Received: 24 May 2021 | Revised: 22 November 2021 | Accepted: 1 December 2021

DOI: 10.1002/spe.3060

RESEARCH ARTICLE

WILEY

DACA: Distributed adaptive grid decision graph based clustering algorithm

Jing He | **Jun Zhou** | **Haoyu Wang** | **Li Cai**

National Pilot School of Software, Yunnan University, Kunming, China

Correspondence
Li Cai, National Pilot School of Software, Yunnan University, Kunming, China.
Email: caili@ynu.edu.cn

Present address
School of Software, Yunnan University, Kunming, China

Abstract

Clustering algorithms play a very important role in machine learning. With the development of big-data artificial intelligence, distributed parallel algorithms have become an important research field. To reduce the computational complexity and running time of large-scale datasets in the clustering process, this study proposes a distributed clustering algorithm DACA (distributed adaptive grid decision graph based clustering algorithm). In a distributed environment, DACA uses relative entropy to adaptively mesh the data to form an obvious sparse grid and dense grid. Then, the decision graph is used to determine the cluster center mesh object. Finally, the KD-tree is used to accelerate the determination of the cluster center of sparse points to complete clustering. The algorithm is implemented using the popular Apache Spark computing framework, compared with other distributed clustering algorithms, DACA can adaptively divide the grid according to the data distribution to obtain better clustering effect. At the same time, KD tree algorithm is used to speed up the decision-making of clustering center. Numerous experiments show that the DACA algorithm has excellent performance and accuracy on six standard datasets and real GPS trajectory datasets.

KEYWORDS

adaptive grid division, clustering algorithms, decision graphs, distributed, KD-tree

1 | INTRODUCTION

Recently, with the rapid development of the internet, the amount of data generated each year has been exponentially increasing. The global data volume could reach 1.8 ZB in 2 days (one of Zettabyte is 2^{70} byte)[1] in 2014. By 2021, the growth of data generation will become even higher. Thus, the conventional single application-driven internet services have to be converted into data- and application-driven internet services. Therefore, big-data artificial intelligence and machine learning have been rapidly developing, the most obvious of which is the research on big data clustering algorithms, which is often discussed in the fields of internet business and academic research.[2]

Clustering algorithms, which are a type of important algorithms used for data mining, have been widely employed for data analysis, image processing, market research, user segmentation, web document classification, and other fields. At present, clustering algorithms can be roughly divided into three types—traditional clustering, intelligent

clustering, and big data clustering. Traditional clustering and intelligent clustering can be classified into small data clustering.[3]

Traditional clustering generally comprises partition-based clustering and hierarchical clustering, among which the main representatives, K-means,[4] k-medoids,[5] CURE,[6] and BIRCH,[7] are used to segment data in different ways and evaluate data point categories by Euclidean distance and similarity. However, these algorithms often have limitations, such as limitations in terms of clustering results, depending on the selection of initial clustering centers and sensitive outlier sample points.

Intelligent clustering is also a hot research topic at present. It generally refers to artificial neural network clustering, kernel clustering, and other intelligent methods of clustering. These are competitive learning methods, in which numerous neurons (or processing units) are interconnected to form a network, and feature learning is performed through forward-propagation and back-propagation to complete clustering. Among them, the main representatives are S^2 ConvSCN,[8] ClusterGAN,[9] and kernel k-means clustering.[10] They use artificial neural networks to cluster and corresponding heuristic algorithms to obtain high-quality clustering results. However, the computational complexity is often high, and the results depend on the selection of some empirical parameters.

Big-data clustering often refers to distributed clustering, parallel clustering, and high-dimensional clustering. Its core principle is to deal with the relationship between computational complexity and cost, scalability, and speed.[3] Big-data clustering focuses on minimizing the cost of clustering quality and improving the scalability and execution speed of the algorithm. Among these methods, PK-Means,[11] DBSCAN-PSM,[12] and DENCAST[13] have been proposed recently. They often split and parallelly calculate a clustering task through multiple computers at the same time to speed up the operation. Compared with the operation cost of a single machine, big data clustering can effectively reduce the time complexity of clustering and improve algorithm scalability. This study is a clustering research in the distributed environment of big data.

The density peak clustering algorithm is among many clustering algorithms that suffers from high space-time complexity and cannot effectively cluster large datasets.[14] The grid method can reduce the algorithm space complexity and algorithm computation, and then reduce the space-time complexity of the density peak clustering algorithm. Relative entropy can measure the distance between two random distributions. When two random distributions are the same, their relative entropy is zero. When the difference between two random distributions increases, their relative entropy will also increase.[15] Therefore, in clustering, we can use relative entropy to determine the grid, so we can well distinguish between dense grid and sparse grid. Based on the idea of decision graph, clustering does not require specifying the number of parameter in advance; only one the number of cluster needs to be set in advance, and the sensitivity is low; thus outliers can be easily determined to complete the clustering.

Recently, most research works show that the computational load of the clustering process can be effectively reduced by using distributed grid partition.[14] Therefore, we propose a distributed clustering algorithm called DACA (distributed adaptive grid decision graph based clustering algorithm). In DACA, relative entropy is used[16] to adaptively partition the data space according to the actual distribution of data, which result in notable sparse and dense meshes. The grid instead of single data point is used as the clustering object to reduce the computing complexity of the distance computation between points. Then, we adopt the idea of the decision graph to determine the clustering center of grid objects and the adjacent grid is searched and merged using a KD-tree[17] to complete clustering. Numerous experiments show that the algorithm not only shows excellent accuracy but also significantly reduces the running time of the algorithm.

We used the Apache Spark cluster to implement our algorithm, Hadoop's HDFS file system for corresponding data storage, and the spark context for data loading. In the implementation of the DACA algorithm, we used spark's RDD operator to perform operations such as the distribution and aggregation of the corresponding grid, thereby achieving the final DACA algorithm and achieving good results.

The main contributions of this study are as follows:

- A novel distributed clustering algorithm DACA based on adaptive grid division and decision graph is proposed, and its effectiveness and correctness were proved through numerous experiments.
- We proposed to use the decision-making idea to determine the grid cluster centers in a distributed environment. The cluster grid centers of the decision grid were determined on different machines, and then, the cluster centers of the entire data distribution were merged and calculated.
- In this study, we used spark to implement the distributed form of the KD-tree and used a KD-tree to query the neighbor nodes of the grid data to optimize the detection of discrete points in the clustering process;

experiments show that this method is better than clustering without KD tree, and the process showed high accuracy and efficiency.

- We conduct sufficient experiments to show the advantage of DACA. The experimental results show that DACA algorithm has better performance and accuracy than parallel optics, spark DBSCAN and other distributed algorithms in the distributed environment of large-scale datasets.

2 | RELATED WORK

This section mainly describes the content related to distributed parallel clustering and spark parallel operation.

2.1 | Distributed parallel clustering

At present, the K-means algorithm is the most traditional employed clustering algorithm. However, with the popularization of big data cloud computing, traditional clustering is apparently insufficient in terms of the execution time and performance on large datasets. Since Oliva et al.[4] proposed the distributed K-means algorithm in 2013, various distributed algorithms have been developed and applied. K-means is also well supported in the Mlib machine learning library of Apache Spark.[18] For example, DBSCAN,[19] OPTICS,[20,21] and other algorithms perform clustering based on the dense density of datasets in spatial distribution, wherein the number of clusters need not be set in advance; thus, they are particularly suitable for clustering datasets with unknown content. In the context of big data, the optimization and innovation of these algorithms are still very important research prospects.[18]

In the field of distributed clustering research, Patwary et al. proposed a distributed parallel algorithm called POPTICS[21] based on the core idea of OPTICS in 2013. To solve the problem of strong sequential data access order in the clustering of the OPTICS algorithm, they proposed a scalable parallel optics algorithm called POPTICS. This algorithm uses the minimum spanning tree algorithm of options and prim and shows a certain parallel advantage in regular density datasets. However, in each clustering process, the caching of intermediate data incurs a significant network overhead, which increases the computing cost. MR-DBSCAN[22] proposed by Yaobin He et al. in 2014, which is an extensible DBSCAN algorithm using MapReduce. All the key sub-processes of the algorithm were completely parallel, and the data segmentation method based on calculation cost estimation was used to segment and cluster data. However, because of the performance bottleneck of MapReduce, it is insufficient in performing real-time tasks.

In 2018, Ameya Malondkar and others proposed[23] spark GHSOM algorithm based on the traditional GHSOM algorithm designed to only deal with datasets with digital attributes. This algorithm can not only deal with mixed attribute datasets, but also has strong scalability and description ability, the accuracy of deeper feature fusion needs to be further optimized. However, there is some complexity in the construction of feature network, which is difficult to implement. DBSCAN-PSM was proposed by Chen et al. in 2019.[12] The partition method of DBSCAN is very simple, and the steps of the GETNEIGHBORS query repeatedly visit the datasets on spark. A new DBSCAN-PSM, which adopts a new method of data segmentation and merging, is proposed to reduce the number of visits to the dataset and the impact of IO on the algorithm. This method can improve the parallel efficiency and clustering algorithm performance to a certain extent. However, after the KD-tree is created, it performs poorly on large-scale data when combined with the GETNEIGHBORS query. The DENCAST algorithm, which was proposed by Corizzo et al. in 2019,[13] is a density-based distributed clustering algorithm, which uses the identified clusters to solve the problems of a single target and multi-objective (thus solving complex tasks such as time series prediction). However, the parameter setting of the algorithm in multi-objective is complex, and it is difficult to perform general operations. A parallel label diffusion and label selection (PLDLS[24]) community detection algorithm based on spark is proposed by Roghani et al., a new and fast candidate-based merge method is utilized to obtain dense communities. The proposed method is implemented in spark and GraphX and is scalable for big networks.

To compensate for the shortcomings of traditional clustering algorithms in the big data environment, we propose a parallel algorithm DACA based on adaptive grid division and decision graph density clustering, which uses Apache Spark Framework 2.4.3[18] to implement the corresponding parallel algorithm. At the same time, experiments

were performed on several standard datasets and real GPS datasets. DACA shows excellent performance and good scalability.

2.2 | RDD operator of Apache Spark

Apache Spark is a fast-general cluster-computing framework. It is based on a distributed memory computing method that saves the intermediate results of each job in memory, thus greatly reducing the large number of read and write operations on the disk.[18] Spark introduces the concept of an elastic distributed dataset to realize task scheduling, distribution, and processing, and provides many computing mode components, such as spark SQL, spark streaming, MLIB, and GraphX, which can be applied to distributed platform scenarios. It has made many attempts and researches in big data processing and algorithm research, such as stack community detection,[25] link prediction,[26] text classification and so on.[17]

In spark, RDD is the core concept. Using RDD, we can process different businesses in a basically consistent manner. The essence of RDD is a set of immutable read-only distributed elements. Each RDD contains different partitions, which are multiple dataset fragments, run on different cluster nodes, and can be processed simultaneously. In fact, the calculation process of the spark parallel framework is to create RDD through pending data, convert it into a new RDD, and evaluate the action operation of calling RDD to obtain the result.[18] RDD supports two types of operations—transformation and action. The transformation operation converts the existing RDD into a brand-new RDD. The transformation operation in spark is a lazy evaluation and performs calculation only when the action operation uses these RDDS. The action operation triggers the actual calculation and returns the result to the driver or store the result in the external storage system.

In this study, we make full use of the advantages of spark RRD in the implementation of the DACA algorithm. For example, the map operator is used to segment and distribute the data. The reduce operator is used to merge and calculate each cluster center point, and the reduceByKey operator is used to count and analyze the grid index and other information. Experiments show that the performance of the DACA algorithm has been completely demonstrated in the distributed computing of spark.

3 | IMPLEMENTATION OF DACA

In this section, we first introduce the overall clustering process of the DACA algorithm and then describe the distributed adaptive grid division method in detail. Finally, the sparse grid determination based on the KD-tree and KNN algorithm in the spark-distributed environment is expatiate.

In spark, we implement the corresponding algorithm through the corresponding distributed operation RDD. The most commonly used map operation and reduce operation can well describe the core process of the whole algorithm execution. In Figure 1, we can use the map operation for cluster center grid and cluster center grid class label The reducebykey operator can be used in merging and final result collection.

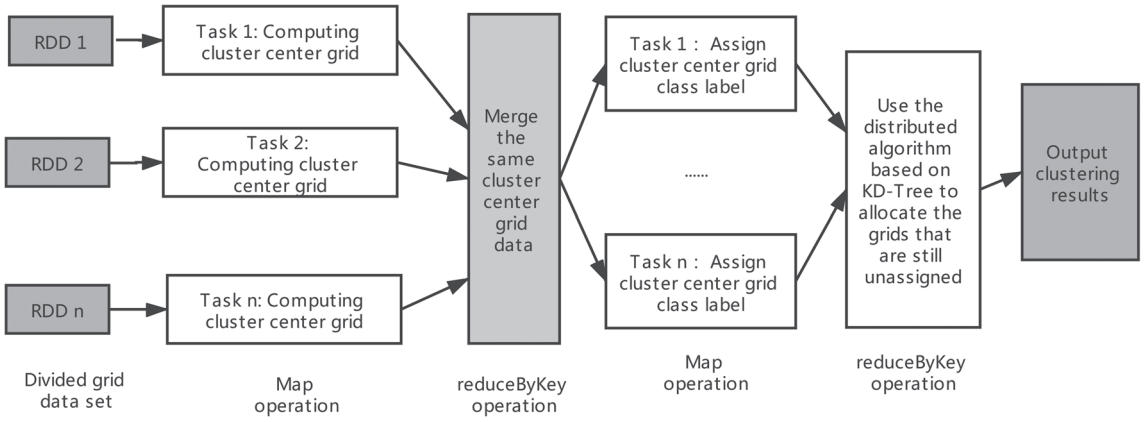

FIGURE 1 Flow chart of decision cluster center grid clustering process under distributed environment

3.1 | Overview of DACA algorithm

In DACA, the clustering process can be mainly divided into three steps. First, the massive data is loaded into the spark system, and the data is partitioned into adaptive grids with different sizes according to the relative entropy. Then, the cluster centers of grids are computed by the decision graph. Especially, the sparse grid with data point density less than the set threshold needs further processing. Thus, the KD-tree and KNN algorithm are used to rebuilt the cluster for sparse points. The overall process of DACA is shown in Figure 2.

A. Relative entropy adaptive mesh generation

To better distinguish the sparse and dense regions of the whole data set, the relative entropy is used to construct adaptive mesh. Initially, the data set is divided into equal length mesh with the size of the grid ε, and a histogram of the ith dimension is constructed. Then, the relative entropy of each histogram θ_i is calculated by sequentially scanning the relative entropy of all grids θ_k. If $\theta_i \geq \theta_k$, the algorithm continue to scan the next grid. If $\theta_i < \theta_k$, the previous grid is merged and the scanning continues until the multidimensional grid scanning is completed. So the relative entropy of histogram in the ith dimension can be expressed as Definition 1.

Definition 1. The relative entropy of histogram in the i th dimension can be expressed by the following formula:

$$H_{(X_{ij})} = -\frac{\sum_{x_{ij} \in X_i} d(x_{ij}) log_2(x_{ij})}{\sum_{T}^{T} \frac{1}{T} log_2 T} \tag{1}$$

Here, x_{ij} is the jth square on the ith dimension, and $d_{(x_{ij})}$ is the proportion of the number of data in the square to the whole data set. T is the number of square lattices in the ith dimension.

B. Decision graph and cluster center grid determination

In the previous step, we divided n k-dimensional non-overlapping grids according to the characteristics of the dataset. In this section, we first delete the grid with 0 data points and generate a new non-empty grid, denoted as N_P, and then determine the set of mesh objects N_P and the number of mesh objects, N_G. Next, we calculate the density of each grid. We take the nearest distance between $Cell_i$ and $Cell_j$ as the distance value d_{ij} of the grid object. Finally, according to the calculation of the cluster center position of the most clustered grid objects with higher density and larger distance value, the distance value d_{ij} is calculated as follows:

$$d_{ij} = \sum_{j:\rho_j > \rho_i} \sqrt{\sum_{i,j \in Cell} (Cell_i - Cell_j)^2} \tag{2}$$

Here, $Cell_i$, $Cell_j$ represent the center points of the ith and jth grids respectively, ρ_i and ρ_j represent the density values corresponding to the ith and jth grid, respectively.

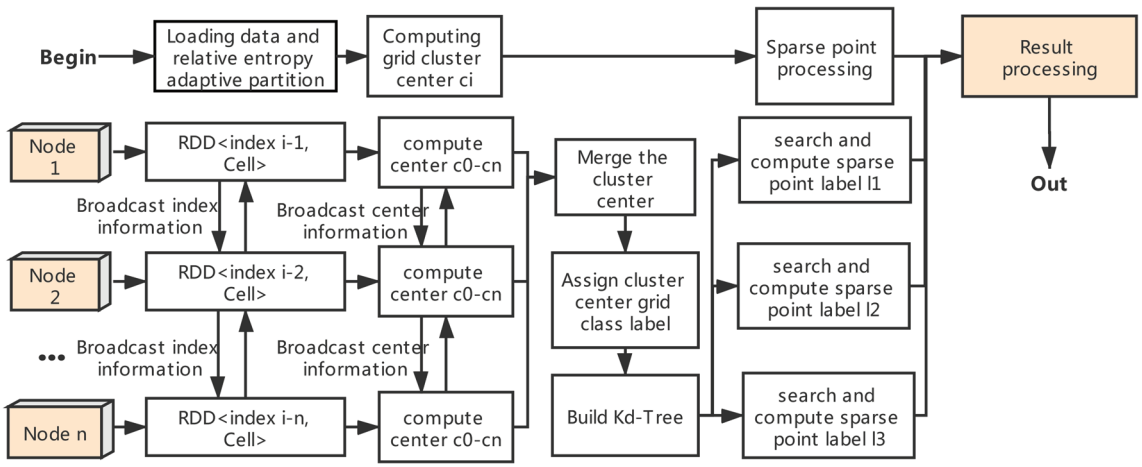

FIGURE 2 Overall process of DACA clustering

C. Clustering process

After the cluster core grid object is determined, we use a density-based division of the method to complete clustering. First, we need to assign a different class label to the data in each cluster center grid. Then, we scan the adjacent grids, aggregate grid object data with similar density values of adjacent grids, and redistribute class labels until all grids are allocated. Finally, to detect the grid containing discrete points, we use the KD-tree to find the K grids closest to it, assign the class label of the closest and densest grid object to the discrete grid, and then, merge the corresponding data into the category grid. Once all data grid category labels are allocated, the entire clustering process is completed.

3.2 | Distributed adaptive meshing

In the big data era, the computing overhead of massive data become one of the most factors restricting the development of cluster algorithm. In DACA, we cluster data grids instead of data points, which greatly reduce the amount of data. Relative entropy can reflect the degree to which the distribution of data points tends to be uniform. The more uniform the data is, the greater the relative entropy value is. Correspondingly, the more the data distribution tends to be concentrated, the smaller the relative entropy value is.[16]

In spark, each data node independently meshes the local data set and merges the sparse grid according to the relative entropy value to obtain the local adaptive data grid. Then, each local grid is merged to obtain the adaptive grid division of the whole data. Through parallel computing, the computational efficiency of grid division is effectively improved.

Algorithm 1 describes the concrete steps of distributed adaptive meshing generation. Under spark parallelization schema, the pre-processed dataset is uploaded to the distributed HDFS (Hadoop Distributed File System) environment first. SparkContext read data information from the HDFS cluster and converts it into the actual operational data RDD. Then, spark RDD's operators include map, reduce, and reduceByKey are invoked to initialize the grid and generate a new RDD data form of <grid ID, feature, label>. Here, cell represents the relevant information of a data point, grid ID, feature and class label, RDD<grid ID, feature, label> is the RDD<Cell> described in the following algorithm. In this way, we can calculate the entropy of each dimension in the current RDD grid data and output the RDD<Cell>. Finally, the RDD data are merged and distinguished according to the entropy value. The RDD form of cell is then output. To facilitate the clustering operation later, the data points are converted to data grids, which maintain the actual data distribution.

3.3 | Clustering based on distributed decision graph

Decision graph clustering algorithm (DG algorithm) obtains clustering center based on local density. That means, the local density of cluster center points is higher than that of its neighbor nodes, and the distance between different cluster center points is longer. DG algorithm has many excellent properties. (1) The algorithm does not need to specify the number of clusters in advance; (2) the cluster with non-spherical shape can be obtained use DG algorithm; (3) the complexity of the algorithm is lower than that of the general K-means algorithm. In DACA algorithm, we adopts distributed decision graph to determine the cluster center grids.

Figure 3 illustrated the process of distributed decision graph cluster. The grids that have been initialized in Section 3.1 are loaded into each data node and converted to the format of RDD<Cell>. Then, the local cluster center grids in each data node are found through parallel map operation. In global layer, the cluster center grids are refined. On the one hand, the grids with the same cluster center are merged into the same cluster. On the other hand, the outliers with lower local density values on local data nodes need to be recalculated to determine whether they will form new cluster centers in the global layer. Finally, each grid within a cluster are marked with the center grid's label. A grid that does not belong to any cluster is identified as noise data. It's cluster label is denote as 0.

Definition 2. the local density of a grid. For a grid $cell_i$, its local density ρ_{Cell_i} can be defined as follows:

$$\rho_{Cell_i} = \sum_j count(d_{Cell_{ij}}, d_c) \tag{3}$$

Here, $d_{Cell_{ij}}$ represent the distance from the grid center of $Cell_i$ to the grid center of $Cell_j$, d_c represent the distance across domain central grids. $count(d_{Cell_{ij}}, d_c)$ represent the density of points on grid i.

Algorithm 1. Distributed adaptive meshing

Require:
The input path of the data set: *inputPath*;
Initial step size of the grid: *gL*;
The number of adjacent points: *neigPointNum*;
The relative entropy threshold ratio parameter: *rate*;
Ensure: Grid RDD operator containing data after initialization: RDD<Cell>

```
function ENTROPY(dataSet)
    result : Double ← −1.0
    feature : List[Double] ← dataSet.map(line = >line.getFeature())
    if feature.nomEmpyty then
        result ← 0
    end if
    for x ← feature do
        if x! = 0 then
            result ← result + (−x) × log2x
        end if
    end for
    return result
end function
dataSet ← sc.textFile(inputPath)
dataSet: Convert the read data format RDD<index, feature, label>.
dataSet ← ENTROPY(dataSet)
dataArea: Combine grid data information based on relative entropy, gL, ratio and neigPointNum
dataRes :RDD<Cell> ← Convert dataArea to Cell object
return dataRes
```

Through formula 3, we know that the larger ρ_{Cell_i}, the greater the local density of the grid represented by ρ_{Cell_i}, and the more likely it is to become the cluster center. In the selection of cluster center distance σ_i, we first need to find the grid with the largest density, then arrange the density from small to large, and then calculate the cluster center distance of other grids. In the set with density greater than the grid point, the distance from the grid center distance is the smallest.

Definition 3. The distance between dense grids.

$$\sigma_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{4}$$

Here, the d_{ij} represent the distance between $Cell_i$ and $Cell_j$, and it is calculated as formula 1. Here, when the local density of the grid is maximum, the $\sigma_i = max_j(d_{ij})$.

In Section 3.1, according to the relevant characteristics of the dataset, we use the relative entropy to divide the initial grid and convert the original dataset into the grid dataset of RDD<Cell> corresponding to the Apache Spark operation. The next step involves determining the cluster center grid.

First, we calculate the cluster center of the corresponding grid according to the grid data information read by each machine and map it to the RDD operator of <cluster center, grid object data set> using the map operator operation. For discrete points, that is, the grid object with a grid density of less than 2, we initialize the cluster center label as 0. Here, we need to use the broadcast mechanism of Apache Spark to broadcast the calculation results of each machine, so that the information can be communicated and transmitted on each machine to ensure global correctness of each node's computing cluster center.

Then, we use the RDD operator of the reduceByKey to merge the RDD of the <cluster center, grid object dataset> on each node according to the cluster center as the key and assign the class label to the grid object corresponding to the data point. Then, we determine the label of sparse meshes (meshes with density less than 2).

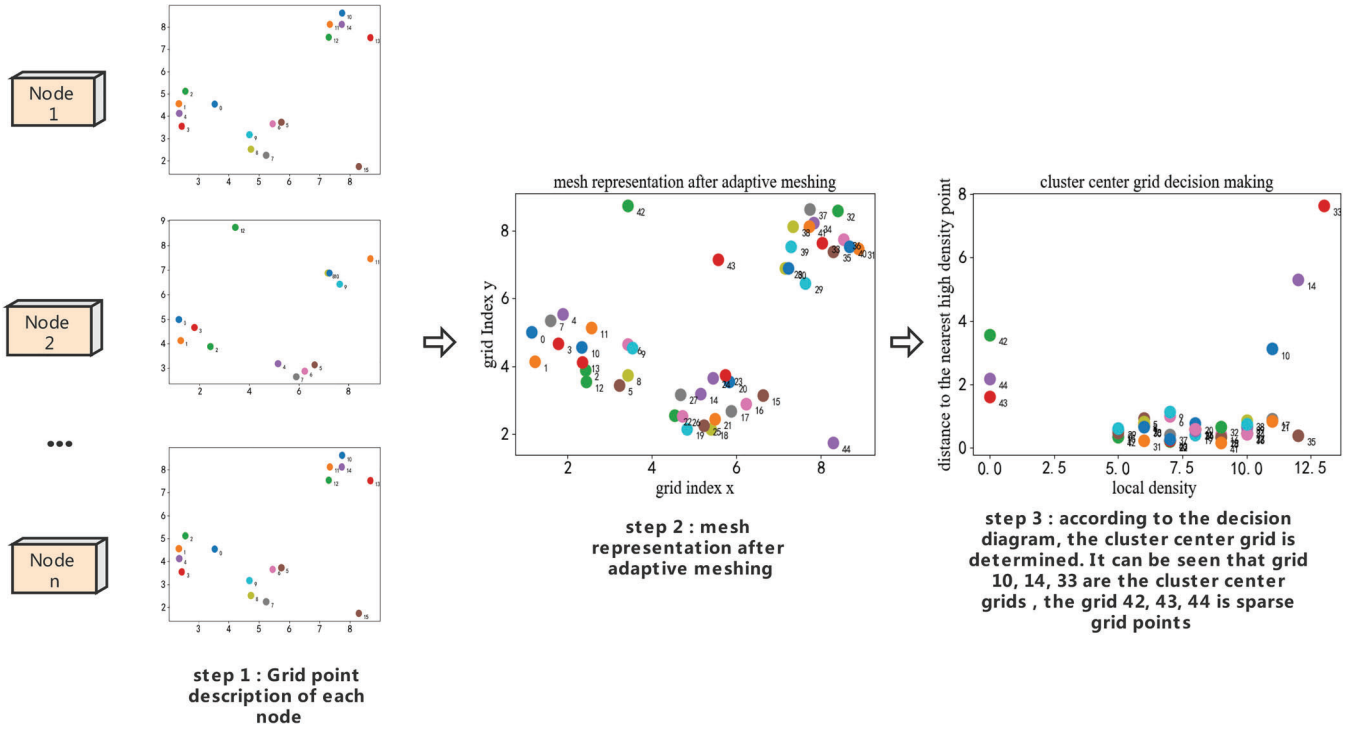

FIGURE 3 The process of distributed decision graph cluster

To determine the label of the data points in the sparse grid, we use the KNN algorithm based on the KD-tree to query the neighbor grid of the data points, calculate the weight of the category of the neighbor grid, obtain the grid category with the largest weight of the sparse grid point, and assign the label of the neighbor grid to the corresponding sparse grid. In this way, all the grid data are given corresponding category labels. Finally, we can save the final results using the relevant API interface of the Apache Spark. The clustering process is shown in Algorithm 2. The search process of the KNN algorithm based on KD-tree is explained in Section 3.3.

3.4 | Implementation of KNN algorithm based On KD-tree in spark environment

To find and determine the sparse grid data category in Section 3.2, we need to implement the KNN algorithm based on the KD-tree in the Apache Spark environment according to the existing grid data information to ensure the correctness and efficiency of our algorithm.

We know that the k-nearest neighbor query algorithm is a commonly used algorithm for querying large-scale spatial data.[18] KD-tree is used to build an index of large-scale spatial data, and then, to hierarchically divide the search space and finally to k-nearest neighbor query, which can ensure the efficiency of the search. In the traditional machine learning algorithm, the performance and efficiency of large-scale spatial data queries are relatively weak.[27] The k-nearest neighbor query algorithm based on the KD-tree can effectively accelerate the query of large-scale data and the optimization of related algorithms.

In the Apache Spark environment, we first need to use spark RDD operators to construct the KD-tree and then apply it to the KNN algorithm. Therefore, we can use the parallel computing characteristics of spark to conduct distributed data queries. The main algorithm flow is shown in Figure 4. First, we have to select the value k with the largest variance from the existing k-dimensional dataset and select the median value m as the partition axis to divide the dataset to obtain two subsets. Simultaneously, a tree node is created for storage. Here, the broadcast form of spark is used for data communication of each node. Then, the above steps are repeated for the two subsets until all subsets cannot be partitioned again. If a subset cannot be partitioned again, the data in the subset will be saved to the leaf node. Thus, we create a KD-tree corresponding to the grid data. When we query the k-nearest neighbor query algorithm, we can use this KD-tree to query the corresponding points of the sparse grid.

Algorithm 2. Distributed decision graph and clustering process of cluster center network

Require:
Grid RDD operator divided according to relative entropy: *dataCell* : *RDD<Cell>*
The number of adjacent points: *neigPointNum*;
Ensure: RDD data after clustering: RDD<Cell>

function GETCENTERS(*points* : *List*[*Double*])
  $x : Double \leftarrow 0.0, y : Double \leftarrow 0.0$
  for $p \leftarrow points$ **do**
    $x = (x + p.x), y = (y + p.y)$
  end for
  $centerX = \frac{x}{p.size}$, $centerY = \frac{y}{p.size}$
  return $(centerX, centerY)$
end function
for $p \leftarrow dataCell$ **do**
  $centers \leftarrow$ GETCENTERS($p.points$)
  $p.centers = centers$
  p.cluster: Assign class labels according to the center point and density ρ
end for
$neighbors \leftarrow KNN(neigPointNum, dataCell)$
for $p \leftarrow dataCell$ **do**
  if $p.cluster == 0$ **then**
    p.cluster ← Calculate the most dense and closest grid class mark
  end if
end for
return *dataCell*: RDD data after clustering

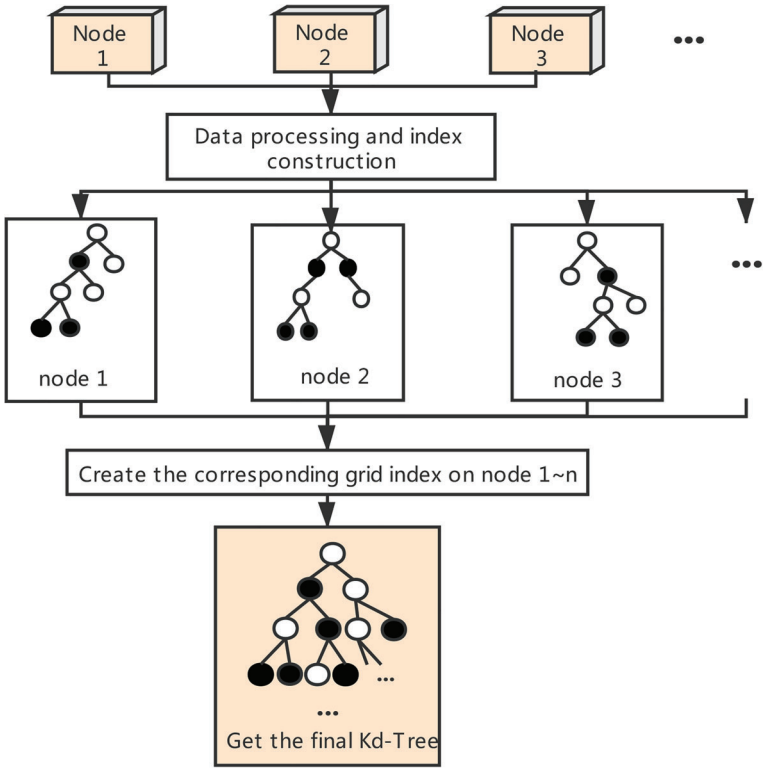

FIGURE 4 Process of searching sparse grid data points by the KNN algorithm based on the KD-tree in the Apache Spark-distributed environment

Algorithm 3. The DACA algorithm

Require:
The input path of the data set: *inputPath*;
Initial step size of the grid: *gL*;
The number of adjacent points: *neigPointNum*;
The relative entropy threshold ratio parameter: *rate*;
Output path of clustering results: *savePath*

1: **function** DACA(*inputPath* : *String*, *gL* : *Double*, *neigPointNum* : *Int*, *rate* : *Double*, *savePath* : *String*)
2: *sc* ← *SparkContext*()
3: *dataSet* : RDD<Cell>← *Algorithm*1(*inputPath*, *gL*, *rate*) Get the RDD based on the dataset after adaptive meshing.
4: *result* : RDD<Cell>← *Algorithm*2(*savePath*, *neigPointNum*) Get the results after clustering.
5: *sc*.*saveAsTextFile*(*savePath*): Save the clustering results in *savePath*
6: **end function**

3.5 | The DACA algorithm

Although we have described in detail the core clustering steps and related algorithm implementation of DACA algorithm in Algorithms 1 and 2, in Algorithm 1, we extract the relevant features of the original dataset and divide the grid features in the form of relative entropy. In Algorithm 2, we use decision graph clustering algorithm (DG algorithm) cluster centers are selected by distributed grid clustering, and KD tree is used to optimize the selection of cluster centers, so as to complete the whole clustering process. We can describe the overall process of DACA distributed grid clustering algorithm as Algorithm 3.

4 | EXPERIMENT

In this section, we describe our experiment. First, we introduce the environment and dataset of the experiment and then verify the accuracy of the algorithm. We compare the running time of the algorithm, the effectiveness of the KD-tree, and the test of parallel performance.

4.1 | Experimental environment and dataset

We performed corresponding experiments on a spark cluster with four nodes. Each node was allocated 4 cores, 6 GB running memory, and 50 GB storage space. We used the yarn of Apache Hadoop as our resource allocator, HDFS as our distributed storage system, and Apache Spark version 2.4.3. The contrast algorithms and experiments were based on the Scala language.

We conducted experiments based on the six standard data sets of aggregation, compound, spiral, flame,[28] 20newsgroups*, MNIST† and real taxi GPS trajectory datasets. The aggregation, compound, spiral, and flame four standard datasets were used to prove the effectiveness of our DACA algorithm. However, the number was small, and not enough to explain the application in a big data environment. We expanded it based on the original data distribution, and each dataset was expanded by 20 times to verify the effectiveness of the clustering algorithm in the big data environment. The clustering results of the four clusters are shown in Figure 5. The 20newsgroups dataset can show that DACA algorithm also has certain advantages in the clustering effect of text data. MNIST handwritten numeral dataset clusters pictures, and the results show that it also has good clustering effect on picture data. The detailed data description is shown in Table 1.

4.2 | Clustering accuracy

The clustering accuracy of the algorithm describes the quality of the algorithm.

*20newsgroups: http://qwone.com/ jason/20Newsgroups/
†MNIST: http://yann.lecun.com/exdb/mnist/

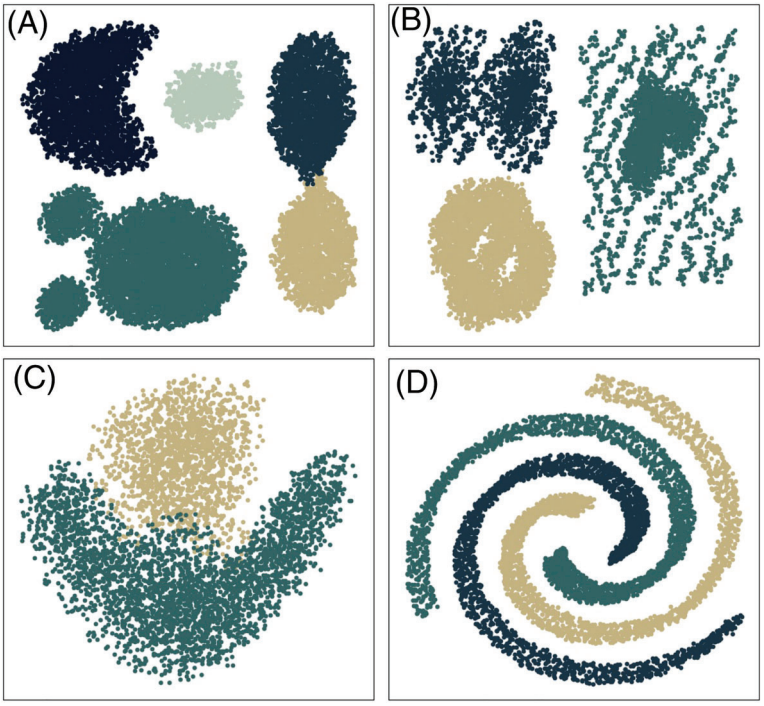

FIGURE 5 The clustering effect of DACA on aggregation, compound, spiral, and flame datasets in the Apache Spark-distributed environment. Each dataset is expanded by 20 times according to the original data distribution. (A) Aggregation dataset; (B) compound dataset; (C) flame dataset; (D) spiral dataset

TABLE 1 Classification of experimental dataset

Dataset	Number of records	Number of categories
Flame	4820	2
Compound	7980	5
Spiral	12,000	3
Aggregation	15,760	7
20newsgroups	18,846	20
MNIST	70,000	10
Taxi GPS track data	250,000	Unsupervised

4.2.1 | Evaluation method

Clustering is an unsupervised learning algorithm. Unlike classification, clustering is a statistical method for separating similar data from dissimilar data, and it often includes an optimization goal and a learning process. In our experiment, to determine the accuracy of clustering, we used three indicators for verification, Purity, RI, and F-measure indicators:

Purity method is a very simple clustering evaluation method, which only needs to calculate the proportion of the number of correct clusters in the total number of samples. For a cluster i, first calculate p_{ij}, p_{ij} represents the probability that the members in cluster i belong to class j, $p_{ij} = \frac{m_{ij}}{m_i}$, Of which $m_i$is the number of all members in cluster i, m_{ij} is the number of members belonging to class j in cluster i. We define the purity of cluster i as $p_i = \max(p_{ij})$, The purity of the whole cluster is

$$\text{Purity} = \sum_{i=1}^{K} \frac{m_i}{m} P_i \tag{5}$$

Here, K is the number of clusters and m is the number of members involved in the whole cluster division. The purity value range is 0–1. The closer the value is to 1, the better the clustering effect.

Rand index calculates the similarity between the predicted value and the real value of the sample. The RI value range is [0,1]. The larger the value, the more consistent the clustering results are with the real situation.

$$RI = \frac{TP + TN}{TP + FP + TN + FN} = \frac{a + b}{C_m^2} \tag{6}$$

where C represents the actual category information, K represents the clustering result, a represents the logarithm of elements in the same category in C and K, and b represents the logarithm of elements in different categories in C and K. An instance is a positive class and is predicted to be a positive class (TP). If the instance is a negative class and is predicted to be a positive class, it is called a false positive class (FP). If the instance is a negative class and is predicted to be a negative class, it is called a true negative class (TN). If a positive class is predicted to be a negative class, it is a false negative class (FN). Since each sample pair can only appear in one set, $TP + FP + TN + FN = m(m-1)/2$ represents the logarithm of samples that can be combined in the dataset.

The F-measure is the weighted harmonic average of precision and recall. However, precision (P) is defined as Equation (7). P represents the proportion of examples divided into positive examples that are actually positive examples.

$$P = \frac{TP}{TP + FP} \tag{7}$$

The recall rate (R) is a measure of coverage, and measure how many positive examples are divided into positive examples in all positive examples:

$$R = \frac{TP}{TP + FN} \tag{8}$$

Sometimes there are contradictions between P and R indicators. Thus, they need to be considered comprehensively. The most common method is F-measure (also known as F-score). It is a common evaluation standard in the field of information retrieval. The larger the value of the F-measure, the better the clustering effect. F-measure is the weighted harmonic average of precision and recall:

$$F = \frac{(a^2 + 1)P * R}{a^2(P + R)} \tag{9}$$

When parameter $a = 1$ is the most common F1, this is:

$$F = \frac{P * R}{P + R} \tag{10}$$

It can be seen that F1 combines the results of P and R. when F1 is high, it can show that the test method is more effective.

4.2.2 | Clustering result and analysis

In the previous section, we introduced the concepts of F-measure, purity, and RI.

For the standard dataset, because we know the actual sample label, to achieve good clustering effect, we need to specify the initial step size of the grid GL, the number of adjacent points neigPointNum, the number of adjacent grids neigGridNumand, and the ratio of relative entropy rate. After repeated experiments, the parameters GL, neigPointNum, and neigGridNumand rate of the aggregation dataset are 0.346, 2, 1, and 0.89, respectively, those of the compound dataset are 0.245, 1, 1, and 0.892, respectively, those of the flame dataset are 0.28, 2, 1, and 0.75, respectively, and those of the flame dataset are 0.4, 1, 3, and 1, respectively.

In the spark-distributed environment, we performed experiments on the parallel optics algorithm,[22] spark DBSCAN[28] algorithm, and DACA algorithm. The experimental results are shown in Table 2. The table shows that DACA has the

TABLE 2 F-measure, purity, and RI index values of DACA, parallel optics, spark DBSCAN on aggregation, compound, spiral, flame, 20newsgroups, MNIST data

Dataset	Algorithm	F-measure	Purity	RI
Aggregation	Parallel optics	0.7598	0.953	0.7038
	Spark dbscan	0.8487	0.7896	0.7892
	DACA	0.7976	0.9987	0.8788
Compound	Parallel optics	0.8024	0.7794	0.7107
	Spark dbscan	0.8609	0.8361	0.8995
	DACA	0.8636	1	0.9129
Spiral	Parallel optics	0.8567	0.8769	0.81
	Spark dbscan	0.8985	0.8558	0.832
	DACA	1	1	1
Flame	Parallel optics	0.6978	1	0.7358
	Spark dbscan	0.9916	0.9917	0.9834
	DACA	0.9276	0.9917	0.9443
20newsgroups	Parallel optics	0.8218	0.8200	0.8315
	Spark dbscan	0.8333	0.8710	0.8333
	DACA	0.8637	0.8600	0.8576
MNIST	Parallel optics	0.8321	0.8185	0.8418
	Spark dbscan	0.8712	0.8716	0.8716
	DACA	0.8921	0.8892	0.8933

best overall clustering effect; the parallel optics clustering effect is poor, which may be caused by the error of setting the minimum point and radius. On the dataset compound, spark can be observed. The clustering effect of DBSCAN is better than that of the DACA algorithm mainly because the distribution of clusters in the composite dataset is uneven, and DBSCAN distributes data points based on density and clusters data points with minimum radius to identify correct categories. Furthermore, DACA has some disadvantages in the recognition of certain dense categories. However, the overall clustering effect is better than that of DBSCAN.

Meanwhile, we can know from Table 2 that DACA also shows good results in the clustering effect of text data. From the table, on the data set 20newsgroups, not only DACA can adapt to the clustering task of text data, but also the overall effect of DACA has better clustering effect than parallel optics and spark DBSCAN. On the one hand, it mainly benefits from the relative entropy adaptive meshing of DACA, which can more finely distinguish the categories between data. On the other hand, it uses the clustering judgment of decision graph and KD tree to classify outliers.

Finally, we can see from Table 2 that the effect of DACA on MNIST, a dataset for handwritten numeral recognition, also has certain advantages. From the value of F-measure, we can see that DACA algorithm is equally effective compared with the other two algorithms. From the RI value, we can also see that DACA can achieve the same effect as the other two algorithms in the matching degree of clustering effect.

4.3 | Parallelization run time comparison

It is well-established that spark's machine learning library[18] provides the implementation of some basic common machine learning algorithms. Among them, the most famous is the application of K-means and the parallel implementation of DBSCAN[22] and optics algorithm in density clustering.[21] We tested and compared the clustering time performance of these three clustering algorithms on our real GPS dataset. As shown in Figure 6A, with the increase of datasets, the

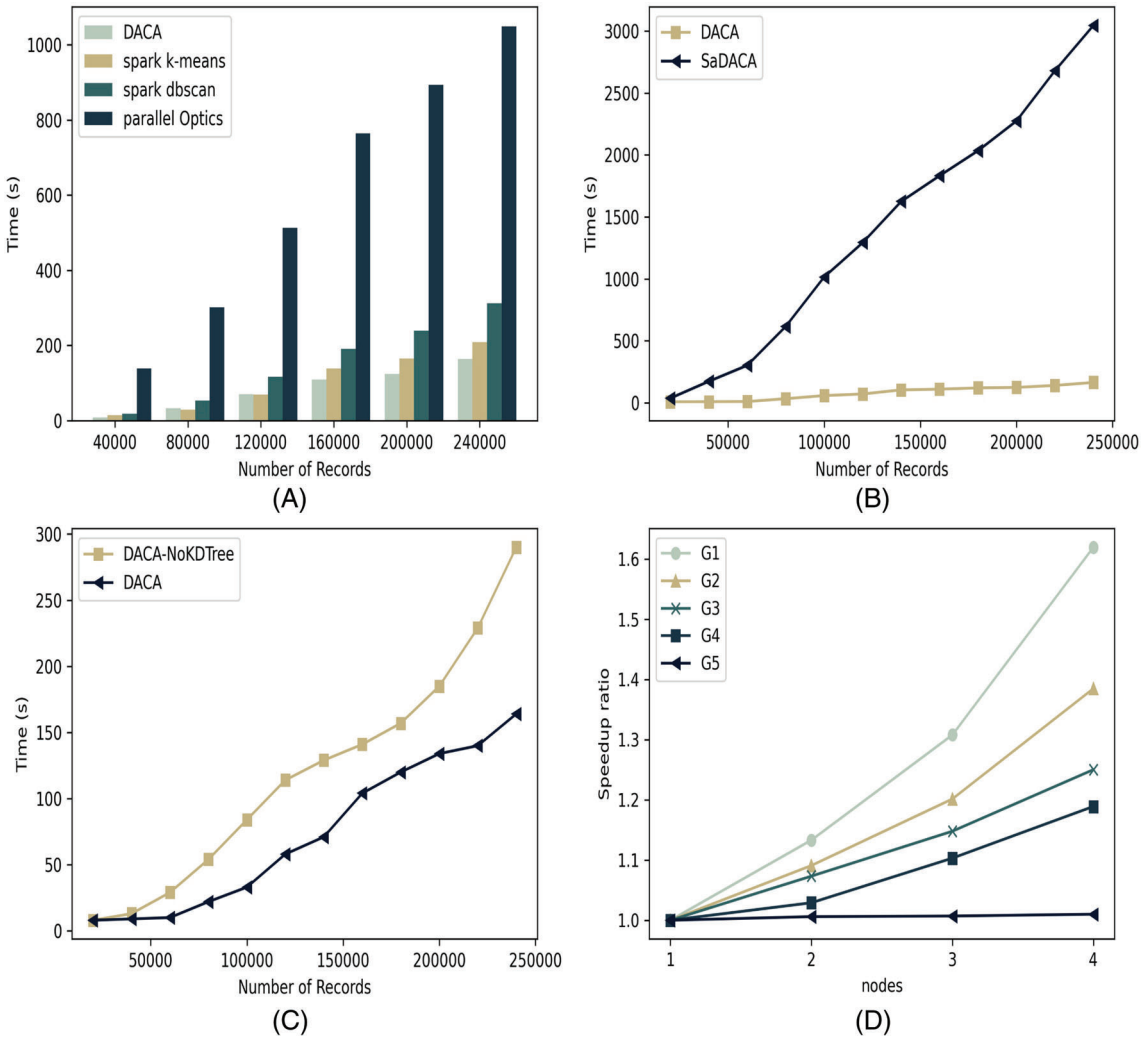

FIGURE 6 (A) Comparison of clustering time performance of DACA, spark Mlib K-means, DBSCAN on spark, and parallel optics on real GPS trajectory data. (B) Comparison of the clustering time performance of DACA and SaDACA algorithms on real GPS trajectory data in Apache Spark-distributed environment. (C) Comparison of the clustering time performance between DACA with KD-tree and DACA with traditional cyclic search on real GPS trajectory data. (D) The acceleration ratio of DACA with the increase of nodes under the spark framework

execution time of each algorithm increases, and the parallel algorithm performs parallel Options[21] has the largest increase in time. The main reason is that in the process of clustering, data are forwarded and communicated many times, and the communication between each node increases the network overhead time. With the increase in data sets, DACA has the most ideal execution schedule.

Experiments on the four standard datasets are not sufficient to prove the efficiency of the algorithm in a real big data environment. We used the collected taxi GPS trajectory data, which contains more than 500,000 GPS track data from two days—Monday and Sunday—corresponding to four periods of data every day. As shown in Table 3, we compared parallel optics, spark DBSCAN, and DACA algorithms based on density. As can be seen from Table 3, the clustering effects of parallel optics and spark DBSCAN are similar on these 2 days, whereas the overall clustering effect of DACA is more refined for the data of the 2 days. However, in general, the clustering effects of parallel optics, spark DBSCAN, and DACA are similar, which shows the effectiveness of DACA on real large datasets.

We conducted experiments on the spark cluster built in the laboratory and compared the corresponding experiments of a single machine (SaDACA) and parallel (DACA). As shown in Figure 6B, with an increase in the dataset, SaDACA will increase the clustering time of the algorithm with an increase in the amount of data, showing a trend of linear growth, whereas the parallel DACA appears to be much smoother.

TABLE 3 Cluster results of parallel optics, spark DBSCAN, and DACA on four different time periods on different days (weekdays and rest days)

Date	Time	Parallel optics	Spark DBSCAN	DACA
Monday	7:00–9:00	6	6	7
	9:00–11:00	13	14	18
	13:00–15:00	12	14	11
	15:00–17:00	5	7	8
Sunday	7:00–9:00	7	6	11
	9:00–11:00	8	9	11
	13:00–15:00	10	13	11
	15:00–17:00	8	11	10

4.4 | The effect of the KD-tree

A KD-tree is a type of tree data structure that stores instance points in a k-dimensional space for quick retrieval. In the implementation of DACA, to speed up the search and determination of the sparse grid data category, we added a KD-tree to the implementation of DACA to more efficiently determine the sparse point category in the sparse grid.

We compared the traditional loop search and the KDTree-based search. As shown in Figure 6C, we found that on DACA without KD-tree optimization, the execution time of the corresponding algorithm is significantly higher than that of the KD-tree search. The execution time of the algorithm. However, with the increase in datasets, the running time of both algorithms was clearly improved. In addition, the search algorithm using KD-tree proceeded more slowly as the amount of data increased.

4.5 | Parallel performance test

To verify and analyze the parallel computing performance of the algorithm in the spark framework, we need to calculate the speedup ratio of the algorithm. The speedup ratio is an important index for verifying parallel computing.

In this experiment, we randomly selected five GPS trajectory data of G1 (10,000), G2 (100,000), G3 (200,000), G4 (300,000), and G5 (500,000) for clustering verification. The experimental results are shown in Figure 6D. As can be seen from the figure, the algorithm in this study has a good speedup after parallelization. When the amount of data is small, as the number of nodes increases, the speedup gradually tends to become flat or gradually decreases. As shown in Figure 6D, with an increase in the computing nodes, the speedup ratio always tends to 1. This is mainly because when the data scale is small, the data volume is far less than the data that the cluster can handle. At this time, the data are divided into many small pieces and are distributed to each node, which increases the communication time and task scheduling time and reduces the calculation speed. From G2 to G5, we can observe that with an increase in data volume and node number, the acceleration ratio increases linearly. As shown in Figure 6D, when the number of data reaches 500,000, the acceleration ratio reaches 1.62, which indicates that the algorithm in this study has a good speedup ratio after parallelization and is suitable for processing large-scale data.

5 | CONCLUSION

In this study, we proposed a parallel clustering algorithm (DACA) based on adaptive meshing and decision graphs and the corresponding implementation and optimization through Apache Spark. We proved the advantages of this algorithm in running time and single-machine large-scale datasets through experimental comparison. To adapt to the clustering needs of the real-world big data environment, and provide a better extension. We found that for large-scale datasets, under the premise of ensuring the accuracy of the algorithm, the implementation of parallelization can effectively reduce the

execution time of the algorithm. The algorithm is not only suitable for traditional large digital data sets, but also suitable for text and image clustering tasks. At the same time, we also conducted corresponding comparative experiments with the current common parallel clustering algorithms. We showed that DACA also had obvious advantages in the execution efficiency of the algorithm. Through these results, we can conclude that our parallelization implementation is successful and effective and can also be applied in real clustering business environments.

In future research, we will focus on the basic improvement of the algorithm. The clustering algorithm can potentially be used for practical application after the further improvements in basic algorithm, considering the implementation and optimization of the corresponding distributed algorithm.

AUTHOR CONTRIBUTIONS

Jing He: propose the basic idea and algorithm design and improve the proposed method; finish the final version. **Jun Zhou:** implement the experiment and design the draft method. provide the draft version. **Haoyu Wang:** propose the idea on domain related algorithm design and the evaluation method; contribute the algorithm design. **Li Cai:** propose the basic idea and algorithm design, funding acquisition, supervision, review and editing.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Jun Zhou https://orcid.org/0000-0002-5294-8296

REFERENCES

1. Chen M, Mao S, Liu Y. Big data: a survey. *Mob Netw Appl*. 2014;19(2):171-209.
2. Liu R, Li X, Du L, Zhi S, Wei M. Parallel implementation of density peaks clustering algorithm based on spark. *Procedia Comput Sci*. 2017;107:442-447.
3. Yonglai Z, Yaojian Z, School S. Review of clustering algorithms. *J Comput Appl*. 2019;39(7):1869–1882.
4. Oliva G, Setola R. Distributed K-means algorithm. *CoRR*. 2013;arXiv:1312.4176.
5. Park H, Jun C. A simple and fast algorithm for K-medoids clustering. *Expert Syst Appl*. 2009;36(2):3336-3341.
6. Guha S, Rastogi R, Shim K. CURE: an efficient clustering algorithm for large databases. In: Haas LM, Tiwary A, eds. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998*. ACM Press; 1998:73-84.
7. Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. In: Jagadish HV, Mumick IS, eds. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*. ACM Press; 1996:103-114.
8. Zhang J, Li C, You C, et al. Self-supervised convolutional subspace clustering network. *Computer Vision Foundation*. IEEE; 2019:5473-5482.
9. Mukherjee S, Asnani H, Lin E, Kannan S. *ClusterGAN: Latent Space Clustering in Generative Adversarial Networks*. AAAI Press; 2019:4610-4617.
10. Zhang Y, Lu J, Liu F, et al. Does deep learning help topic extraction? a kernel K-means clustering method with word embedding. *J Inf Secur*. 2018;12(4):1099-1117.
11. Deng C, Liu Y, Xu L, et al. A MapReduce-based parallel K-means clustering for large-scale CIM data verification. *Concurr Comput Pract Exp*. 2016;28(11):3096-3114.
12. Chen G, Cheng Y, Jing W. DBSCAN-PSM: an improvement method of DBSCAN algorithm on Spark. *Int J High Perform Comput Netw*. 2019;13(4):417-426.
13. Corizzo R, Pio G, Ceci M, Malerba D. DENCAST: distributed density-based clustering for multi-target regression. *J Big Data*. 2019;6:43.
14. Fei W, Guo-Yin W, Zhi-Xing LI, Si-Yuan P. Clustering by fast search and find of density peaks based on grid. *J Chinese Comput Syst*. 2017;38(5):1034–1037.
15. Liu R, Huang W, Fei Z, Wang K, Liang J. Constraint-based clustering by fast search and find of density peaks. *Neurocomputing*. 2019;330(22):223-237.
16. Olabelurin A, Veluru S, Healing A, Rajarajan M. Entropy clustering approach for improving forecasting in DDoS attacks; 2015:315-320.
17. XiaoKang C, ZhuSong L. K nearest neighbor query based on improved KD-tree construction algorithm. *J Guangdong Univ Technol*. 2014;31(3):119–123.
18. Documentation S. Spark overview apache software foundation [online]; 2019. https://spark.apache.org/docs/2.4.3/
19. He Y, Tan H, Luo W, Feng S, Fan J. MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data. *Front Comp Sci*. 2014;8(1):83-99.
20. Markiewicz M, Koperwas J. Hybrid partitioning-density algorithm for K-means clustering of distributed data utilizing OPTICS. *Int J Data Warehous Min*. 2019;15(4):1-20.

21. Patwary MMA, Palsetia D, Agrawal A, Liao W, Manne F, Choudhary AN. Scalable parallel OPTICS data clustering using graph algorithmic techniques. In: Gropp W, Matsuoka S. , eds. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC'13, November 17 - 21, 2013.* , ACM. 2013;49(1–49):12.
22. Chang H, Yeung DY. Robust path-based spectral clustering; 2008.
23. Malondkar A, Corizzo R, Kiringa I, Ceci M, Japkowicz N. Spark-GHSOM: growing hierarchical self-organizing map for large scale mixed attribute datasets. *Inf Sci*. 2018;496:5–24.
24. Roghani H, Bouyer A, Nourani E. PLDLS: a novel parallel label diffusion and label selection-based community detection algorithm based on spark in social networks. *Expert Syst Appl*. 2021;183:115377.
25. Zhang Y, Yin D, Wu B, Long F, Bian X. PLinkSHRINK: a parallel overlapping community detection algorithm with link-graph for large networks. *Soc Netw Anal Min*. 2019;9:66.
26. Dharavath R, Arora NS. Spark's GraphX-based link prediction for social communication using triangle counting. *Soc Netw Anal Min*. 2019;9(1):28.
27. Chen J, Azer ES, Zhang Q. A practical algorithm for distributed clustering and outlier detection. In: Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, eds. *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018*; Conference and Workshop on Neural Information Processing Systems; 2018:2253-2262.
28. Ghani NA, Hamid S, Hashem IAT, Ahmed E. Social media big data analytics: a survey. *Comput Hum Behav*. 2019;101:417-428.

How to cite this article: He J, Zhou J, Wang H, Cai L. DACA: Distributed adaptive grid decision graph based clustering algorithm. *Softw Pract Exper*. 2022;52(5):1199-1215. doi: 10.1002/spe.3060