Hindawi
Complexity
Volume 2018, Article ID 7698274, 16 pages
https://doi.org/10.1155/2018/7698274

Research Article

Self-Adaptive K-Means Based on a Covering Algorithm

Yiwen Zhang,[1] **Yuanyuan Zhou**,[1] **Xing Guo**,[1] **Jintao Wu**,[1] **Qiang He**,[2] **Xiao Liu**,[3] **and Yun Yang**[2]

[1]*School of Computer Science and Technology, Anhui University, Hefei 230601, China*
[2]*School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia*
[3]*School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia*

Correspondence should be addressed to Xing Guo; guoxingahu@qq.com

Received 29 December 2017; Accepted 26 March 2018; Published 1 August 2018

Academic Editor: Xiuzhen Zhang

The K-means algorithm is one of the ten classic algorithms in the area of data mining and has been studied by researchers in numerous fields for a long time. However, the value of the clustering number k in the K-means algorithm is not always easy to be determined, and the selection of the initial centers is vulnerable to outliers. This paper proposes an improved K-means clustering algorithm called the covering K-means algorithm (C-K-means). The C-K-means algorithm can not only acquire efficient and accurate clustering results but also self-adaptively provide a reasonable numbers of clusters based on the data features. It includes two phases: the initialization of the covering algorithm (CA) and the Lloyd iteration of the K-means. The first phase executes the CA. CA self-organizes and recognizes the number of clusters k based on the similarities in the data, and it requires neither the number of clusters to be prespecified nor the initial centers to be manually selected. Therefore, it has a "blind" feature, that is, k is not preselected. The second phase performs the Lloyd iteration based on the results of the first phase. The C-K-means algorithm combines the advantages of CA and K-means. Experiments are carried out on the Spark platform, and the results verify the good scalability of the C-K-means algorithm. This algorithm can effectively solve the problem of large-scale data clustering. Extensive experiments on real data sets show that the accuracy and efficiency of the C-K-means algorithm outperforms the existing algorithms under both sequential and parallel conditions.

1. Introduction

The development of big data technologies, cloud computing, and the proliferation of data sources (social networks, Internet of Things, e-commerce, mobile apps, biological sequence databases, etc.) enables machines to handle more input data than human being could. Due to this dramatic increase in data, business organizations and researchers have become aware of the tremendous value the data contain. Researchers in the field of information technology have also recognized the enormous challenges these data bring. New technologies to handle these data, called big data, are required. Therefore, it is vital for researchers to choose suitable approaches to deal with big data and obtain valuable information from them. Recognizing valuable information in data requires the use of ideas from machine learning algorithms. Thus, big data analysis must combine the techniques of data mining with those of machine learning. Clustering is one such method that is used in both fields. Clustering is a classic data mining method, and its goal is to divide datasets into multiple classes to maximize the similarities of the data points in each class and minimize the similarities between the classes. The cluster analysis method has been widely used in many fields of science and technology, such as modern statistics, bioinformatics, and social media analytics [1–5]. For example, clustering algorithms can be applied to social events to analyze big data to determine peoples' opinions, such as predicting the winner of an election.

Based on the characteristics of different fields, researchers have proposed a variety of clustering types, which can be divided into several general categories, including hierarchy clustering, density-based clustering, graph theory-based clustering, grid-based clustering, model-based clustering, and

partitional clustering [1]. Each clustering type has its own style and optimization approaches. We focus on partitional clustering algorithms. The most popular algorithm is K-means [2, 3, 6, 7], which is one of the top ten clustering algorithms in data mining. The advantages of the K-means algorithm are its easy implementation and understanding, whereas its disadvantages are that the number of clusters k cannot be easily determined and the selection of the initial centers is easily disturbed by outliers, which has a significant impact on the final results [6]. Due to the simple iteration of the K-means algorithm, it has good scalability when dealing with big data and is easy to implement in parallel execution [8–10]. Researchers have proposed improved K-means algorithms to address the drawbacks of the K-means algorithm, and most of the improvements were made by optimizing the selection of the initial K-means centers [11–13]. Good initial centers can significantly affect the performance of the Lloyd iterations in terms of quality and convergence and eventually help the K-means algorithm to obtain the nearly optimal clustering results.

However, K-means and its improved algorithms still need to ascertain the number of clusters in advance and then determine the best data partitioning based on this parameter. However, the obtained results do not always represent the best data partitioning. To address these problems, this paper proposes a K-means clustering algorithm that is combined with an improved covering algorithm, which is called the C-K-means algorithm. Our improved covering-initialized algorithm has "blind" features. Without determining the number of clusters in advance, the algorithm can automatically identify the number of clusters based on the characteristics of the data and is independent of the initial centers. The C-K-means algorithm combines the advantages of the CA and K-means algorithms; it has both the "blind" characteristics of the CA and the advantages of fast, efficient, and accurate clustering of high dimensional data of the K-means algorithm. Moreover, CA is easy to implement in parallel and has good scalability. We implemented the parallel C-K-means clustering algorithm and baseline algorithms in the Spark environment. The experimental results showed that the proposed algorithm is suitable for solving the problems of large-scale and high-dimensional data clustering.

In particular, the major contributions of this paper are as follows:

(1) We propose a covering-based initialization algorithm based on the quotient space theory with "blind" features. The initialization algorithm requires neither the number of clusters to be prespecified nor the initial centers to be manually selected. CA determines the appropriate number of clusters k and the k-specific initial centers quickly and adaptively.

(2) The convergence algebra of the Lloyd iterations of the C-K-means clustering algorithm is much simpler than that of baseline algorithms.

(3) The parallel implementation of C-K-means is much faster than parallel baseline algorithms.

(4) Extensive experiments on real datasets show that the proposed C-K-means algorithm outperforms existing algorithms in both accuracy and efficiency under sequential and parallel conditions.

The remainder of this paper is organized as follows. Section 2 provides an overview of the related work. Section 3 gives an introduction to baseline algorithms and details of the C-K-means algorithm under both sequential and parallel conditions. Section 4 presents the experimental results and analysis, and Section 5 concludes the paper with future work identified.

2. Related Work

As a classic clustering algorithm, the K-means algorithm is widely used in the fields of database and data anomaly detection. Ordonez [14] implemented efficient K-means clustering algorithms at the top of a relational database management system (DBMS) for efficient SQL. They also implemented an efficient disk-based K-means application that takes into account the needs of the relational DBMS [15]. Efficient parallel clustering algorithms and implementation techniques are key to meet the scalability and performance requirements for scientific data analysis. Therefore, other researchers have proposed parallel implementation and applications of the K-means algorithm. Dhillon and Modha [16] proposed a parallel K-means clustering algorithm based on a message passing model, which utilized the inherence of the K-means algorithm. Due to data parallelism, as the amount of data increases, the speedup and extendibility of the algorithm improve. Zhao et al. [8] implemented a K-means clustering algorithm based on MapReduce, which significantly improved the efficiency of the K-means algorithm. Jiang et al. [17] proposed a two-stage clustering algorithm to detect outliers. In the first stage, the algorithm used improves K-means to cluster the data. In the second stage, while searching for outliers in the clustering results of the first stage, it identifies the final outlier. Malkomes et al. [18] used the k-center clustering variant to handle noisy data, and the algorithms used are highly parallel. However, the selection of the initial center point of the K-means algorithm is easily disturbed by abnormal points, which has a significant impact on the final results. However, efficient methods to solve the issue in which the K-means algorithm is influenced by the initial centers have not been proposed.

Recently, scholars have focused on research into the issue that the selection of the initial centers of the K-means algorithm is easily disturbed by outlier points and have proposed several improved algorithms to help the K-means algorithm select the initial centers. The most classic improved algorithms are the K-means++ algorithm and the K-means|| algorithm. The K-means++ algorithm, which was proposed by Arthur and Vassilvitskii [12], helps the K-means algorithm to obtain the initial centers prior to the Lloyd iteration. It randomly selects a data point as the first cluster center, which is followed by selection based on the probability of the number of data points constituting the center point of

the initial set of k. The probability of selecting each successive center point is dependent on the previously selected cluster centers. However, due to the inherent sequential execution characteristics of K-means++, the k clustering centers must traverse the datasets k times and the current clustering center calculation depends on all of the previously obtained clustering centers, which makes the K-means++ initialization algorithm difficult to implement in parallel. Inspired by the K-means++ algorithm, Bahmani et al. [13] proposed the K-means|| algorithm to improve the performance of the parallelization and initialization phases. The K-means|| initialization algorithm introduces oversampling factors, obtains initial centers that are much larger than the value of k after a constant number of iterations, and assigns the weights to the center points. It then reclusters these weighted center points using the known clustering algorithm to obtain the final initial centers containing k points. K-means|| initialization has the advantages of the K-means++ algorithm and also addresses the drawback of K-means++ being difficult to extend. In follow-up research, researchers have proposed more improved algorithms of K-means and most are compared to these two classic improved algorithms. Cui et al. [10] proposed a new method of optimizing K-means based on MapReduce to process large-scale data, which eliminated the iterative dependence and reduced the computational complexity. Wei [19] improved the K-means++ algorithm by selecting the cluster centers using the sampling method in the K-means++ algorithm and then producing k centers with the expectation of having an approximately constant factor for the best clustering result. Newling and Fleuret [20] used the CLARANS to help K-means solve the problem of selecting k initial centers.

However, the number of clusters k in the K-means algorithm and its variations must be known in advance, and the best data division based on this parameter is then defined. The data division defined in this way is actually based on an imaginary model; it is not necessarily suitable for the best data division. In addition, the final clustering result is based on clustering under a hypothetical parameter without considering the actual structural relationship of the data.

In response to the problems described above, this paper presents a novel clustering algorithm called C-K-means that has both the "blind" feature of the CA and the fast, efficient clustering advantage of the K-means algorithm. It can be applied to high-dimensional data clustering with strong scalability. We implement the parallelized C-K-means algorithm on the Spark cloud platform. Extensive experimental results show that the C-K-means clustering algorithm is more accurate and efficient than the baseline algorithms.

3. The Algorithms

In this section, we first introduce the K-means clustering, K-means++ clustering, and K-means|| clustering algorithms. The motivation for using the CA as the initialization algorithm of the C-K-means clustering algorithm is then introduced, and the reason that the CA initialization can obtain clustering results that are approximately optimal is explained. Finally, we implement the parallel C-K-means algorithm. Before explaining these questions, we summarize the notions used throughout this paper in Table 1.

Table 1: Mathematical notations.

Symbol	Explanation
$X = \{x_1, \ldots, x_n\}$	Denotes a set of points in the d-dimensional Euclidean space
$m = \lvert X \rvert$	Denotes that there are m data points in dataset X
k	Denotes a positive integer specifying the number of clusters
$C = \{c_1, \ldots, c_k\}$	Denotes the set of cluster centers
$\lVert x_i - x_j \rVert$	Denotes the Euclidean distance between x_i and x_j
$Y \subseteq X$	Denotes that Y is a subset of X
$d(x, Y)$	Denotes the minimum Euclidean distance between x and set Y
centroid(Y)	Denotes the centroid of set Y
$\min_{y \in Y} \lVert x - y \rVert$	Denotes the minimum Euclidean distance between x and y in set Y
$\phi_Y(C)$	Denotes the cost of Y with respect to C
$\sum_{y \in Y} d^2(y, C)$	Denotes the sum of the squares of the minimum Euclidean distance between y in set Y and set C
$\sum_{y \in Y} \min_{i=1,\ldots,k} \lVert y - c_1 \rVert^2$	Denotes the sum of the minimum Euclidean distances between y in set Y and set $C(c_i \in C)$
ϕ^*	Denotes the cost of optimal clustering algorithms
$\boldsymbol{\sigma}_i = (\sigma_{1i}, \sigma_{2i}, \ldots, \sigma_{ni})^T$	Denotes the standard deviation vector of a cluster

3.1. State-of-the-Art Algorithms

3.1.1. K-Means. The K-means algorithm is one of the most classic clustering algorithms, because of its simple and fast performance, leading it to be widely-used. The description of the K-means algorithm is shown in Algorithm 1. First, we randomly select k data points from the original dataset X as the initial k cluster centers denoted by C, and we then calculate the distance between each data point x_i in X and each center in the initial centers C. Each data point can independently determine which center is closest to it, given an assignment of data points to clusters, the closest center is denoted by c_j. Then, the center of each cluster is updated, and each data point is repeatedly assigned to the cluster of the nearest center until the new set of cluster centers is equal to or less than the set of former cluster centers. This local search is called Lloyd iteration. The simple iteration of the K-means algorithm gives it good flexibility and can work effectively even with today's big data. Algorithm 1 presents the pseudocode for the K-means algorithm [6, 12, 13].

3.1.2. K-Means++. Because the selection of the initial centers has a significant influence on the K-means clustering results,

Input: X, θ
Output: A set of clusters C_1, C_2,…
Begin
1: $C \leftarrow$ sample k points uniformly at random from dataset X
2: $C\_new \leftarrow C$, $C\_old \leftarrow \phi$
3: **while** $|C\_new - C\_old| \leq \theta$ **do:**
4: $C\_old \leftarrow C\_new$
5: calculate all of the distances between x_i and C_old_j:
get_distance (x_i, C_old_j), $x_i \epsilon X$, $C\_old_j \epsilon\ C\_old$
6: assign x_i to the nearest C_old_j
7: calculate new centroid C_new:

$$C\_new_i = \left(\frac{1}{|C\_old_i|} \sum_{i=1}^{C\_old_i} x_i \right)$$

8: **end while**
End

Algorithm 1: (K-means algorithm).

Input: X
Output: Initial center set C
Begin
1: $C \leftarrow$ sample a point uniformly at random from dataset X
2: **while** $|C| < k$ **do:**
3: sample $x \in X$ with probability $\frac{d^2(x, C)}{\phi_X(C)}$
4: $C \leftarrow C \cup \{x\}$
5: **end while**
End

Algorithm 2: (K-means++ initialization).

the K-means algorithm can only find a local optimal solution. To obtain the global optimal solution, it may be necessary to select the initial centers several times and then acquire the final values by constantly choosing these initial centers.

To overcome the disadvantages of K-means, researchers have proposed improved methods to help K-means find suitable initialization centers. K-means++, which was proposed by Arthur and Vassilvitskii [12], is a typical representative algorithm (shown in Algorithm 2). The main idea of this algorithm is to select the initial centers one by one in a controlled way, and the calculation of the current cluster centers depends on all of the previously obtained cluster centers. Intuitively, the initialization algorithm selects relatively decentralized initial center points for K-means clustering, and the K-means++ initialization algorithm prioritizes the data points away from the previously selected centers when selecting a new clustering center. However, from the scalability point of view, the main disadvantage of K-means++ initialization is its inherent sequential execution properties. The acquisition of k centers must traverse the entire dataset k times, and the calculation of the current cluster center relies on all of the previously obtained clustering centers, which makes the algorithm not scalable in parallel and therefore greatly limits the applications of the algorithm to large-scale datasets. Algorithm 2 presents the pseudocode for the K-means++ algorithm [12].

3.1.3. K-Means||. Based on the advantages and disadvantages of the two initialization algorithms described above, researchers have proposed a new initialization algorithm called K-means|| [13] (see Algorithm 3 for details). The main idea of this algorithm is to change the sampling strategy during each traverse and propose an oversampling factor $l = \Omega(k)$. Each time the sample points are traversed in a nonuniform way and the sampling process is repeated for approximately $O(\log \psi)$ iterations, $O(\log \psi)$ is the clustering cost of the selected centers. We can then obtain the centers of $lO(\log \psi)$ sample points with repeated sampling. The number of intermediate centers is larger than k and much smaller than the original data size. Line 7 of Algorithm 3 shows that the center points in the set of center points C are assigned weights, and the center points of these weights are then reclustered in line 8, that is, the clustered k centers obtain the final k centers. Finally, these k points are fed into the Lloyd iteration as the initial centers. Algorithm 3 presents the pseudocode for the K-means|| algorithm [13].

3.2. Intuition behind the Proposed Algorithm. The traditional K-means random initialization method requires only one iteration and selects k centers uniformly and randomly. The K-means++ initialization method improves the method by randomly selecting the center point by selecting the initial center in a nonuniform way, but it requires k iterations. Only one data point is selected for each iteration to join the set of center points. Moreover, the selection of the current center point depends on the previously selected center. K-means++, which is a constantly updated nonuniform selection operation, increases the accuracy of K-means++ over random initialization, but it makes the K-means++ algorithm difficult to expand on a big dataset. Therefore, researchers proposed the K-means|| algorithm to improve the shortcomings of random initialization and K-means++ initialization and to choose k initial centers in a nonuniform manner with fewer iterations. However, both the K-means algorithm and its variant algorithms require the input of the clustering parameter k in advance and must define the best data partitioning for this parameter. However, the defined division of data is

Input: X
Output: Initial center set C
Begin
1: $C \leftarrow$ sample a point uniformly at random from dataset X
2: $\psi \leftarrow \phi_X(C)$
3: **for** $O(\log \psi)$ times **do:**
4: $C' \leftarrow$ sample each point $x \in X$ independently with probability $p_x = \frac{l \cdot d^2(x, C)}{\phi_X(C)}$
5: $C \leftarrow C \cup \{x\}$
6: **end for**
7: For $x \in C$, set w_x as the number of points in X that are closer to x than any other point in C
8: Recluster the weighted points in C into k clusters
End

Algorithm 3: (K-means|| initialization).

actually based on a hypothetical value of k and may not be suitable for the best division of data, so the actual accuracy of the clustering results cannot be guaranteed.

Based on the geometric meaning of neural networks and the M-P neuron model, the covering algorithm was proposed by Zhang and Zhang [21]. It obtains a rule based on field covering and does not require the numbers of clusters and initial centroids to be prespecified. However, the traditional covering algorithm may face a problem in which some data points of the existing clusters are too large in the clustering process, which results in unreasonable clustering results. Therefore, based on the quotient space theory, we propose a covering algorithm called CA. The concept of granularity was first proposed by Zadeh in the 1970s [22], and Zhang and Zhang proposed the theory of quotient space [23]. This theory provided a reasonable formal model for mankind's ability to analyze and synthesize problems on a macroscopic and granular scale. Different granularities describe information at different levels. When the granularity is too small, all of the data points are self-formed and the inner knowledge cannot be mined. When the granularity is too coarse, all of the data are aggregated into a cluster, so some properties of the problems are obscured. Granularity is introduced to scientifically accomplish the task of covering clustering and obtain the optimal clustering results.

The CA requires neither the number of clusters to be prespecified nor the initial centers to be manually selected, and it automatically finds a set of fields that can separate samples with low similarity and merge samples with high similarity. The center of the set constitutes the initial clustering centers. Therefore, the CA has the beneficial feature of being "blind". Without knowing the number of clusters a priori, based on the relationships of the data, the CA can automatically identify the number of clusters and has no dependence on the initial clustering centers as well as fast computational speed. The CA also has good scalability. It is easy to implement in parallel, which is suitable for data processing in a big data environment. Therefore, this paper uses the improved CA as a K-means initialization algorithm to obtain the set of initial center points.

3.3. Overview of the C-K-Means Algorithm. In this section, we introduce the realization of the C-K-means clustering algorithm in detail. Figure 1 depicts the entire process of the C-K-means algorithm. The C-K-means algorithm is divided into two main phases: phase 1 and phase 2. Phase 1 performs the CA initialization, and phase 2 performs the Lloyd iterations. Next, we describe both phases in detail.

3.3.1. Phase 1: Overall Procedure of the CA. Algorithm 4 presents the pseudocode for the CA initialization. Below, we introduce the implementation process of CA in detail.

(1). Find the center of gravity of all of the sample sets X that have not been clustered (covered) and then take the point denoted by center that is closest to the gravity as the initial center of the first cluster; this process is get_center (C_u) in Algorithm 4.

(2). Find the distance r_x between each data point $x \epsilon X$ and center that has not been clustered separately and obtain the sum of all of the distances denoted by $r_{X\rightarrow \text{center}}$. Next, $w_x = (r_x / r_{X\rightarrow \text{center}}) / (\sum_{x\epsilon X}(r_x / r_{X\rightarrow \text{center}}))$ we set the weight $w_x = (r_x / r_{X\rightarrow \text{center}}) / (\sum_{x\epsilon X}(r_x / r_{X\rightarrow \text{center}}))$ on all data points. Finally, we use r_x and w_x to calculate the covering radius, radius $= \sum_{x\epsilon X} r_x w_x$, which is introduced in get_weight_radius(c,C_u) in Algorithm 4.

(3). Find the centroids of the current spheres continually according to the obtained center and radius and obtain new clusters until the number of clusters in the data points does not increase. We can then determine the spheres (covering or clustering), which is introduced in get_covering (c, r, C_u) and lines 10 to 15 in Algorithm 4.

(4). Repeat steps (1), (2), and (3) until all of the data points have been completely covered. This is introduced in lines 3 to 16 in Algorithm 4.

During the data clustering process, we can also automatically adjust the inner class and interclass relationships based

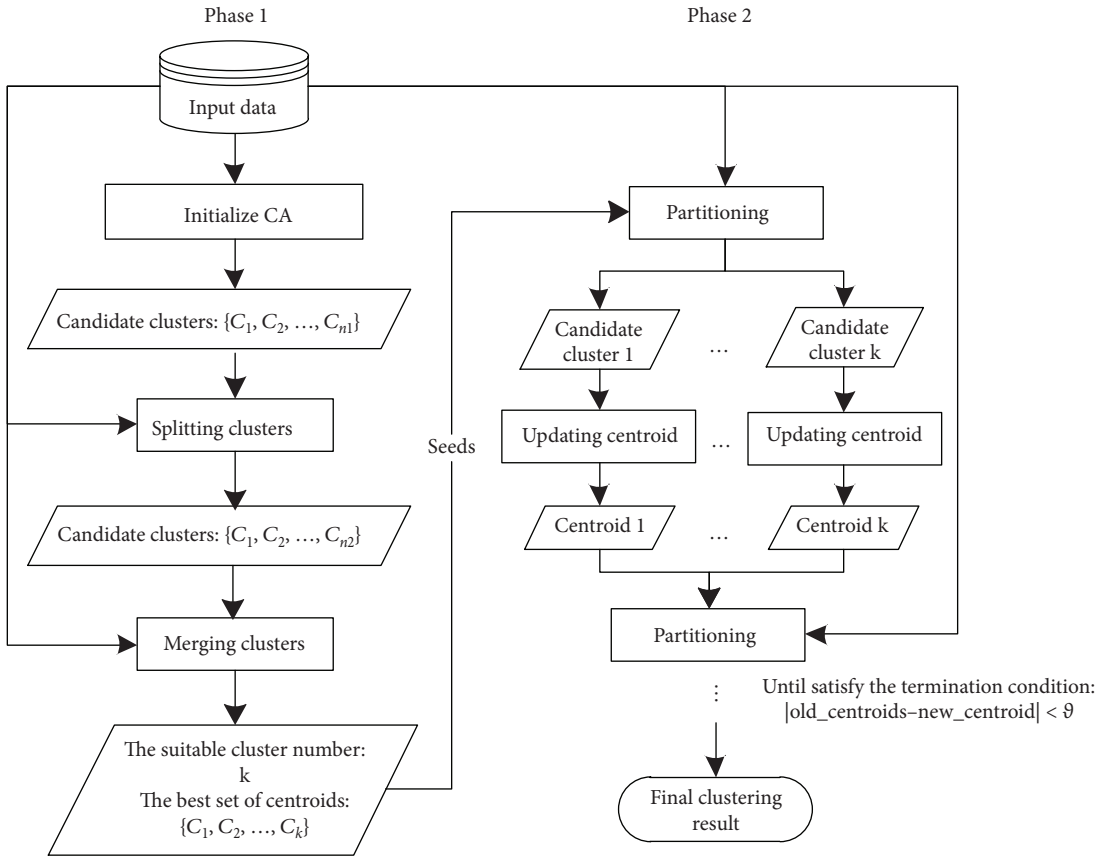


Figure 1: Overall procedure of the C-K-means algorithm.

Input: X
Output: Results of parallel covering with granularity analysis–A set of clusters $C = \{C_1, C_2, \dots\}$
Begin
center c = null
Set $C_u = X$
do
  center $c \leftarrow$ get_center(C_u)
  radius $r \leftarrow$ get_weight_radius(c,C_u)
  Covering C_{form}=get_covering(c,r,C_u)
  c $\leftarrow$ get_centroid(C_{form})
  $r \leftarrow$ get_weight_radius(c,C_u)
 Covering C_{last}=get_covering(c,r,C_u)
 while C_{last}.subtractByKey(C_{form})>0
    $C_{form} \leftarrow C_{last}$
    c $\leftarrow$ get_centroid(C_{form})
    $r \leftarrow$ get_radius_centroid(c,C_u)
    C_{last}=get_covering(c,r,C_u)
 end while
while ($C_u \neq \varnothing$)
 Do Split Operation
 Do Merge Operation
return $C = \{C_1, C_2, \dots\}$
End

Algorithm 4: (CA initialization).

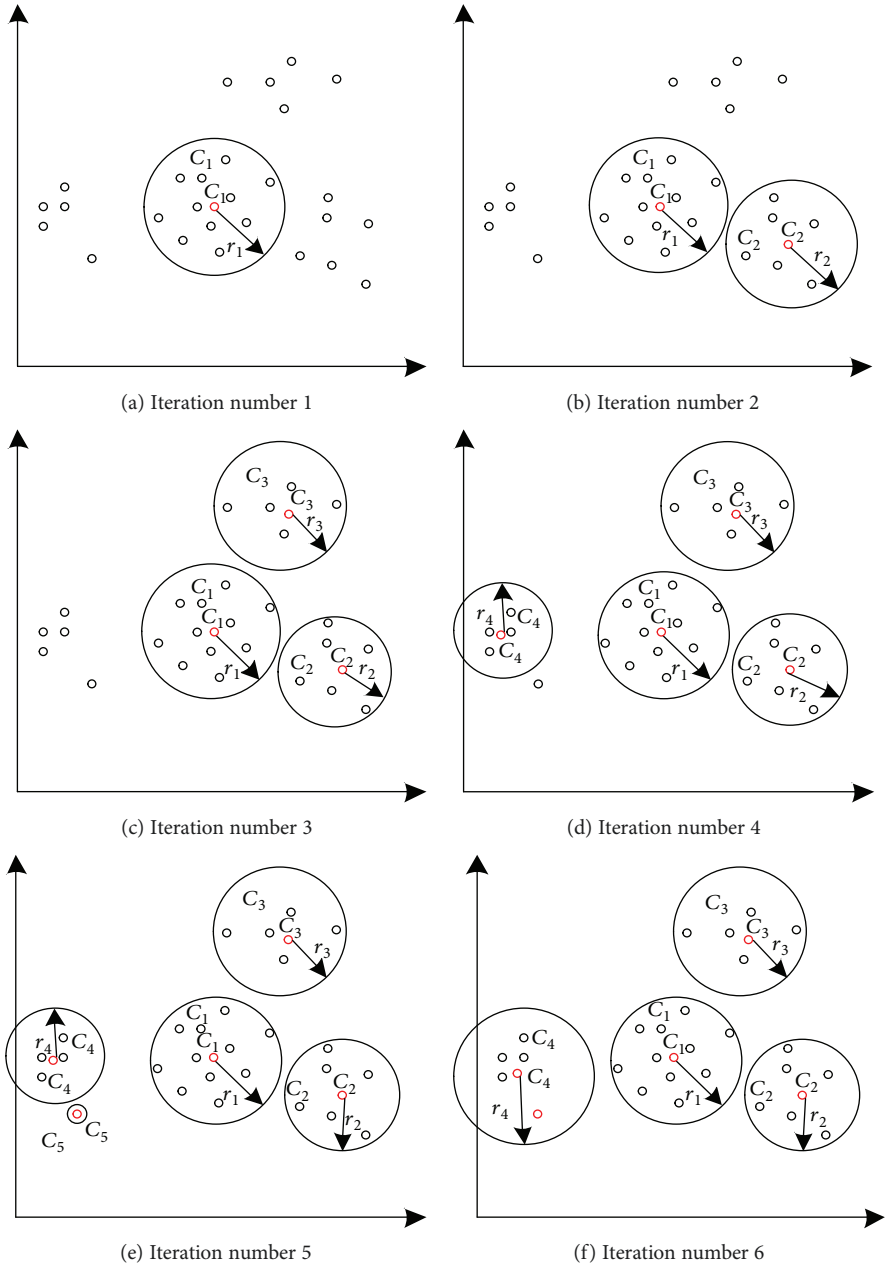


Figure 2: An example of clustering.

on the actual demand or the relationship between the data in the dataset. For a covering with fewer sample points, the single linkage method (using the Euclidean distance) in the hierarchical clustering algorithm [24, 25] is adopted to merge them to form an ellipsoidal domain, which means combing the most similar pair of clusters into a new cluster. Then, the similarities between the new cluster and the other clusters are updated, and the two most similar clusters are again merged. Based on the relationship between the data in the dataset or the actual demand, we can decide whether to continue merging the clusters with fewer data points or to split the spheres with more data points. Finally, we can obtain reasonable covering divisions with all of the similar data points that are distributed in one area (spherical or ellipsoidal), which is introduced in lines 17 and 18 in Algorithm 4.

Figure 2 presents an illustrative example to intuitively demonstrate the clustering process of Algorithm 4. To cluster the data points, Algorithm 4 goes through five iterations to identify five clusters (covering or fields), $C_1, \ldots, C_5$. We then compute the relationship between the inner class and the interclasses and find that clusters C_4 and C_5 are very similar. Therefore, the sixth iteration merges them into one cluster and then updates the similarities between each cluster, where $c_1, \ldots, c_5$ are the centers and $r_1, \ldots, r_5$ are the radii, respectively.

When we study a dataset, we can divide it in different ways. Each division is a quotient space of different

granularities. We observe and analyze this dataset from different granularities. Based on the different granularities of the observation and analysis datasets, we can solve the problem in different granular worlds and can jump quickly from one granular world to another. This ability to handle different worlds of granularity is a powerful manifestation of the solution of human problems [26]. When we study the problem of reasonably clustered datasets, we can put the problem in the quotient space with different granularities for analysis. We can then obtain the solution to the clustering problem synthetically. In a different granularity quotient space, we can observe the different nature of the dataset and then find the properties of interest to the user, which can be maintained in different granular worlds or preserved to a certain extent. However, not every arbitrary division can achieve this goal. Therefore, the dataset division and its choice of granularity must be studied, that is, we need to select the appropriate dataset division. Based on the above, we propose the split-operation and merge-operation mechanisms in the C-K-means algorithm to help the datasets determine the appropriate partitioning and granularity. The C-K-means algorithm automatically adjusts the number of clusters during the iteration by merging similar clusters and splitting clusters with larger standard deviations. Finally, after a small number of constant iterations, C-K-means helps the dataset find the appropriate number of clusters k and k initial centers, and it then feeds the clustering centers into the Lloyd iteration to complete the final clustering process and determine the reasonable quotient space for the original dataset.

Adjustment Mechanism 1: Split Operation. First, we calculate the vector of the standard deviations for all of the samples in the cluster to the center of the cluster in all of the clusters: $\boldsymbol{\sigma}_i = (\sigma_{1i}, \sigma_{2i}, \ldots, \sigma_{ni})^T$, $i = 1, 2, \ldots, N_c$, where N_c is the number of existing classes and n is the dimension of the samples. We then calculate the maximum component on $\sigma_{i\,\max}$ of the standard deviation vector $\boldsymbol{\sigma}_i$ of each class and determine the threshold value σ_s. For cluster C_u, we consider the following conditions: (1) the maximum component-wise standard deviation in the cluster, that is, $\max_{j=1,\ldots,n}\sigma_{uj} > \sigma_s$; (2) the average distance between the samples in the cluster is greater than the overall average distance, that is, $\bar{d}_i > \bar{d}$, where $\bar{d}_i$ and $\bar{d}$ represent the average inner class distance of the i cluster (i.e., the average distance from the sample to the centroid in the calculation cluster) and the overall average distance (i.e., the overall average distance of each sample to its inner class centers), respectively; (3) the number of samples in the cluster is greater than θ_N, that is, $|C_u| > \theta_N$, where θ_N is the threshold cluster number, θ_N is the minimum number of samples allowed in each cluster (if less than this number, it cannot form a cluster), and $|C_u|$ denotes the number of samples in the ith cluster; and (4) the number of clusters is greater than $k/2$. If all of these conditions are satisfied, then split cluster C_u into two clusters with two cluster centers C_{u+} and C_{u-} and delete the original class C_i. The current number of clusters will increase by 1. The values of C_{u+} and C_{u-} are the components corresponding to $\sigma_{i\,\max}$ in the original C_u that to $\sigma_{i\,\max}$ are added to and subtracted from, respectively, while their components remain unchanged.

Adjustment Mechanism 2: Merge Operation. To sort the numbers of points contained in all clusters that have been formed, for clusters with fewer points, we calculate the similarity values between all other clusters and them: $S_{ij} = 1/1 + d_{ij}$, $i = 1, 2, \ldots, N_{c-1}$ and $j = 1, 2, \ldots, N_c$. To sort all of the obtained S_{ij}, values according to the value of the final number of clusters k; we merge the two clusters with the largest S_{ij} values and update the merged cluster centers. The current number of clusters will decrease by 1.

3.3.2. Phase 2: Overall Procedure of Lloyd's Iterations. Phase 1 determines the suitable value of k and k specific initial centers by performing the CA initialization. In phase 2, we assign the data points in the dataset to the cluster whose center is closest to the data point according to the cluster centers obtained in phase 1. We then update the class centers until the convergence condition is satisfied. All of the data are distributed to the cluster when the data point is closest to the cluster center, that is, the Lloyd iteration of the K-means clustering algorithm is completed, and the clustering results near the optimal clustering solution are obtained to complete the proposed C-K-means algorithm. Our CA initialization and final C-K-means algorithm can be easily parallelized, and we can rapidly complete the clustering operations.

3.4. Computational Complexity Analysis. This section discusses the computational complexity of the C-K-means algorithm with two phases. First, we analyze the computational complexity of the forming phase of C-K-means (i.e., CA initialization). In Algorithm 4, the computational complexity of line 5 is $O(m)$ because dataset X contains a maximum of m points. Similarly, the computational complexities of lines 5 and 6 are also $O(m)$, and those of lines 7, 8, and 9 are also $O(m)$ because the number of clusters is smaller than m. Lines 10–15 will be repeated until the data points in the cluster do not change. Lines 3–15 must also be repeated until all of the data points in X are covered, and the number of repetitions num_C is much smaller than m. In line 3, the radius of a cluster is the average distance between the center of the cluster and all of the data points that are not covered by any clusters. On average, each newly created cluster covers half of the uncovered data points, and the computational complexity is $O(\log m)$. In line 17, the computational complexity is $O(p)$ because there is a maximum of p clusters after the initial covering process. Similarly, the computational complexity of line 18 is $O(p)$ because there is a maximum of p clusters. The number of clusters is much smaller than m. Thus, the computational complexity of Algorithm 4 is $O(m) \times O(\log m) + O(p) = O(m \log m)$. We then introduce the second phase's computational complexity, which is the Lloyd iterations. The second phase performs num_iter iterations until the cluster centers do not change, so its computational complexity is $O(k * m * num\_iter)$. The numbers of clusters k and iterations num_iter are much smaller than m. Therefore, the computational complexity of the C-K-means algorithm is $O(m) \times O(\log m) + O(p) + O(k * m * num\_iter) = O(m \log m)$.

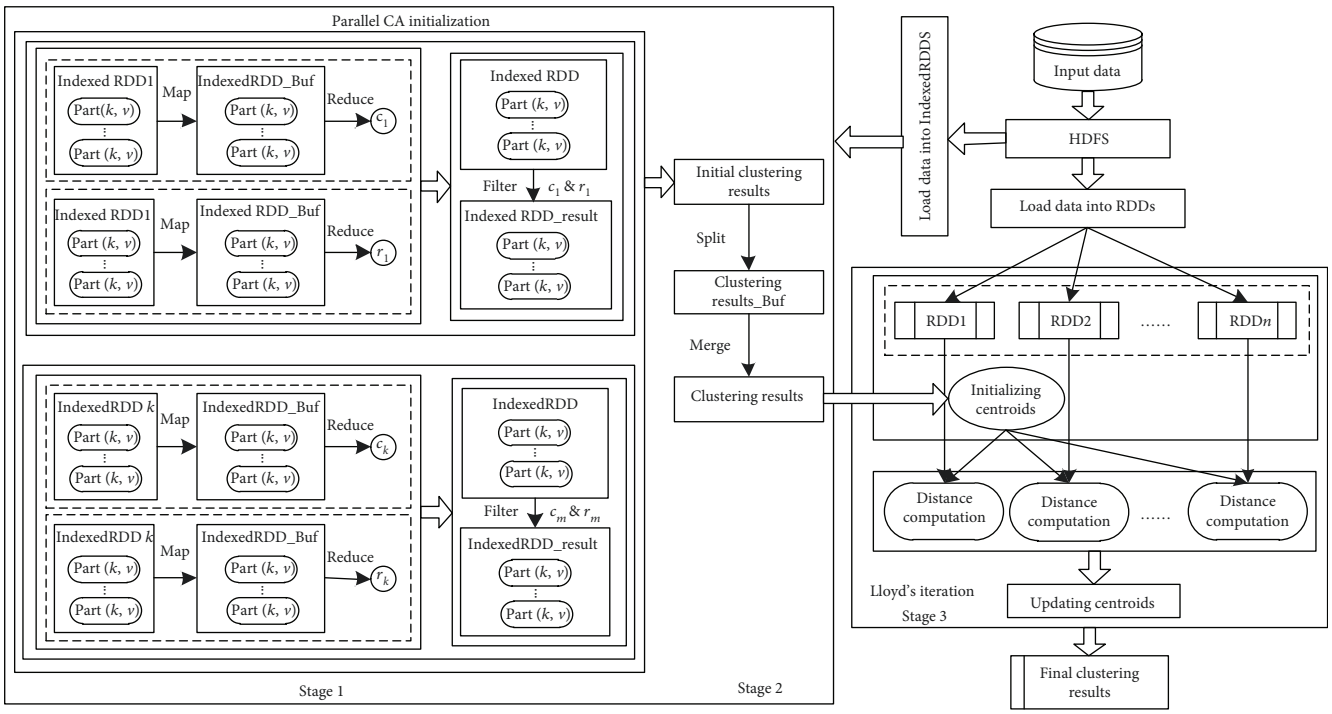


Figure 3: Overall procedure of the parallel C-*K*-means algorithm.

3.5. A Parallel Implementation. In this section, we discuss the proposed CA initialization and the parallel implementation of the C-*K*-means algorithm on Spark.

Spark is the de facto distributed computing platform for large data processing and is particularly suitable for iterative calculations. A main component of Spark is the *resilient distributed dataset* (RDD), which represents a read-only collection of objects partitioned across multiple machines that can be rebuilt if a partition is lost. Users can explicitly cache an RDD in memory across multiple machines and reuse it in multiple parallel operations. The RDD is the main reason that Spark is able to process big data efficiently. Due to the performance of memory computing, data locality, and transport optimization of Spark, it is particularly suitable for performing recursive operations on big data [27]. However, not all large-scale data can be efficiently processed via parallel implementation. Partitional clustering algorithms require an exponential number of iterations [28]. Simultaneously, exponential job creation time and time of large-scale data shuffling are difficult to accept, especially for large amounts of data, so mere parallelism is not sufficient. High performance can be reached only by eliminating the partitional clustering algorithm's dependence on the iteration.

The parallel implementation principle of the C-*K*-means clustering algorithm in Spark is illustrated in Figure 3. As demonstrated, C-*K*-means consists of three main stages. Stage 1 performs the parallel CA on Spark, and stage 2 analyzes the results of the initial covering clustering obtained from Stage 1 and splits or merges the clustering results through self-organization to determine the number of clusters k and the specific initial center set. Together, stages 1 and 2 constitute the parallel CA initialization process. Stage 3 is the Lloyd iteration phase, in which Lloyd iteration is conducted on k initial centers to obtain the optimal clustering results.

The covering algorithm implemented on Spark is illustrated in stages 1 and 2 in Figure 3. The distributed files are read from Hadoop Distributed File System (HDFS) and transformed into IndexedRDD [29]. The parallel covering process in stage 1 consists of many covering processes. Each covering process comprises three processes, which obtain the cluster and its center, radius, and the cluster, respectively. Stage 1 describes the process for obtaining all of the clusters. We obtain the first cluster center c_1 through the *reduce* operation on Spark. This operation obtains the data point that is nearest to the centroid of all of the data in parallel. Next, we obtain the radius r_1 of the cluster through the *map* and *reduce* operations on Spark. Specifically, an intermediate variable IndexedRDD_Buf is obtained through the *map* operation on Spark. The *map* operation calculates the distance between the cluster center c_1 and each uncovered data point and forms IndexedRDD_Buf. Then, the radius r_1 is obtained through the *reduce* operation. This operation produces the radius r_1 by calculating IndexedRDD_Buf in parallel. Finally, we obtain *cluster_1* through the *filter* operation on Spark. Simultaneously, the *filter* operation filters the data points, where the distances between center c_1 and each uncovered data point are less than the radius r_1. The radius and center are acquired using the process introduced above. The remaining clusters are obtained in a manner similar to the first cluster. These processes are repeated until no more data points can be identified, which indicates that all of the data points have been included in these clusters. This is the end of the covering process, which also indicates that stage 1 is complete. After the covering process, C-*K*-means

Table 2: Description of seven datasets.

Dataset	Number of attributes	Number of instances	Number of clusters (k)
Iris	4	150	3
Wine	13	178	3
Abalone	8	4177	29
Gauss	3	10,000	?
SPAM	57	4601	?
Cloud	10	1024	?
Individual household	7	2,049,280	?

performs the split and merge operations in stage 2 to obtain the final initialization centers. Through the CA initialization process, the initialization centers are adaptively obtained and then fed into Lloyd's iteration in stage 3. As described earlier, Lloyd's iteration can also be easily parallelized on Spark. Therefore, it is imperative that we implement an efficient CA initialization and C-K-means algorithm on the Spark platform.

4. Experimental Results

This section presents a detailed analysis and comparison of the experimental results, including sequential and parallel versions of the algorithm to confirm the merits of our C-K-means algorithm, which include the following: (1) the C-K-means algorithm can adaptively determine the number of clusters k and obtain a set of k cluster center points according to the similarity between the data, which then allows the C-K-means algorithm to obtain high-precision clustering results, (2) the C-K-means algorithm can obtain a clustering result that is near the optimal value which outperforms K-means in terms of its cost and is very similar to k-means++ and k-means||, and (3) compared with k-means++ and k-means||, the number of Lloyd's iterations in the C-K-means algorithm is relatively small which converges quickly when accuracy and cost are ensured, meaning that the proposed C-K-means algorithm is accurate and efficient under parallel conditions.

In this paper, the C-K-means clustering algorithm and its counterparts are implemented sequentially and in parallel. The sequential implementation is evaluated on a standalone computer with a 6-core 3.60 GHz processor and 20 GB of memory. All of the parallel algorithms are implemented on a cluster of Spark 1.6 with Hadoop 2.6. The cluster has 16 nodes, each of which is an 8-core 3.60 GHz processor with 20 GB memory.

4.1. Datasets. We used 7 datasets in our experiments to evaluate the performance of the C-K-means algorithm. The summary statistics and information about these 7 datasets are shown in Table 2.

The question marks in Table 2 indicate that the number of clusters in the dataset is unknown.

Some of the datasets, such as Gauss, are synthetic, and the others are from real-world settings and are publicly available from the University of California Irvine (UCI) machine learning datasets [30]. The Iris dataset [31–33] is a well-known database in clustering algorithm comparisons. It consists of three types of *Iris* plants (*setosa*, *versicolor*, and *virginica*) with 50 instances, each of which was measured with four features. The Wine dataset [31–33] is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It contains 178 instances measured with 13 continuous features. The Abalone dataset [34, 35] contains physical measurements of abalone shellfish. It contains 4177 instances with 9 features each (1 cluster label and 8 numeric and we apply 8 primary features), which are divided into 29 clusters. The age of an abalone can be determined by cutting the shell through the cone, staining it, and counting the number of rings with a microscope. In practice, measurements are used to estimate the age. The SPAM dataset [13] consists of 4601 instances with 57 dimensions and represents features available to an e-mail spam detection system. The Cloud dataset [12] consists of 1024 instances in 10 dimensions and represents the 1st cloud cover database. The individual household electric power consumption dataset [10] contains 2,049,280 instances with 9 features, 7 of which are applied in this paper because the other 2 are related to time, which are not applicable.

To effectively evaluate the experimental performance of the algorithm, we normalized the datasets. All of the algorithms use datasets that are normalized to frequent cases. When the dimension of the data points in a dataset is too high, it reduces the discrimination of the other dimensions with lower values during the clustering process. We normalized the datasets in an operation by

$$x_i^j = \begin{cases} \dfrac{x_i^{\max} - x_i^j(ori)}{x_i^{\max} - x_i^{\min}}, & \text{if } x_i^{\max} \neq x_i^{\min}, \\ 1, & \text{otherwise}, \end{cases} \quad (1)$$

where $x_i^j(ori)$ and x_i^j represent the jth dimension values of the ith data point in the dataset before and after normalization, respectively, and $x_i^{\max}$ and $x_i^{\min}$ are the maximum and minimum values of the jth dimension of all data points in the dataset, respectively.

4.2. Baselines. In the remainder of this paper, we assume that both the k-means++ and k-means|| initialization algorithms implicitly follow the Lloyd iteration process. The proposed C-K-means clustering algorithm outperforms the baseline algorithms as described below:

(i) Traditional K-means algorithm (or K-means algorithm): this algorithm is based on random initialization and is often applied to randomly select k sample points as the initial centers for Lloyd's iteration and complete the final clustering process accordingly (see Algorithm 1) [6].

(ii) K-means++ algorithm: this method selects k centers as the initial centers for Lloyd's iteration through multiple iterative processes. Based on the probability of each sample point, each iteration selects 1

sample point from the dataset to join the center set and completes the final clustering process (see Algorithm 2) [12].

(iii) K-means|| algorithm: this method selects k centers as the initial centers for Lloyd's iteration through a constant number of processes. Based on the probability of each sample point, each iteration selects l sample points from the dataset to join the center set. It then reclusters the initial center set to obtain the final center set and feeds the final initial center point into Lloyd's iteration. The final clustering process is then completed (see Algorithm 3) [13].

4.3. Evaluation Metrics. The effectiveness of clustering is evaluated by numerous factors that determine the optimal number of clusters and the granularity of checking the clustering results. The evaluation of clustering results is often referred to as cluster validation, and researchers have proposed many measures of cluster validity. In this paper, we choose six standard validity measures to examine the soundness of the clustering algorithms, including Davies-Bouldins index (DBI) [10, 35, 36], the Dunn validity index (DVI) [36, 37], normalized mutual information (NMI) [38–40], the clustering cost function (ϕ), the Silhouette index (SI) [41, 42], and the SD index (SDI) [42]. These measures are described as follows:

$$\begin{aligned}
\mathrm{DBI} &= \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\bar{C}_i+\bar{C}_j}{\left\|w_i-w_j\right\|_2}\right),\\
\mathrm{DVI} &= \frac{\min\limits_{0<m\neq n\leq k}\left\{\min\limits_{\forall x_i\in\Omega m, x_j\in\Omega n}\left\{\left\|x_i-x_j\right\|\right\}\right\}}{\max\limits_{0<m\leq k}\left\{\max\limits_{\forall x_i,x_j\in\Omega m}\left\{\left\|x_i-x_j\right\|\right\}\right\}},\\
\mathrm{NMI} &= \frac{I(X,Y)}{\sqrt{H(X)H(Y)}},\\
\phi_Y(C) &= \sum_{y\in Y}d^2(y,C)=\sum_{y\in Y}\min_{i=1,\ldots,k}\|y-c_i\|,\\
\mathrm{SI} &= \sum_{0<i\leq k}\frac{b(i)-a(i)}{\max\{a(i),b(i)\}},\\
\mathrm{SDI}(k) &= a\cdot Scatt(k)+Dis(k),
\end{aligned} \tag{2}$$

where

$$\begin{aligned}
\mathrm{Scatt}(k) &= \frac{1}{k}\sum_{0<i\leq k}\frac{\|\sigma(v_i)\|}{\|\sigma(X)\|},\\
\mathrm{Dis}(k) &= \frac{D_{\max}}{D_{\min}}\sum_{0<i\leq k}\left[\sum_{0<j\leq k}\left\|v_i-v_j\right\|\right]^{-1}.
\end{aligned} \tag{3}$$

In the DBI validation measure, k denotes the number of clusters, $\bar{C}_i$ denotes the average distance within the ith cluster, and $\|w_i - w_j\|$ denotes the distance between the ith cluster and the jth cluster. In the DVI validation measure, k denotes the number of clusters and $\|x_i, x_j\|$ denotes the distance between two data points. In the NMI validation measure, X and Y denote the obtained cluster and true classes, respectively, where $I(X, Y)$ is the mutual information between X and Y and $H(X)$ and $H(Y)$ are the Shannon entropies of X and Y, respectively. The variables in the cost function ϕ are described in Table 1. In the SI validation measure, k denotes the number of clusters, $a_{(i)}$ denotes the average distance from the ith object to all of the objects in the same cluster, and $b_{(i)}$ denotes the minimum average distance from the ith object to all of the objects in a different cluster. In the SDI validation measure, k denotes the number of clusters, $\mathrm{Scatt}(k)$ denotes the average scattering of the clusters, where $\sigma(v_i)$ denotes the variance of cluster i, $\sigma(X)$ denotes the variance of data set X, and $Dis(k)$ denotes the total separation between the clusters, where $D_{\max} = \max(\|v_i - v_j\|)$ denotes the maximum distance between the cluster centers, $D_{\min} = \min(\|v_i - v_j\|)$ denotes the minimum distance between cluster centers, and a denotes the weighting factor that is equal to $\mathrm{Dis}(c_{\max})$, where $c_{\max}$ is the maximum number of input clusters. DBI is a function of the ratio of the sum of the inner cluster distribution to the intercluster separation. The lower the DBI value is, the better the clustering performance will be because the distance within the clusters is small, but the distance among the clusters is large. DVI is a function of the ratio of the intercluster distribution separation to the sum of the inner cluster distributions. The larger the DVI value is, the better the clustering performance will be because the distance among the clusters is large and the distance within the clusters is small. NMI indicates the difference between the actual data type of the original data and the data type calculated by the clustering algorithm. Therefore, the NMI validation measure requires that the actual data type and the calculated data have the same number of class elements. The NMI values are in the interval [0, 1], and a larger value means that the two clusters are very similar and also indicates a better clustering result. The value of the cost function ϕ indicates the sum of the distances from each data point to the nearest cluster center. Therefore, the lower the cost function ϕ is, the better the clustering performance will be. The purpose of SI is to calculate the average dissimilarity between points in the same cluster and a different cluster to describe the structure of the data. The SI values are in the interval [−1, 1], and a larger SI value indicates a more optimal number of clusters in the dataset. The SDI is based on the average scattering of the clustering and the total separation of clusters. The minimum SDI value indicates that k is the optimal cluster number.

4.4. Determination of an Optimal Value of k in C-K-Means. CA self-organizes and recognizes the number of clusters k based on the similarities in the data without prior knowledge. By executing the CA algorithm, we can initially obtain the approximate number of clusters k. Next, we will conduct the split-operation and merge-operation mechanisms (see Section 3.3) to help the datasets determine the appropriate partitioning and granularity. To evaluate the resultant clusters for finding the optimal number of clusters, properties

Table 3: The value of k is known (Iris and Wine; $*$ denotes the optimal value).

C-K-means	Iris				Wine			
	DBI	DVI	SI	SDI	DBI	DVI	SI	SDI
3	0.8280*	0.4958*	0.5043*	6.0402*	1.3702*	0.3683*	0.3013*	3.0225*
4	0.9792	0.3490	0.4435	6.8549	1.8091	0.2139	0.2313	3.7947
5	1.0775	0.2503	0.4100	9.4341	2.0714	0.2601	0.2055	4.0299
6	1.0612	0.2365	0.4304	10.3641	2.0543	0.2611	0.1996	4.2521
7	1.1013	0.4223	0.3416	10.1253	2.1805	0.2219	0.1259	4.7275
8	1.0926	0.4286	0.3281	10.2710	2.0461	0.2402	0.1284	5.0283
9	1.0649	0.4286	0.3200	11.1102	1.8996	0.2402	0.1337	5.1846

Table 4: The value of k is unknown (Cloud and Gauss; $*$ denotes the optimal value).

C-K-means	Cloud					Gauss			
	DBI	DVI	SI	SDI		DBI	DVI	SI	SDI
5	1.0479	0.2893	0.3611*	4.5859	9	1.1869	0.2672	0.2070	5.5891
6	1.0746	0.3469	0.3580	4.1325	10	1.1484	0.2945	0.2159	5.4769
7	1.0967	0.2935	0.3295	5.1985	11	1.1492	0.3121	0.2177	5.4715
8	1.0364	0.2748	0.3184	4.6350	12	1.1539	0.3375	0.2181	5.4793
9	1.0099	0.3428	0.3182	4.4439	13	1.1058*	0.3396*	0.2233*	5.4130*
10	0.9868	0.3516*	0.3160	4.0679*	14	1.1704	0.2788	0.2139	6.0448
11	1.0542	0.2739	0.2921	4.6985	15	1.1595	0.2230	0.2102	6.5191
12	1.0285	0.2708	0.2982	4.6481	16	1.2001	0.2535	0.2092	6.5085
13	1.0745	0.2482	0.2866	4.7405	17	1.1883	0.2697	0.2073	6.3310
14	0.9614*	0.2905	0.3036	4.4374	18	1.1653	0.2823	0.2080	6.3249

such as the cluster density, size, shape, and separability are typically examined by such as the DBI, DVI, SI, and SDI cluster validation indices. The clustering validity approach uses internal criteria to evaluate the results with respect to the features and quantities inherited from the data to determine how close the objects within the clusters are and the distances among the clusters.

Performing the CA on datasets Iris and Wine, the numbers of clusters are known (see Table 2). We initially obtain the approximate number of clusters 6 for the Iris dataset and 7 for the Wine dataset. We then conduct the split-operation and merge-operation mechanisms to get several numbers of clusters that close to 6 for the Iris dataset and 7 for the Wine dataset, respectively. The numbers of split operations are between 1 and 5 for both the Iris and Wine datasets. The numbers of merge operations need not be pregiven because they are determined by the numbers of clusters and split operations. To further evaluate the results, we choose the Cloud and Gauss datasets to execute the CA, in which the numbers of clusters are unknown (see Table 2). We initially obtain the approximate number of clusters 7 for the Cloud dataset and 13 for the Gauss dataset, respectively. Similarly, we then conduct the split-operation and merge-operation mechanisms to get several numbers of clusters that close to 7 for the Cloud dataset and 13 for the Gauss dataset, respectively. The numbers of split operations are between 0 and 6 for both the Cloud and Gauss datasets.

Table 3 shows a comparative analysis of the Iris and Wine datasets, using four validity measures. Because the numbers of clusters in the datasets are known, we can intuitively determine that the finite number k is obtained by our CA when most of the clustering indexes obtain the optimal value. Table 3 shows that 3 clusters are optimal on both datasets, which exactly match the actual numbers of clusters in the datasets. We used the results of the clusters from CA to check the performance of C-K-means in the Cloud and Gauss datasets and compared them to four existing validation indices. As shown in Table 4, the optimal validation indicators for the Cloud dataset are obtained with 10 clusters, thus the optimal cluster value is 10. For the Gauss dataset, each index shows that the optimal value is 13. The CA combined with split-operation and merge-operation mechanisms self-organizes and recognizes the reasonable number of clusters k based on the similarities in the data for any dataset.

4.5. Clustering Validation. Clustering validation is generally concerned with determining the optimal number of clusters and checking the suitability of the clustering results [10]. The evaluation of the clustering results is commonly referred to as cluster validation [10, 35, 43]. The accuracies of the baseline approaches and the C-K-means algorithm are measured in terms of three standard validity measures, namely DBI, DVI, and NMI, on datasets of different sizes. Other than the individual household dataset, the other datasets are small

Table 5: Accuracy comparison (Iris, Wine, and Abalone).

Algorithms		Dataset		
		Iris	Wine	Abalone
K-means	DBI	0.9503	1.3970	1.1342
	DVI	0.0381	0.1378	0.0094
	NMI	0.656	0.8088	0.1697
K-means++	DBI	0.9220	1.3909	1.1309
	DVI	0.0577	0.1407	0.0106
	NMI	0.6737	0.8230	0.1706
K-means||	DBI	0.8571	1.3903	1.1278
	DVI	0.0481	0.1393	0.0118
	NMI	0.7208	0.8268	0.1692
Covering	DBI	0.9608	1.4864	1.6721
	DVI	0.0860	0.1336	0.0073
	NMI	0.8342	0.7237	0.1594
C-*K*-means	DBI	**0.8280**	**1.3702**	**1.1170**
	DVI	**0.0693**	**0.1893**	**0.0105**
	NMI	**0.7419**	**0.8529**	**0.1739**

enough to be evaluated on a single machine. We compare the accuracies of C-*K*-means and the baseline approaches on the Iris, Wine, and Abalone datasets because the numbers of clusters and the labels to which the data belong are known in those datasets. The value of k is kept constant to effectively compare the C-*K*-means algorithm and the baseline algorithms. Using the split- and merge-operation mechanisms, the number of clusters of C-*K*-means is adjusted to be consistent with the number of clusters in the baseline algorithms. Table 5 shows a comparative analysis of the different approaches on the three datasets and the three validity measures. For the Iris and Wine datasets, the numbers of split operations are both 1. And for the Abalone dataset, the number of split operations is 8. To better verify the performance of the algorithms, we also choose the Gauss, SPAM, and Cloud datasets, the class categories of which are unknown for the experiments. To examine the soundness of our clusters, we discuss the DBI and DVI values of these three unknown data label datasets to those of C-*K*-means for moderate values of $k \in \{10, 20, 50\}$. For the Gauss dataset with different values of k, the numbers of split operations are 32, 4, and 10, respectively. For the SPAM dataset, the numbers of split operations are 10, 30, and 50, respectively. And for the Cloud datasets, the numbers of split operations are 6, 8, and 8, respectively. We also use other values of k and obtain similar results. The clustering results for C-*K*-means and the baseline approaches are listed in Table 6 for the Gauss dataset, Table 7 for the SPAM dataset, and Table 8 for the Cloud dataset. Obviously, the three tables show that the accuracies of proposed C-*K*-means are better than baseline approaches.

4.6. Cost. To evaluate the clustering cost of C-*K*-means, we compare it to the baseline approaches. We compare the cost of the SPAM and Gauss datasets to that of C-*K*-means for moderate values of $k \in \{20, 40, 50\}$. For the Gauss dataset

Table 6: Accuracy comparison (Gauss).

Gauss	$k = 10$		$k = 20$		$k = 50$	
	DBI	DVI	DBI	DVI	DBI	DVI
K-means	1.1511	0.0037	1.1620	0.0045	1.1121	0.0056
K-means++	1.1507	0.0051	1.1593	0.0056	1.1079	0.0061
K-means||	1.1439	0.0049	1.1593	0.0055	1.1093	0.0065
C-*K*-means	**1.1350**	**0.0061**	**1.1412**	**0.0053**	**1.1070**	**0.0081**

with different values of k, the numbers of split operations are 4, 5, and 10, respectively. For the SPAM dataset, the numbers of split operations are 5, 4, and 4, respectively. The results of the Gauss and SPAM datasets are presented in Tables 9 and 10, respectively. For each algorithm, we list the cost of the solution at the end of the initialization step before Lloyd's iteration as well as the final cost. In Tables 9 and 10, "seed" represents the cost after the initialization step and "final" represents the cost after the final Lloyd iteration. The initialization cost of C-*K*-means is similar to that of *K*-means|| and lower than that of *K*-means++. These results suggest that the centers produced by C-*K*-means, like those produced by *K*-means||, are able to avoid outliers. In addition, C-*K*-means guarantees high precision with high efficiency because CA runs very fast.

4.7. Computational Time. The individual household dataset is sufficiently large for large values of $k \in \{100, 200, 500\}$. We now consider the parallel algorithms for the individual household dataset. For the household dataset with corresponding values of k, the numbers of split operations are 6, 9, and 7, respectively. C-*K*-means is faster than *K*-means, *K*-means++, and *K*-means|| when implemented in parallel. The running time of C-*K*-means consists of two components: the time required to generate the initial solution and the time required for Lloyd's iteration to converge. The former is proportional to k. The latter is considered, and C-*K*-means is compared to the baseline approaches. Table 11 shows the total running time of the clustering algorithms. For some values of k, C-*K*-means runs much faster than *K*-means and *K*-means++. C-*K*-means runs much faster than *K*-means|| when $k \in \{100, 200\}$. However, when k is 500, the total running time of C-*K*-means is similar to that of *K*-means|| because C-*K*-means needs to split and merge many times to obtain the number of clusters, which means that the initialization occupied a large proportion of the total running time.

Next, an expected advantage of C-*K*-means is demonstrated; the initial solution discovered by C-*K*-means contributed to a faster convergence of Lloyd's iteration. Table 12 shows the number of iterations required to reach convergence of Lloyd's iteration for the Cloud dataset with different initializations. C-*K*-means typically requires fewer iterations than the baseline approaches to converge to a local optimal solution. The convergence times of the iteration for datasets of different dimensions are also evaluated, and the Gauss and SPAM datasets are selected to verify the performance of the proposed C-*K*-means algorithm. The graphical representations of the number of iterations required to reach

Table 7: Accuracy comparison (SPAM).

SPAM	$k = 10$		$k = 20$		$k = 50$	
	DBI	DVI	DBI	DVI	DBI	DVI
K-means	2.3282	0.0010	2.0349	0.0007	1.7914	1.3874e-5
K-means++	2.2716	0.0020	1.8876	0.0044	1.5760	0.0042
K-means||	1.9924	0.0023	1.7733	0.0031	1.5152	6.7375 e-4
C-K-means	**1.8906**	**0.0034**	**1.6713**	**0.0085**	**1.2744**	**0.0076**

Table 8: Accuracy comparison (Cloud).

Cloud	$k = 10$		$k = 20$		$k = 50$	
	DBI	DVI	DBI	DVI	DBI	DVI
K-means	1.1736	0.0207	1.23	0.02	1.3303	0.0186
K-means++	1.1644	0.0258	1.1946	0.0288	1.1973	0.0325
K-means||	1.1474	0.0233	1.1888	0.029	1.2163	0.0386
C-K-means	**0.9863**	**0.0369**	**0.9592**	**0.0484**	**1.1637**	**0.0582**

Table 9: Median cost (over 10 runs) on the Gauss dataset.

Gauss	$k = 20$		$k = 40$		$k = 50$	
	Seed	Final	Seed	Final	Seed	Final
K-means	—	0.0108	—	0.007	—	0.006
K-means++	0.0124	0.0107	0.0082	0.007	0.0071	0.006
K-means||	0.0118	0.0107	0.0078	0.007	0.0067	0.006
C-K-means	**0.0119**	**0.0108**	**0.0076**	**0.007**	**0.0067**	**0.006**

Table 10: Median cost (over 10 runs) on the SPAM dataset.

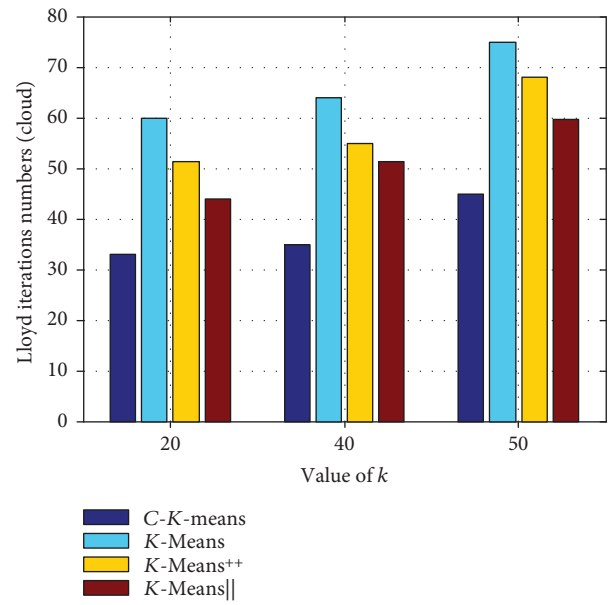
SPAM	$k = 20$		$k = 40$		$k = 50$	
	Seed	Final	Seed	Final	Seed	Final
K-means	—	0.1036	—	0.0771	—	0.071
K-means++	0.1136	0.0987	0.0886	0.076	0.08	0.0688
K-means||	0.1098	0.0968	0.0846	0.0752	0.0765	0.0692
C-K-means	**0.1022**	**0.0939**	**0.0861**	**0.0744**	**0.0788**	**0.0692**

Table 11: Times (in minutes) for SPAM.

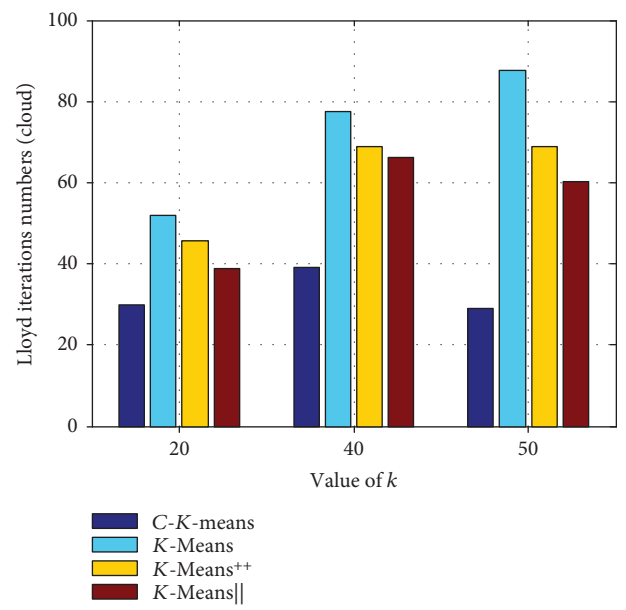
SPAM	$k = 100$	$k = 200$	$k = 500$
K-means	24.12	77.66	605.19
K-means++	31.66	63.64	153.00
K-means||	28.11	41.96	94.98
C-K-means	**16.38**	**24.78**	**99.42**

Table 12: Numbers of Lloyd's iterations until convergence (averaged over 10 runs) for the Cloud dataset.

Cloud	$k = 10$	$k = 20$	$k = 50$
K-means	49	25.2	24.2
K-means++	30.4	23.2	19
K-means||	28.2	21.8	18.6
C-K-means	**13**	**16**	**12**



(a) Gauss



(b) SPAM

Figure 4: Numbers of Lloyd's iterations until convergence (averaged over 10 runs).

convergence of Lloyd's iteration for datasets of several different dimensions with different initializations are shown in Figure 4(a) for the Gauss dataset (3 dimensions) and Figure 4(b) for the SPAM dataset (57 dimensions).

5. Conclusions and Future Work

This paper presents a covered K-means algorithm (C-K-means) that uses an improved covering algorithm (CA). First, based on the similarity between the data, the C-K-means algorithm uses the CA initialization to determine the number of clusters k and the specific cluster centers through self-organization. Because it is independent of the initial cluster centers, the CA is characterized as being "blind" without the need to have k prespecified. The K-means algorithm is then used to perform Lloyd's iteration on the k initial cluster centers determined by the CA until the cluster centers do not change, which means that the C-K-means clustering is complete, and the clustering results are close to optimal. In addition, a parallel implementation of C-K-means is performed on the Spark platform. Parallel computing is used to solve a large-scale data clustering problem and improve the efficiency of the C-K-means algorithm. A large number of experiments on real large-scale datasets demonstrated that the C-K-means algorithm significantly outperforms its counterparts under both sequential and parallel conditions. In future, we will optimize C-K-means and focus on the parameters that increase its speed and parallelism.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Technology R&D Program (no. 2015BAK24B01), the Natural Science Foundation of Anhui Province of China (no. 1808085MF197), and a Key Project of Nature Science Research for Universities of Anhui Province of China (no. KJ2016A038).

References

[1] T. S. Madhulatha, "An overview on clustering methods," *IOSR Journal of Engineering*, vol. 2, no. 4, pp. 719–725, 2012.

[2] M. Shindler, A. Wong, and A. W. Meyerson, "Fast and accurate k-means for large datasets," in *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*, pp. 2375–2383, Granada, Spain, 2011.

[3] M. Hajjar, G. Aldabbagh, N. Dimitriou, and M. Z. Win, "Hybrid clustering scheme for relaying in multi-cell LTE high user density networks," *IEEE Access*, vol. 5, pp. 4431–4438, 2017.

[4] Q. Chen, X. Zhang, Y. Wan, J. Zobel, and K. Verspoor, "Sequence clustering methods and completeness of biological database search," in *Proceedings of the Workshop on Advances in Bioinformatics and Artificial Intelligence: Bridging the Gap*, pp. 8–14, Melbourne, VIC, Australia, 2017.

[5] Q. Chen, Y. Wan, X. Zhang, Y. Lei, J. Zobel, and K. Verspoor, "Comparative analysis of sequence clustering methods for deduplication of biological databases," *Journal of Data and Information Quality*, vol. 9, no. 3, pp. 1–27, 2018.

[6] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," in *Proceedings of the 7th International Conference on Neural Information Processing Systems (NIPS'94)*, pp. 585–592, Denver, CO, USA, 1995.

[7] B. Castellani, R. Rajaram, J. Gunn, and F. Griffiths, "Cases, clusters, densities: modeling the nonlinear dynamics of complex health trajectories," *Complexity*, vol. 21, Supplement 1, p. 180, 2016.

[8] W. Zhao, H. Ma, and Q. He, "Parallel K-means clustering based on MapReduce," in *Cloud Computing*, vol. 5931 of Lecture Notes in Computer Science, pp. 674–679, Springer, 2009.

[9] Z. Tang, K. Liu, J. Xiao, L. Yang, and Z. Xiao, "A parallel K-means clustering algorithm based on redundance elimination and extreme points optimization employing MapReduce," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 20, 2017.

[10] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data K-means clustering using MapReduce," *The Journal of Supercomputing*, vol. 70, no. 3, pp. 1249–1259, 2014.

[11] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy, "The effectiveness of Lloyd-type methods for the k-means problem," *Journal of the ACM*, vol. 59, no. 6, pp. 1–22, 2012.

[12] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07)*, pp. 1027–1035, New Orleans, LA, USA, 2007.

[13] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[14] C. Ordonez, "Programming the K-means clustering algorithm in SQL," in *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '04*, pp. 823–828, Seattle, WA, USA, 2004.

[15] C. Ordonez and E. Omiecinski, "Efficient disk-based k-means clustering for relational databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, pp. 909–921, 2004.

[16] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-Scale Parallel Data Mining*, vol. 1759 of Lecture Notes in Computer Science, pp. 245–260, Springer, Berlin, Heidelberg, 2002.

[17] M. F. Jiang, S. S. Tseng, and C. M. Su, "Two-phase clustering process for outliers detection," *Pattern Recognition Letters*, vol. 22, no. 6-7, pp. 691–700, 2001.

[18] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley, "Fast distributed k-center clustering with outliers on massive data," in *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, pp. 1063–1071, Montreal, QC, Canada, 2015.

[19] D. Wei, "A constant-factor bi-criteria approximation guarantee for k-means++," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, pp. 604–612, Barcelona, Spain, 2016.

[20] J. Newling and F. Fleuret, "K-medoids for k-means seeding," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS'17)*, pp. 5201–5209, Long Beach, CA, USA, 2017.

[21] L. Zhang and B. Zhang, "A geometrical representation of McCulloch-Pitts neural model and its applications," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 925–929, 1999.

[22] L. A. Zadeh, "Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems," *Soft Computing*, vol. 2, no. 1, pp. 23–25, 1998.

[23] L. Zhang and B. Zhang, "The quotient space theory of problem solving," *Fundamenta Informaticae*, vol. 59, no. 2-3, pp. 287–298, 2004.

[24] A. Roy and S. Pokutta, "Hierarchical clustering via spreading metrics," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, pp. 2316–2324, Barcelona, Spain, 2016.

[25] R. Greenlaw and S. Kantabutra, "On the parallel complexity of hierarchical clustering and CC-complete problems," *Complexity*, vol. 14, no. 2, p. 28, 2008.

[26] Y. Yao, "A triarchic theory of granular computing," *Granular Computing*, vol. 1, no. 2, pp. 145–157, 2016.

[27] A. G. Shoro and T. R. Soomro, "Big data analysis: Apache Spark perspective," *Global Journal of Computer Science and Technology*, vol. 15, no. 1, pp. 9–14, 2015.

[28] A. Vattani, "*k*-means requires exponentially many iterations even in the plane," *Discrete & Computational Geometry*, vol. 45, no. 4, pp. 596–616, 2011.

[29] S. Peng, J. Sankaranarayanan, and H. Samet, "SPDO: high-throughput road distance computations on Spark using distance oracles," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 1239–1250, Helsinki, Finland, 2016.

[30] A. Asuncion and D. Newman, "UCI machine learning repository," 2007, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[31] P. Zhong and M. Fukushima, "Regularized nonsmooth Newton method for multi-class support vector machines," *Optimization Methods and Software*, vol. 22, no. 1, pp. 225–236, 2007.

[32] B. Jia, B. Yu, Q. Wu et al., "Hybrid local diffusion maps and improved cuckoo search algorithm for multiclass dataset analysis," *Neurocomputing*, vol. 189, pp. 106–116, 2016.

[33] Y. Wang, X. Duan, X. Liu, C. Wang, and Z. Li, "A spectral clustering method with semantic interpretation based on axiomatic fuzzy set theory," *Applied Soft Computing*, vol. 64, pp. 59–74, 2018.

[34] N. Nouaouria and M. Boukadoum, "Improved global-best particle swarm optimization algorithm with mixed-attribute data classification capability," *Applied Soft Computing*, vol. 21, pp. 554–567, 2014.

[35] N. Nouaouria, M. Boukadoum, and R. Proulx, "Particle swarm classification: a survey and positioning," *Pattern Recognition*, vol. 46, no. 7, pp. 2028–2044, 2013.

[36] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[37] X. Ru, Z. Liu, Z. Huang, and W. Jiang, "Class discovery based on k-means clustering and perturbation analysis," in *2015 8th International Congress on Image and Signal Processing (CISP)*, pp. 1236–1240, Shenyang, China, 2015.

[38] B. Wang, J. Yin, Q. Hua, Z. Wu, and J. Cao, "Parallelizing k-means-based clustering on Spark," in *2016 International Conference on Advanced Cloud and Big Data (CBD)*, pp. 31–36, Chengdu, China, 2016.

[39] D. Lai and C. Nardini, "A corrected normalized mutual information for performance evaluation of community detection," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, no. 9, 2016.

[40] A. Amelio and C. Pizzuti, "Correction for closeness: adjusting normalized mutual information measure for clustering comparison," *Computational Intelligence*, vol. 33, no. 3, pp. 579–601, 2017.

[41] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[42] M. Hassani and T. Seidl, "Internal clustering evaluation of data streams," in *Trends and Applications in Knowledge Discovery and Data Mining*, vol. 9441 of Lecture Notes in Computer Science, pp. 198–209, Springer, 2015.

[43] B. K. Mishra, A. Rath, N. R. Nayak, and S. Swain, "Far efficient k-means clustering algorithm," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI '12*, pp. 106–110, Chennai, India, 2012.