: The algorithm minimizes intracluster variance as well, but has the same problems as the k-means algorithm, that is, the obtained minimum is a local minimum and the results depend on the initial choice.

B. Distributed Fuzzy c-Means Algorithm

In this section, a proposed distributed fuzzy c-means algorithm will be presented in detail.

We make the same assumptions as the distributed k-means algorithm, i.e., the n nodes deployed in any high dimensional space are endowed with real vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, n$, which represent the observations, and the underlying topology of the WSN is strongly connected.

The goal of the proposed algorithm is to obtain centers representing different clusters and the membership matrix describing to what extent a node belongs to a cluster so as to minimize the objective function described in (10). We follow the same step as that for the centralized algorithm except that every step is performed in a decentralized manner. If all the centers can be spread across the network via the max-consensus algorithm, then each node can calculate the membership matrix directly according to (12). When computing the new centers, the finite-time average-consensus algorithm can be utilized. The flowchart for the distributed fuzzy c-means algorithm is shown in Fig. 1(b) and the concrete realization of the algorithm is given in Algorithm 3.

For node i, it knows its observation $\mathbf{x}_i \in \mathbb{R}^d$, n (the number of the nodes in the network), k (the number of the clusters), N_i (the set of neighbors of node i), as well as δ (the parameter to judge whether the algorithm converges). All the nodes execute the distributed fuzzy c-means algorithm synchronously and the final centers $\mathbf{c}$ and the membership value u_{ij} describing to what extent node i belongs to cluster j, are obtained by every node. The whole algorithm is executed as follows.

1) Weight Balancing, Normalization and Choice of the Initial Centers: These steps are the same as the distributed k-means algorithm.

2) Calculate Membership Matrix: Since the centers have been provided in step 1), each node can calculate the membership matrix as shown in (12).

3) Calculate New Centers: Each node provides $u_{ij}^m \hat{x}$ and u_{ij}^m to participate in the finite-time average-consensus algorithm. So the resulting finite-time average-consensus algorithm should be $(\sum_{i=1}^{n} u_{ij}^m \hat{x}_i/n)$ and $(\sum_{i=1}^{n} u_{ij}^m/n)$. Apparently, the ratio of the two results is the new center according to (11).

Algorithm 3: Distributed Fuzzy c-Means Algorithm: ith Node

Data: $\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]', n, k, N_i, \delta$
Result: $\mathbf{c}, u_{ij}(j = 1, \ldots, k)$

* Weight balancing *\
a_i =mirror-imbalance-correcting(N_i);
* Normalization *\
$[\max_i, \ldots, \max_d]' =$ max-consensus($\mathbf{x}_i, N_i, n$);
$[\min_i, \ldots, \min_d]' =$ min-consensus($\mathbf{x}_i, N_i, n$);
$\hat{x}_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j}$,for $j = 1, \ldots, d$;
$\hat{\mathbf{x}}_i = [\hat{x}_{i1}, \ldots, \hat{x}_{id}]'$;
* Choice of the initial centers *\
$\mathbf{c}(1) \triangleq [\mathbf{c}_1(1)', \ldots, \mathbf{c}_k(1)']' =$
distributed-k-means++($\mathbf{x}_i, n, k, N_i$);
T = 1;
while *true* **do**
  * Calculate membership matrix *\
  for $j = 1; j \leq k; j++$ **do**
    $u_{ij} = \frac{1}{\sum_{t=1}^{k} \left(\frac{\mathbf{x}_i - \mathbf{c}_j}{\mathbf{x}_i - \mathbf{c}_t}\right)^{2/(m-1)}}$;
  end
  * Calculate new centers *\
  $T = T + 1$;
  for $j = 1; j \leq k; j++$ **do**
    $\mathbf{c}_j(T) =$average-consensus($u_{ij}^m \hat{\mathbf{x}}_i, N_i, a_i, n$);
    $n_j =$ average-consensus(u_{ij}^m, N_i, a_i, n);
    $\mathbf{c}_j(T) = \mathbf{c}_j(T)/n_j$;
  end
  $\mathbf{c}(T) = [\mathbf{c}_1(T)', \ldots, \mathbf{c}_k(T)']'$;
  * Check convergence *\
  if $\|\mathbf{c}(T) - \mathbf{c}(T-1)\| < \delta$ **then**
    break;
  end
end
$\mathbf{c} = \mathbf{c}(T)$;

4) Check Convergence: After the new centers are produced, we need to judge whether the algorithm converges. Obviously, if $\|\mathbf{c}(T) - \mathbf{c}(T-1)\| < \delta$, which means that the centers change a little during two iterations, then $\mathbf{c}(T)$ is the final cluster centers $\mathbf{c}$. Otherwise, $T = T + 1$, we need to calculate the membership matrix and the new centers iteratively until the algorithm converges.

Similar to the distributed k-means algorithm, the computational complexity for a node of the distributed fuzzy c-means algorithm is $O(n^2 \max\{n^2, dkT\})$ if the network is directed and $O(n^2 dkT)$ if the network is undirected.

Remark 6: Similar to the centralized fuzzy c-means algorithm, the distributed fuzzy algorithm can begin with a given initial set of centers or a given initial membership matrix as well. Note that the distributed k-means++ algorithm provided in Section III-B is also applicable in finding an initial set of centers for the distributed fuzzy c-means algorithm.

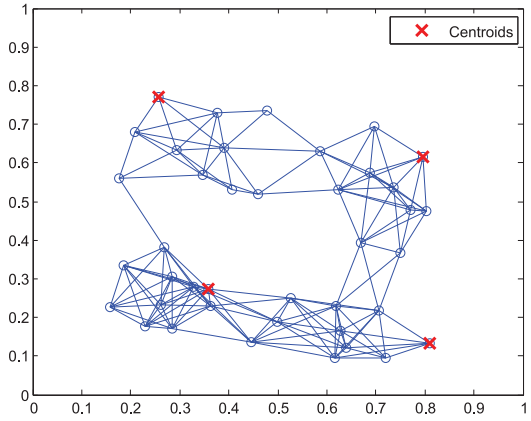


Fig. 2. Observations of the positions with initial centroids and undirected network topology.

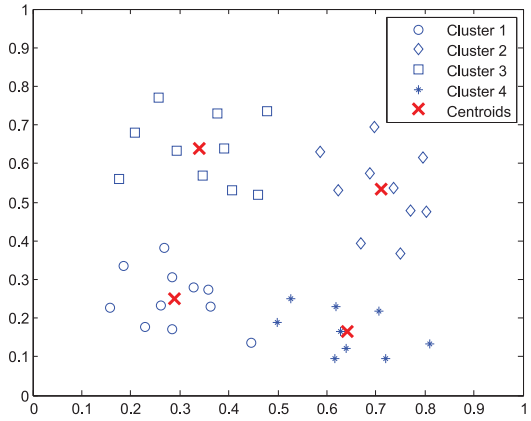


Fig. 3. Clustering result with final centroids of the distributed k-means algorithm in the undirected graph.

V. Simulation Results

Now we provide several examples to illustrate the capabilities of the proposed distributed k-means algorithm and distributed fuzzy c-means algorithm.

Example 1: In this example, we consider that the network topology is an undirected graph. Here the 40 nodes are randomly selected in the region $[0, 1] \times [0, 1]$ and the observation of each node is the position. We need to partition the 40 measures into four clusters. The network is shown in Fig. 2 and the initial centroids chosen by the distributed k-means algorithm are marked by red cross.

First, we use the proposed distributed k-means algorithm to deal with these observations and the clustering result is depicted in Fig. 3, in which the different types of points mean different clusters while the final centroids are marked by red cross.

And then the proposed distributed fuzzy c-means algorithm is used to deal with the observations in Fig. 4. Considering that soft clustering cannot get a fixed partition, we make the following treatment so that the clustering result can be observed directly. For node i, if $u_{ij} \geq 0.5$, then we partition node i into cluster j. If $u_{ij} < 0.5$ for all j, then we call it fuzzy points presented by "plus" in the simulation results. The simulation result is depicted in Fig. 4. Note that the fuzzy points

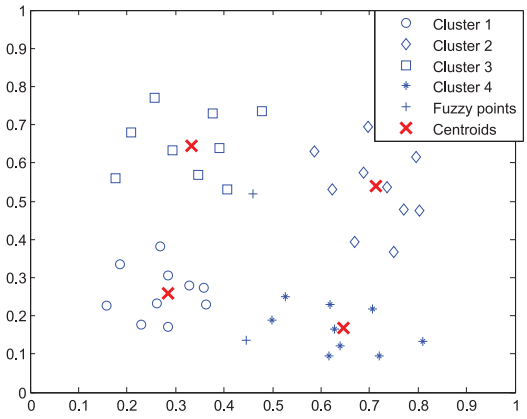


Fig. 4. Clustering result with final centroids of the distributed fuzzy *c*-means algorithm in the undirected graph.

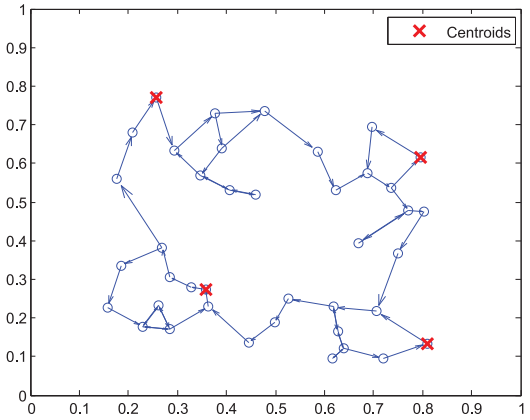


Fig. 5. Observations of the positions with initial centroids and directed network topology.

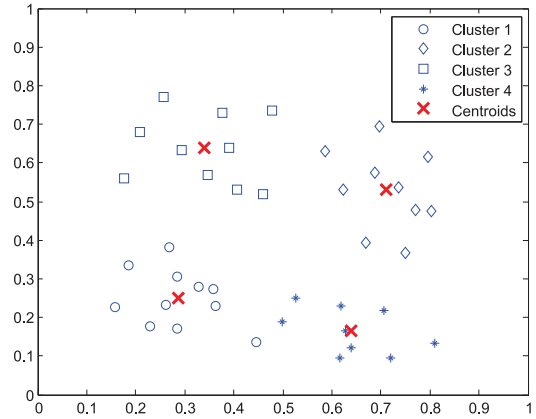


Fig. 6. Clustering result with final centroids of the distributed *k*-means algorithm in the directed graph.

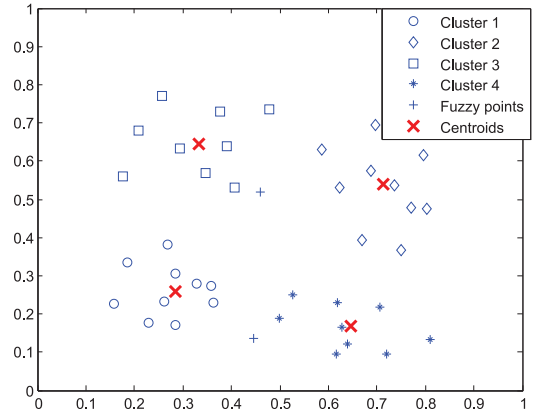


Fig. 7. Clustering result with final centroids of the distributed fuzzy *c*-means algorithm in the directed graph.

are always on the borders of clusters which is tally with the actual situation.

Here we provide the same observations except that the network is a connected directed graph as shown in Fig. 5, where the initial centers are marked by red cross. The clustering result using the distributed *k*-means algorithm is depicted in Fig. 6, in which the different types of points mean different clusters and the final centers are marked by red cross. When the distributed fuzzy *c*-means algorithm is applied, the corresponding results are shown in Fig. 7. From Figs. 3 and 6 which are plotted using the distributed *k*-means algorithm, as well as Figs. 4 and 7 which are plotted using the distributed fuzzy *c*-means algorithm, it can be observed that the clustering is the same for the same observations regardless of whether the network topology is undirected or not.

Example 2: Consider a WSN with 3600 nodes. The network is a connected undirected graph with each node communicating with other 50 nodes. The observed data is generated from nine classes, with vectors from each class generated from a 2-D Gaussian distribution with the common covariance $\Sigma = 2I_2$, and the corresponding means: $\mu_1 = [6, 6]^T$, $\mu_2 = [0, 6]^T$, $\mu_3 = [-6, 6]^T$, $\mu_4 = [6, 0]^T$, $\mu_5 = [0, 0]^T$, $\mu_6 = [-6, 0]^T$, $\mu_7 = [6, -6]^T$, $\mu_8 = [0, -6]^T$, and $\mu_9 = [-6, -6]^T$. Every node randomly draws one data point from the nine classes and 400 data points are chosen from a class.

Fig. 8(b) depicts the evolution of the WCSS functions of the distributed *k*-means algorithm with $k = 6, 9, 12$ and of the centralized *k*-means algorithm with $k = 9$. This illustrates that the convergence of the proposed distributed implementation is similar to the centralized algorithm. The clustering result of the distributed *k*-means algorithm with $k = 9$ is displayed in Fig. 8(a). Table I compares the performance including the final value of the WCSS function and the iteration times before convergence over 100 Monte Carlo runs for the distributed *k*-means algorithm (represented by d*k*-means), the *k*-means algorithm with first centroids chosen by using the *k*-means++ algorithm (represented by *k*-means++), and the *k*-means algorithm with first centroids randomly chosen from the observations (represented by the *k*-means). It can be observed that the distributed *k*-means algorithm performs the best both in terms of speed and accuracy.

Fig. 9(b) depicts the evolution of the objective functions of the distributed fuzzy *c*-means algorithm with $k = 6, 9, 12$ and of the centralized fuzzy *c*-means algorithm with $k = 9$. This illustrates that the convergence of the proposed distributed implementation is similar to the centralized algorithm. The clustering result of the distributed fuzzy *c*-means algorithm

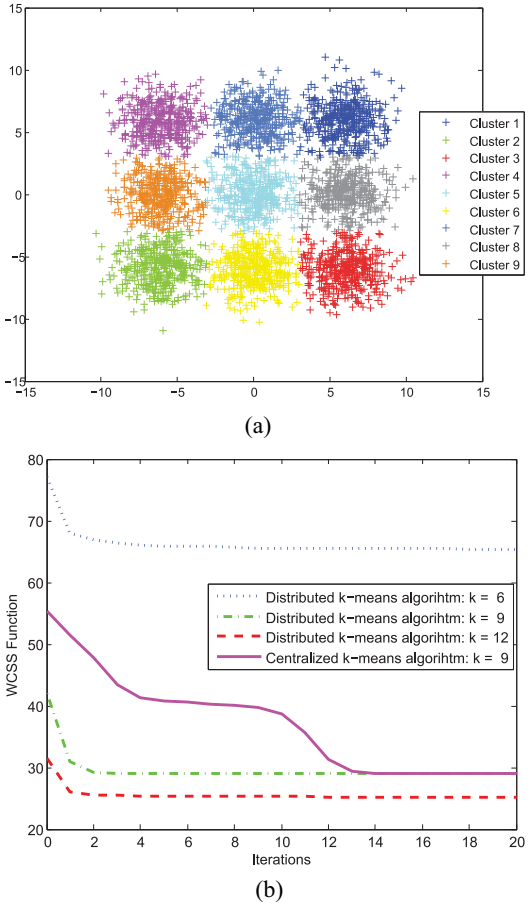


Fig. 8. Clustering of the distributed k-means algorithm in the undirected graph with 3600 observations fitting Gaussian distribution. (a) Clustering result of the distributed k-means algorithm with $k = 9$. (b) Evolution of the WCSS functions of the distributed k-means algorithm with $k = 6, 9, 12$ and of the centralized k-means algorithm with $k = 9$.

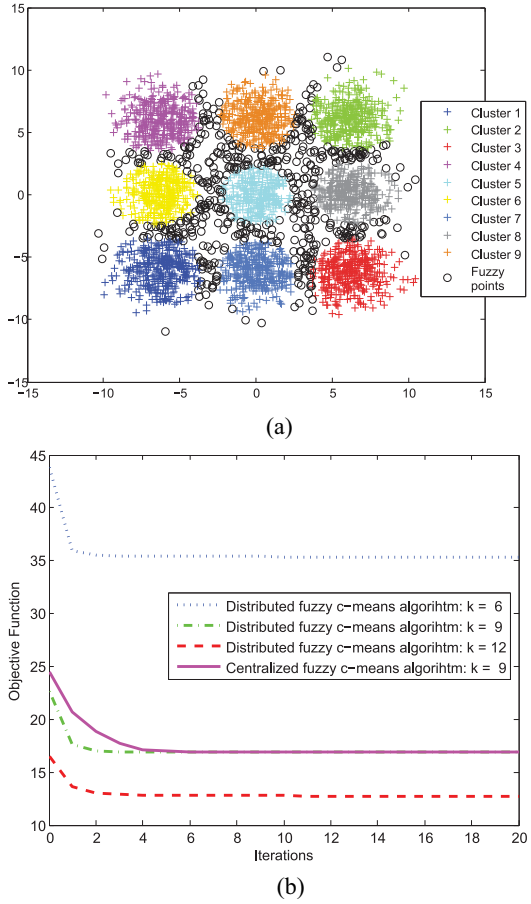


Fig. 9. Clustering of the distributed fuzzy c-means algorithm in the undirected graph with 3600 observations fitting Gaussian distribution. (a) Clustering result of the distributed fuzzy c-means algorithm with $k = 9$. (b) Evolution of the objective functions of the distributed fuzzy c-means algorithm with $k = 6, 9, 12$ and of the centralized fuzzy c-means algorithm with $k = 9$.

TABLE I
CENTRALIZED VERSUS DISTRIBUTED k-MEANS. PERFORMANCE COMPARISON

k	Average WCSS function			Minimum WCSS function		
	dk-means	k-means++	k-means	dk-means	k-means++	k-means
6	67.1636	67.3448	67.6695	65.0229	65.0200	65.0238
9	29.5596	31.7265	34.7794	28.9506	28.9506	28.9506
12	25.1681	25.1970	25.2798	24.9548	24.9674	24.9927
	Average iteration times			Minimum iteration times		
k	dk-means	k-means++	k-means	dk-means	k-means++	k-means
6	23.63	24.50	28.65	9	9	11
9	9.06	16.02	18.87	4	6	6
12	23.99	26.73	27.66	7	8	8

TABLE II
CENTRALIZED VERSUS DISTRIBUTED FUZZY c-MEANS. PERFORMANCE COMPARISON

k	Average objective function			Minimum objective function		
	dc-means	fuzzy c-means++	fuzzy c-means	dc-means	fuzzy c-means++	fuzzy c-means
6	35.5444	35.4551	35.4292	35.0120	35.0120	35.0120
9	16.8984	16.8984	16.9893	16.8984	16.8984	16.8984
12	12.7520	12.7741	12.7642	12.6507	12.6507	12.6507
	Average iteration times			Minimum iteration times		
k	dc-means	fuzzy c-means++	fuzzy c-means	dc-means	fuzzy c-means++	fuzzy c-means
6	56.43	60.35	60.5	13	14	16
9	14.94	26.51	25.19	9	10	14
12	83.83	87.15	87.7	24	25	28

with $k = 9$ is displayed in Fig. 9(a), in which the fuzzy points are presented by black "circle." Table II compares the performance including the finial value of the objective function and the iteration times before convergence over 100 Monte Carlo runs for the distributed fuzzy c-means algorithm (represented by dc-means), the fuzzy c-means algorithm with first centroids chosen by using the k-means++ algorithm (represented by fuzzy c-means++), and the fuzzy c-means algorithm with first centroids randomly chosen from the observations (represented by the fuzzy c-means). It can be seen that the distributed fuzzy c-means algorithm performs the best both in terms of speed and accuracy.

Example 3: This example tests the proposed decentralized schemes on real data for the first cloud cover database available from the UC-Irvine Machine Learning Repository, with the goal of identifying regions sharing common characteristics. The cloud dataset is composed of 1024 points in 10-D. Suppose that the WSN consisting of 1024 nodes is a connected undirected graph with each node communicating with other 20 nodes.

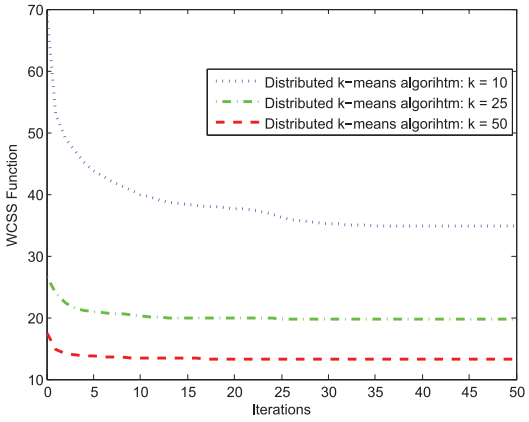


Fig. 10. Evolution of the WCSS functions of the distributed *k*-means algorithm with $k = 10, 25, 50$ in the undirected graph with 1024 10-D observations from cloud dataset.

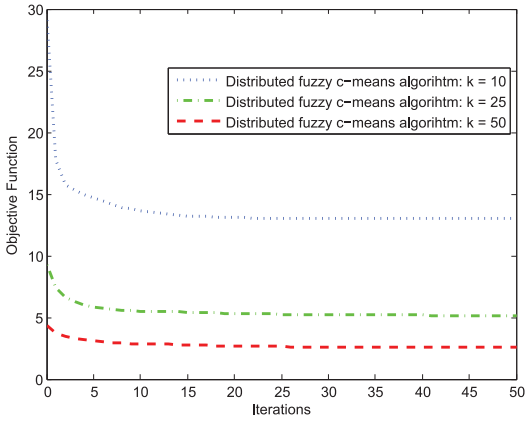


Fig. 11. Evolution of the objective functions of the distributed fuzzy *c*-means algorithm with $k = 10, 25, 50$ in the undirected graph with 1024 10-D observations from cloud dataset.

Fig. 10 depicts the evolution of the WCSS functions of the distributed *k*-means algorithm with $k = 10, 25, 50$, which confirms the convergence of the proposed *k*-means algorithm. Moreover, Fig. 11 plots the evolution of the objective functions of the distributed fuzzy *c*-means algorithm with $k = 10, 25, 50$, which demonstrates the convergence of the proposed *c*-means algorithm.

VI. Conclusion

In this paper, based on the distributed consensus theory in multiagent systems, we have developed a distributed *k*-means clustering algorithm as well as a distributed fuzzy *c*-means algorithm for clustering data in WSNs with strongly connected underlying graph topology. The proposed distributed *k*-means algorithm, by virtue of one-hop communication, has been proven to be feasible in partitioning the data observed by the nodes into measure-dependent groups which has small in-group and large out-group differences. On the other hand, the proposed distributed fuzzy *c*-means algorithm has been shown to be workable in partitioning the data observed by the nodes into different measure-dependent groups with degrees of membership values. Finally, some simulation results have been provided to illustrate the feasibility of the proposed distributed algorithms.

Acknowledgment

The authors would like to thank the Associate Editor and the anonymous reviewers for their valuable comments and suggestions that have helped to improve this paper considerably.

References

[1] K. Sohraby, D. Minoli, and T. F. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*. Hoboken, NJ, USA: Wiley, 2007.

[2] J. Heo, J. Hong, and Y. Cho, "EARQ: Energy aware routing for real-time and reliable communication in wireless industrial sensor networks," *IEEE Trans. Ind. Informat.*, vol. 5, no. 1, pp. 3–11, Feb. 2009.

[3] Q.-S. Jia, L. Shi, Y. Mo, and B. Sinopoli, "On optimal partial broadcasting of wireless sensor networks for Kalman filtering," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 715–721, Mar. 2012.

[4] J. Liang, Z. Wang, B. Shen, and X. Liu, "Distributed state estimation in sensor networks with randomly occurring nonlinearities subject to time delays," *ACM Trans. Sensor Netw.*, vol. 9, no. 1, 2012, Art. ID 4.

[5] W. Kim, J. Park, J. Yoo, H. J. Kim, and C. G. Park, "Target localization using ensemble support vector regression in wireless sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 4, pp. 1189–1198, Aug. 2013.

[6] S. H. Semnani and O. A. Basir, "Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance systems," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 129–137, Jan. 2015.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probab.*, vol. 1. Berkeley, CA, USA, 1967, pp. 281–297.

[8] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, no. 2–3, pp. 191–203, 1984.

[9] G. H. Ball and D. J. Hall, *ISODATA, A Novel Method of Data Analysis and Pattern Classification*. Menlo Park, CA, USA: Stanford Res. Inst., 1965.

[10] J. C. 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.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Disc. Data Mining*, Portland, OR, USA, Aug. 1996, pp. 226–231.

[12] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," *Large-Scale Parallel Data Mining* (LNCS 1759). Berlin, Germany: Springer, 2000, pp. 245–260.

[13] S. Kantabutra and A. L. Couch, "Parallel K-means clustering algorithm on NOWs," *NECTEC Tech. J.*, vol. 1, no. 6, pp. 243–247, 2000.

[14] S. Bandyopadhyay *et al.*, "Clustering distributed data streams in peer-to-peer environments," *Inf. Sci.*, vol. 176, no. 14, pp. 1952–1985, 2006.

[15] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based k-means algorithm for distributed learning using wireless sensor networks," in *Proc. Workshop Sensors Signal Inf. Process.*, Sedona, AZ, USA, May 2008, pp. 11–14.

[16] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 707–724, Aug. 2011.

[17] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.

[18] G. Oliva, R. Setola, and C. N. Hadjicostis. (Nov. 10, 2014). *Distributed K-Means Algorithm*. [Online]. Available: http://arxiv.org/abs/1312.4176

[19] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

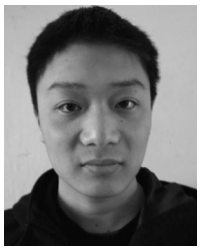
[20] A. B. McBratney and I. O. A. Odeh, "Application of fuzzy sets in soil science: Fuzzy logic, fuzzy measurements and fuzzy decisions," *Geoderma*, vol. 77, nos. 2–4, pp. 85–113, 1997.

[21] S. Balla-Arabé, X. Gao, and B. Wang, "A fast and robust level set method for image segmentation using fuzzy clustering and lattice Boltzmann method," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 910–920, Jun. 2013.

[22] P. Shen and C. Li, "Distributed information theoretic clustering," *IEEE Trans. Signal Process.*, vol. 62, no. 13, pp. 3442–3453, Jul. 2014.
[23] X. Zhao and A. H. Sayed, "Distributed clustering and learning over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3285–3300, Jul. 2015.
[24] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, New Orleans, LA, USA, Jan. 2007, pp. 1027–1035.
[25] Z. Lin, B. Francis, and M. Maggiore, "State agreement for continuous-time coupled nonlinear systems," *SIAM J. Control Optim.*, vol. 46, no. 1, pp. 288–307, 2007.
[26] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 71–82, Apr. 2007.
[27] J. Qin, H. Gao, and W. X. Zheng, "Second-order consensus for multi-agent systems with switching topology and communication delay," *Syst. Control Lett.*, vol. 60, no. 6, pp. 390–397, 2011.
[28] Y. Tang, H. Gao, W. Zou, and J. Kurths, "Distributed synchronization in networks of agent systems with nonlinearities and random switchings," *IEEE Trans. Cybern.*, vol. 43, no. 1, pp. 358–370, Feb. 2013.
[29] J. Qin, W. X. Zheng, and H. Gao, "Coordination of multiple agents with double-integrator dynamics under generalized interaction topologies," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 44–57, Feb. 2012.
[30] F. Garin and L. Schenato, "A survey on distributed estimation and control applications using linear consensus algorithms," in *Networked Control Systems*. London, U.K.: Springer, 2010, pp. 75–107.
[31] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proc. Amer. Control Conf.*, New York, NY, USA, Jul. 2007, pp. 711–716.
[32] C. N. Hadjicostis and A. Rikos, "Distributed strategies for balancing a weighted digraph," in *Proc. 20th IEEE Mediterr. Conf. Control Autom.*, Barcelona, Spain, Jul. 2012, pp. 1141–1146.
[33] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *Eur. J. Control*, vol. 18, no. 6, pp. 539–557, 2012.
[34] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York, NY, USA: Springer, 1981.
[35] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 370–379, Aug. 1995.

Jiahu Qin (M'12) received the first Ph.D. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2012, and the second Ph.D. degree in information engineering from Australian National University, Canberra, ACT, Australia, in 2014.

He was a Visiting Fellow with the University of Western Sydney, Sydney, NSW, Australia, in 2009 and 2015, respectively, and a Research Fellow with the Australian National University, from 2013 to 2014. He is currently a Professor with the University of Science and Technology of China, Hefei, China. His current research interests include consensus and coordination in multiagent systems as well as complex network theory and its application.



Weiming Fu received the B.E. degree in automation from the University of Science and Technology of China, Hefei, China, in 2014, where he is currently pursuing the master's degree.

His current research interests include consensus problems in multiagent coordination and synchronization of complex dynamical networks.

Huijun Gao (F'14) received the Ph.D. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2005.

He was a Research Associate with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong, from 2003 to 2004. From 2005 to 2007, he was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. Since 2004, he has been with the Harbin Institute of Technology, where he is currently a Professor and the Director of the Research Institute of Intelligent Control and Systems. His current research interests include network-based control, robust control/filter theory, time-delay systems, and their engineering applications.

Dr. Gao was a recipient of the IEEE J. David Irwin Early Career Award from the IEEE IES. He is a Co-Editor-in-Chief for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and an Associate Editor for *Automatica*, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, the IEEE/ASME TRANSACTIONS ON MECHATRONICS, and the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. He is an AdCom Member of the IEEE Industrial Electronics Society.

Wei Xing Zheng (M'93–SM'98–F'14) received the Ph.D. degree in electrical engineering from Southeast University, Nanjing, China, in 1989.

He has held various faculty/research/visiting positions with Southeast University; the Imperial College of Science, Technology and Medicine, London, U.K.; the University of Western Australia, Perth, WA, Australia; the Curtin University of Technology, Perth; the Munich University of Technology, Munich, Germany; the University of Virginia, Charlottesville, VA, USA; and the University of California at Davis, Davis, CA, USA. He is currently a Full Professor with Western Sydney University, Sydney, NSW, Australia.

Dr. Zheng served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: FUNDAMENTAL THEORY AND APPLICATIONS, the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE SIGNAL PROCESSING LETTERS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS, and the IEEE TRANSACTIONS ON FUZZY SYSTEMS, and as a Guest Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS. He is currently an Associate Editor for *Automatica*, the IEEE TRANSACTIONS ON AUTOMATIC CONTROL (the second term), the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and other scholarly journals. He has also served as the Chair of the IEEE Circuits and Systems Society's Technical Committee on Neural Systems and Applications and the IEEE Circuits and Systems Society's Technical Committee on Blind Signal Processing. He is a Fellow of IEEE.