Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 6156799, 11 pages
https://doi.org/10.1155/2022/6156799

Research Article

Clustering Algorithm for Big Datasets with Mixed Attribute Features under Spark

Jiankai Wang

School of Information Engineering, Chengdu College of Arts and Sciences, Chengdu, Sichuan 610401, China

Correspondence should be addressed to Jiankai Wang; 0000969@sctu.edu.cn

Received 27 January 2022; Revised 21 February 2022; Accepted 4 March 2022; Published 11 April 2022

Academic Editor: Gengxin Sun

This paper presents an in-depth study and analysis of large datasets of mixed and attribute features under Spark using a large dataset clustering algorithm. The classical algorithm *K*-means based on division and the density-based clustering algorithm DPC, which has become more popular in recent years, are selected as the research objects of this paper. Secondly, the original *K*-means algorithm is improved by combining holdout validation and *K*-means++ method to address the shortcomings of the *K*-means algorithm that the number of class clusters *K* needs to be set in advance and the initial class cluster centers are chosen randomly, which leads to unstable iterations and slow convergence of clustering results. The similarity matrix will be continuously updated during the iterative process. It mainly refers to the process of dividing objects into multiple classes according to the degree of similarity between objects. After the division, the objects within the class are like each other, while the objects between the classes are different from each other. The comparison experiments of the improved algorithm before and after the MovieLens dataset are conducted to verify that the new algorithm has better performance in terms of clustering accuracy and efficiency. Again, to address the drawback that the clustering results in the DPC algorithm rely heavily on the subjective selection of the truncation distance parameter cd, and it is difficult to handle datasets with complex distribution and large density variation, the algorithm can generate the optimal cd adaptively by combining *K*-nearest neighbors and introducing the distance comparison quantity, which has a better performance by considering the overall and local distribution of the data. The feasibility of the improved method is verified by validating the algorithm with artificial datasets and UCI datasets as well as separation tests. Finally, the parallelized design and implementation of the improved *K*-means algorithm and CDPC-KNN algorithm are completed by building a Spark clustering environment, and the parallelized algorithm is verified to have much better data processing capability and be more adaptable to the clustering analysis of large-scale data by comparing algorithm string parallelism experiments.

1. Introduction

With the rapid development of the Internet, the data generated by various industries is exploding, and the situation of "data explosion and knowledge scarcity" has gradually emerged. How to extract useful information from the huge amount of data and understand the knowledge contained in the data has become an urgent problem. Data mining is the process of extracting hidden, unknown, and useful information from the huge amount of data stored in databases or data warehouses, which can help people better analyze data, utilize data, and use it to analyze future trends. Cluster analysis is a key topic in data mining, which mainly refers to the process of dividing objects into multiple classes based on their similarity to each other, after which the objects within the classes are like each other, while the objects between the classes and the classes are different from each other [1]. The parallelization of data mining algorithms under the framework of distributed computing has also become a hot spot in the field of data mining research. Currently, clustering analysis has become a very active topic in the field, but due to the complexity of data, the diversity of data structures, and the dramatic increase in the amount of data, traditional clustering methods have many shortcomings and deficiencies in dealing with the problem. To further optimize the clustering process, we can introduce some bioinspired algorithms to optimize the clustering process, in addition to improving the algorithm itself.

Data mining can uncover hidden and valuable information from massive amounts of data. By analyzing consumer spending records, it was concluded that male consumers tend to buy several bottles of beer when they buy baby diapers [2]. Google built a predictive model in the US by analyzing the words people repeatedly search for and correlating that data with the Centers for Disease Control's data on seasonal flu transmission in previous years, which successfully predicted the spread of the flu and was specific to a particular location; Twitter A series of customized customer data streams were built by clustering the user's address, as well as the location of tweets and the content of tweets [3]. For example, by clustering movie names, tweeting locations, and movie reviews, you can know which movies are the most popular in Tokyo, Shanghai, and London. The collection of diverse data may be the real value of social networking sites now, and they may be able to squeeze out advertising as a major source of revenue for social networking sites [4]. However, the new generation of big data computing method framework and programming model is running in parallel, and the traditional data mining algorithm based on the serial mode of a single machine is difficult to apply directly to the distributed framework; therefore, the parallelization research of data mining algorithm in the distributed computing framework has also become a major hot spot in the field of data mining research.

Spark cluster is very suitable for processing massive data because of its distributed characteristics with powerful arithmetic [5]. Combining data mining technology with the Spark platform to realize the parallelization of these traditional data mining algorithms can well meet the demand for information mining of big data, which is of great significance for processing big data with large scale and variety. Due to the simplicity and efficiency of Spark, it will be very likely to replace Hadoop as the new generation of the data mining platform. The existing clustering algorithm is improved and integrated into the Spark platform, and this clustering algorithm can overcome the defect of the *K-means* algorithm which can only be used for the spherical dataset and make this algorithm more stable, and this algorithm makes the library richer for MLlib, the machine learning algorithm library, and enriches the computing power of Spark computing platform. It is also valuable for the continued development of the Spark computing platform to enrich the computational power of the platform and improve the clustering effect of the clustering algorithm of the platform, therefore, improving the existing machine learning algorithms and embedding them in the MLlib library, and finally, the distributed deployment of such algorithms on the Spark platform is of great importance for the development of the current Spark platform.

2. Related Works

The study of the clustering problem is one of the quite active areas in data mining. Clustering is a basic method for data analysis, and the main purpose is to divide a set of objects (usually data points in space) into several classes according to different attribute values and to require that objects within the same class be as similar as possible, and objects in different classes are as different from each other as possible [6]. In real life, the relationship between many things is fuzzy and unclear. In this fuzzy situation, K-means and another hard clustering algorithm are no longer applicable, and then a fuzzy clustering algorithm is needed. The fuzzy C-means clustering algorithm is studied in terms of the weight assignment and the number of clusters [7]. It is very likely that it will replace Hadoop as a new generation of data mining platform, improve the existing clustering algorithm, and integrate it on the Spark platform, so that this clustering algorithm can overcome the defect that the *K-means* algorithm can only be used for spherical datasets, making this algorithm more stable. The number of clusters is determined automatically according to the characteristics of the data, especially the nonlinear relationship between clusters, and the fuzzy clustering rules are weighted to improve the clustering accuracy. The processing of boundary points, outlier points, and noise points has also been studied successively [8]. For example, considering that the current methods can obtain more information about the earth surface in a short time, but the objects on the images are usually unclear and the boundaries are not obvious due to various factors, a hybrid optimization method based on the semisupervised probabilistic fuzzy C-mean clustering algorithm and particle swarm algorithm with interval type 2 is proposed to deal with this kind of image problem [9]. Experiments prove that the algorithm proposed in this paper is more accurate and better in dealing with outliers and noise.

It needs to determine k clusters before the algorithm runs, is sensitive to the initial centroid selection, and is highly influenced by outlier points. However, due to the obvious advantages of the K-means algorithm, it has also attracted many scholars to improve it by plotting the number of different clusters k against the sum of squared errors to derive the value of k at the elbow point as the most clustered number of a dataset. The clustering effect is improved by calculating the density of each data in the dataset and selecting the initial centroid from the highest to the lowest density [10]. A detailed review of the K-means clustering algorithm is given, and the future of the K-means algorithm is summarized and investigated in two aspects. The problem of privacy security in statistical databases is proposed, and the differential privacy model is a privacy-preserving model for this problem [11]. In the differential privacy model, whether a piece of data is in the database or not does significantly change the query result of the database. The differential privacy model ensures that, in querying a statistical database, even if an attacker has all the information except the target information, he cannot accurately obtain the target information by querying the database [12]. Differential privacy achieves privacy protection by adding random noise that matches certain characteristics to sensitive information in the data but does not affect certain statistical properties of the data.

Regardless of the size and structure of the database, if it contains valuable information, you can rely on data mining techniques to complete the discovery and analysis of useful information. The basic principle of such algorithms is to divide the data object space into several small cells, which is done by using a multiresolution grid data structure [13]. Choose an appropriate cluster analysis algorithm. Finally, compare the difference between the large clustering results under standard clustering and the clustering results obtained by this clustering. And through the trained test classification, the effectiveness of the algorithm is analyzed and the result is calculated. Because of this feature, the processing speed of this class of algorithms is faster, but the quality of clustering is very dependent on the grid division of data objects; if the granularity of the bottom layer of the grid is divided too fine, it leads to high time overhead of the algorithm, reducing the efficiency of the algorithm; but if the granularity of the bottom layer of the grid structure is too coarse, it will affect the final quality of the grid algorithm clustering. Model-based clustering methods include neural networks and statistical methods.

3. Analysis of Clustering Algorithms for Big Datasets with Mixed Attribute Features under Spark

3.1. Design of the Hybrid Attribute Feature Clustering Algorithm under Spark. The original dataset required for clustering analysis is preprocessed, and then the feature attributes to be analyzed in the implementation of clustering are extracted from the complex dataset [14]. Next, the preprocessed dataset needs to be partitioned, and the partitioning process is to select the applicable similarity measure formula for the dataset based on the feature attributes obtained in the step and use this formula to partition the dataset. Then, to obtain a faster and more accurate classification of the data objects, a suitable clustering analysis algorithm is selected by combining the feature attributes and similarity measure formulas obtained in the step. The similar data objects are divided into the same cluster, and the data objects with small similarity are divided into different clusters. Clustering algorithms can be described symbolically using the concept of sets in mathematics. Finally, the differences between the large clustering results under standard clustering and the clustering results obtained by this clustering are compared. And the validity of the algorithm is analyzed and the final results are calculated by the trained test classification.

The essence of cluster analysis is an unsupervised classification process, which is the same as classification algorithms in that they both result in dividing data objects into several categories, with the difference that the data objects of cluster analysis are unlabeled and the category information is unknown [15]. The clustering analysis algorithm is based on the data characteristics of the objects and calculates the similarity between the objects by a specific similarity measure so that similar data objects are grouped into the same cluster and data objects with less similarity are grouped into different clusters. The clustering algorithm can be described symbolically using the concept of sets in mathematics. Suppose the set of data objects is C.

$$C = \left\{X_1^2, X_2^2, X_3^2, \ldots, X_n^2\right\}. \tag{1}$$

The clustering algorithm divides the dataset into m disjoint subsets $C_1, \ldots, C_m$, which can also be called data clusters, and all the clusters constitute the whole dataset. Each data object in the dataset is finally classified into a class cluster and will be classified into only one class cluster.

$$C = \begin{cases} C_i \cap C_j & = & O \\ C_i \cup C_j & = & \mathrm{C} \end{cases}. \tag{2}$$

The similarity measure is the basis for the execution of clustering algorithms. All clustering algorithms are based on the similarity between data points for data relationship discrimination, and then the specific clustering algorithm division strategy is to achieve the clustering requirements of high intracluster similarity and low intercluster similarity, and the choice of the similarity measure directly affects the clustering results of the data. The most commonly used similarity measure for clustering is to calculate the distance between data objects, and the object distance is inversely proportional to the similarity; the smaller the distance value, the higher the similarity between data objects, and the larger the distance value, the lower the similarity.

$$\mathrm{Sim}\left(o_i, o_j\right) = \frac{1}{1 - \mathrm{dist}\left(o_i, o_j\right)}. \tag{3}$$

Hierarchical-based clustering algorithms need to calculate the distance between data objects, then synthesize the closest data points into a cluster, and then calculate the distance between each cluster and keep clustering. Hierarchical clustering algorithms can be divided into two types: top-down and bottom-up. Top-down is a split hierarchical clustering algorithm, which treats all data objects as one class and continuously splits them by calculating distances; bottom-up is the opposite, which is a cohesive hierarchical clustering algorithm, which starts by treating all data objects as a single class and then finds similar classes by calculating distances. To achieve the clustering requirements of high similarity within clusters and low similarity between clusters, the choice of similarity measure directly affects the clustering results of data. The commonly used hierarchical clustering algorithms are the BIRCH algorithm, CURE algorithm, and ROCK algorithm. Hierarchical clustering algorithm does not need to set the number of clusters and can reveal the relationship of different hierarchical classes and can define the distance and rule similarity well, but it is relatively sensitive to outliers and the algorithm is computationally expensive.

$$\begin{aligned} R_{ij} &= \frac{\sum_{k=1}^{m}\left(x_{ik}^2 + x_i^2\right)\left(\left(x_{jk}^2 - x_j^2\right)\right)}{\left[\sum_{k=1}^{m}\left(x_{ik}^2 + x_i^2\right)\left(\left(x_{jk}^2 - x_j^2\right)\right)^2 \sum_{k=1}^{m}\left(x_{ik}^2 - x_i^2\right)\left(\left(x_{jk}^2 + x_j^2\right)\right)^2\right]}, \\ x_{ik}^2 &= \frac{1}{m}\sum_{k=1}^{m} x_{ik}, x_{jk}^2 \\ &= \frac{1}{m}\sum_{k=1}^{m} x_{jk}. \end{aligned} \tag{4}$$

If the absolute value of the correlation coefficient is closer to 1, the data objects are more similar, and if the absolute value is closer to 0, the data objects are less similar [16]. The K-means algorithm is to randomly select k initial centroids from the dataset, traverse the distance between each point in the dataset and the k centroids, divide the data points into clusters with the closest centroids, and then each cluster updates. The center of mass of the data points in each cluster is used as the centroid of the next iteration until the loss function no longer changes or the number of iterations reaches the upper limit.

Therefore, Hadoop and Spark-based frameworks are widely used in the field of itemset mining due to their easy data management and excellent fault tolerance. According to the design principle of the FPG algorithm, it can be effectively combined with a distributed platform to extend the itemset mining, and the easiest way is to port the algorithm directly to the platform for implementation. In the previous chapter, a detailed overview of the two platform technologies has been given, and this chapter will compare the item set mining based on the two platforms and analyze the differences between them by combining theory with practice. The parallelism principle based on data slicing decomposes the solution problem into multiple similar subproblems and performs the same function on different data blocks (Figure 1). The execution requires the set of candidate itemsets to be small enough to satisfy the idea that all subproblems can be stored in the main memory.

First of all, for Spark's data cache, the RDD structure used can load the dataset directly in memory and perform multiple reads to help achieve faster and more efficient mapping and computation operations. For datasets with complex data forming extremely large data volumes, more complex algorithms are usually used to analyze such data. According to the design principle of the FPG algorithm, it can effectively combine the distributed platform to expand the itemset mining. The simplest method is to directly transplant the algorithm to the platform for implementation. For iterative algorithms, more complex logical calculations and repeated operations are often required. Based on the shortcomings derived from the MapReduce framework, it does not effectively solve the two problems of data caching and algorithm iteration. When using this simple framework to process more complex data, it is necessary to repeatedly overlay Map and Reduce operations to complete. The overall overhead of the algorithm increases with each new Map and Reduce [17]. As a result, MapReduce-based programs implemented on Hadoop are not well adapted to the iterative process. When dealing with large datasets, it puts a huge burden on the network and disks. The above analysis shows that the MapReduce-based framework has major shortcomings when combined with algorithms that require multiple iterations.

$$\mathrm{dis}\left(x_m, c_k\right) = \sum_{n=1}^{N}\left(w_n^2 \times \left|x_{nm} - c_{kn}\right|^r\right). \tag{5}$$

This is caused by the system architecture of Hadoop itself, which cannot be circumvented by any algorithmic optimization. For the Spark platform, which has the core design structure of RDD, based on the advantage of RDD, after the first iteration is loaded, the subsequent iterations can use the intermediate results without additional computation time. Therefore, it is necessary to study a parallel itemset mining algorithm based on the Spark platform.

$$\mathrm{Minimize}\, S(U, c) = \sum_{m=1}^{M}\sum_{k=1}^{K} u_{mk}^2 \times \mathrm{dis}\left(x_m, c_k\right). \tag{6}$$

Spark is a fast data processing framework and its system architecture is shown in Figure 2. Spark uses a cluster resource management platform to share and allocate resources in a clustered environment; i.e., datasets are allocated or partitioned to multiple nodes in the same cluster to be processed in batches simultaneously. In addition, Spark reads and writes data objects through memory, which is very fast.

With the continuous development of parallelism, many distributed file systems such as HDFS, Amazon, HBase, etc. have been developed for Spark's underlying use. In addition to Spark Core, Spark's other libraries are built on top of the RDD programming model. In particular, the data stream input and output are controlled through Spark Streaming, which allows programs to process large volumes of real-time data in a way that is as superior as normal RDDs. For iterative algorithms, more complicated logical calculations and repeated operations are often required. Based on the shortcomings derived from the MapReduce framework, it cannot effectively solve the two problems of data caching and algorithm iteration. Like other IDEs that have their database interaction platforms, Spark interacts with Apache Hive's SQL query language through the Spark SQL API. This API enables the conversion of Spark SQL queries to Spark operations. Each RDD is a table created by the database.

Spark programming can be implemented in Scala, Python, or Java. Compared to other programming environments, programs written in Python are less code-intensive, faster and more explanatory and have the same functionality.

$$J_c = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{i,j}^2 \left\| c_i^2 + x_j^2 \right\| \times \mathrm{dis}\left(x_m, c_k\right). \tag{7}$$

Spark's fault tolerance mechanism relies heavily on the dependency relationships between RDDs, which form a directed acyclic graph (DAG) that records the parent RDDs on which the child RDDs depend. The DAG diagram plays a key role; by traversing the dependencies of the lost partition and tracing back along with the dependencies, we can find the parent RDD information of the lost partition and the source of the partition data. In this case, if the lost partition has a narrow dependency, the data recovery efficiency is higher because the partition is fully corresponded by one or more parent partitions, and it only needs to be recalculated according to the parent RDD, which is less overhead, but for the lost partition with wide dependency, although the parent RDD can be found through the DAG graph, the parent RDD corresponds to more than one child RDD, which involves

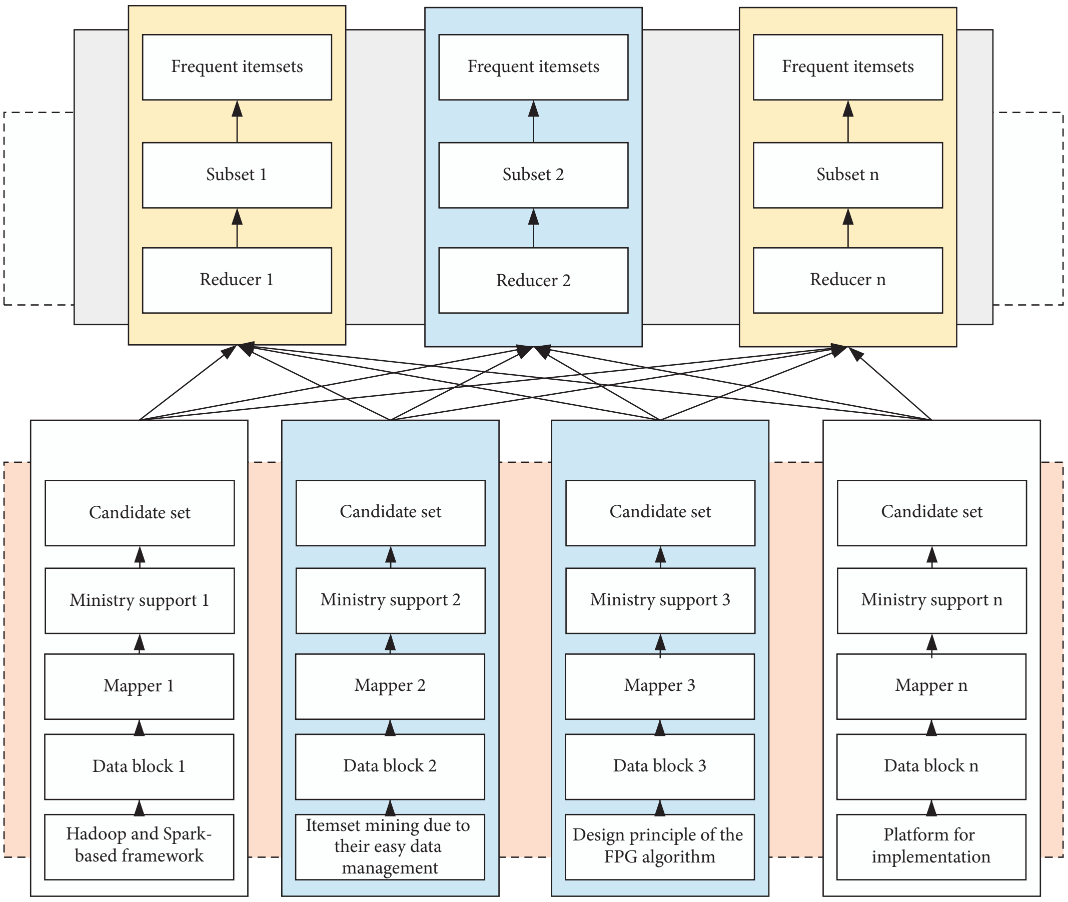


Figure 1: Parallel scheme based on data slicing.

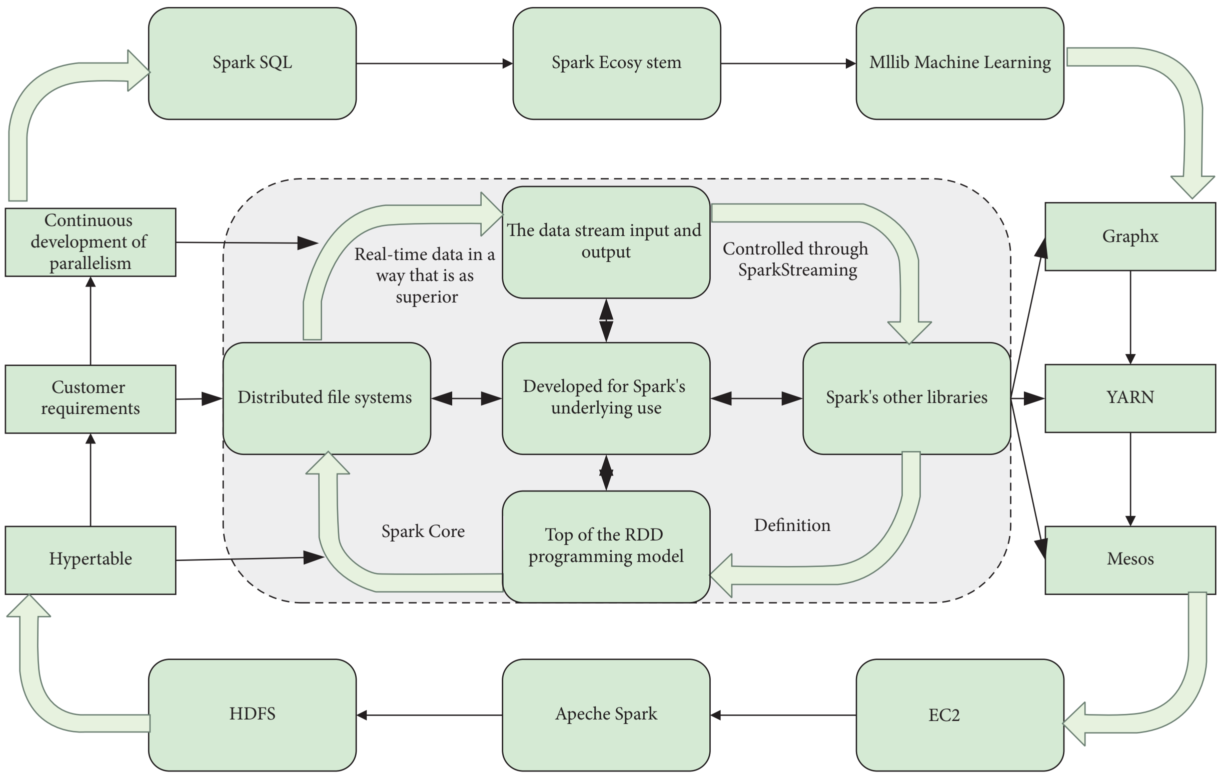

Spark SQL
Spark Ecosy stem
Mllib Machine Learning
Continuous development of parallelism
The data stream input and output
Real-time data in a way that is as superior
Controlled through SparkStreaming
Graphx
Customer requirements
Distributed file systems
Developed for Spark's underlying use
Spark's other libraries
YARN
Spark Core
Top of the RDD programming model
Definition
Hypertable
Mesos
HDFS
Apeche Spark
EC2

Figure 2: Spark architecture diagram.

more than one parent partition, so it cannot be calculated directly and is difficult to trace. Both checkpoint and logging are important means of RDD persistence, and Spark can compare the overhead of different methods to automatically choose the optimal data recovery method.

There are two mechanisms for implementing differential privacy, a Laplace mechanism that provides privacy protection for numerical results and an exponential mechanism that provides privacy protection for nonnumerical results [18]. In the interactive data publishing environment, when a user initiates a query request to the database, the administrator initiates a query to the database based on the user request and sends the results to the user with differential privacy protection, thus protecting the privacy of the data individuals in the database. In the noninteractive data publishing environment, the administrator publishes a "sanitized" version of the original dataset based on the differential privacy conditions; i.e., the published dataset is a dataset processed by differential privacy technology and made available to users for statistical queries. The parent RDD on which the child RDD depends is clearly documented in the DAG diagram. When a single point of failure occurs and data is lost, the partition data needs to be restored, and the DAG graph plays a key role by traversing the dependencies of the lost partition. In differential privacy-preserving data mining, we want to improve the performance of data mining while protecting data security. Differential privacy provides two implementation models for data mining: the interface model and the full access model.

In the interface model, the manager only provides an access interface that implements differential privacy protection and does not publish the original dataset. The data miner can only get the needed information for data mining through the access interface, and the access interface stops the data miner's query requests after the privacy budget is exhausted. In this environment, the data miner is not trusted. In contrast, in the full access model, the data miner is trusted, so the data miner can access the original dataset and perform data mining but must adopt a differential privacy algorithm to protect the privacy of the data during the mining process.

3.2. Experimental Analysis of the Feature Large Dataset. First, we experimented with three experimental algorithms with the same privacy budget, and the maximum number of iterations of the algorithms was set to 10. Ten experiments were conducted for each algorithm and the actual number of iterations after their algorithms was recorded. In the DPK-means algorithm, blindly selecting the centroids and unreasonable privacy budget allocation will cause the DPK-means clustering algorithm to converge slower and the number of iterations to increase. Therefore, reasonable initial centroids and privacy allocation can reduce the number of iterations of the algorithm, and we observe the average number of iterations of the three clustering algorithms in datasets $D1$, $D2$, and $D3$ with increasing ε. The average number of iterations of the three algorithms in different datasets was shown in Figure 3.

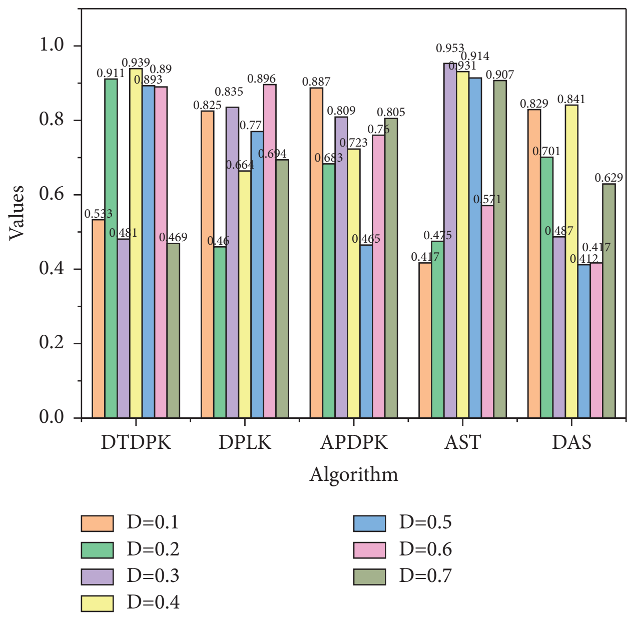


Figure 3: Average number of iterations of the three algorithms on $D1$.

Then, the privacy budget allocation AST method in the DTDPK-means algorithm is validated by first selecting the initial centroids using the DAS method and then comparing the privacy budget allocation methods using the AST method, dichotomous method, and the equivariant series, respectively. The maximum number of iterations is also set to 10, and the privacy budget ε is gradually increased from 0.5 using dataset $D1$ to observe the $F1$ values of the three methods.

The AST method can get a certain degree of improvement in the clustering effect of the dataset compared to the dichotomous and equivariant series methods. The reason is that the dichotomous method may cause the deformation of the center of mass when the noise is added in the later iterations, while the AST method allocates more privacy budget to the earlier iterations compared with the equivariant method, and the noise added in the later iterations is guaranteed not to cause the deformation of the center of mass. In Figure 3, the $F1$ value of the dichotomous method is the highest when the number of iterations is 4, which indicates the best clustering effect, but the $F1$ value drops sharply when the number of iterations is 6 and 8 because the privacy budget allocated in the dichotomous method is smaller than the minimum privacy budget required for each iteration of the dataset when the number of iterations is greater than 5, which causes the deformation of the center of mass and makes the clustering effect decrease. Therefore, it cannot be calculated directly, and it is difficult to trace back. Both checkpoint and logging are important means of RDD persistence. Spark can compare the cost of different methods and automatically select the optimal data recovery method. Method and the equivariant method are relatively stable, and the AST method has a certain degree of improvement in $F1$ value compared with the equivariant method [19].

The experimental dataset needs to be not too large compared to the serial algorithm; otherwise it will cause the serial algorithm to fail in mining. Therefore, the experimental datasets in this section are downloaded from the UCI website, using two datasets with different characteristics: mushroom and accidents. The mushroom dataset contains features of 23 mushroom species, specifically covering 119 attribute values and 8124 instances; the accidents dataset outlines data related to traffic accidents in the United States, with 307 attribute values and 2125 actual cases in the dataset. The accidents dataset summarizes data related to traffic accidents in the US. The dataset includes 307 attribute values and 2125 actual cases. Since the dataset provider has processed it beforehand, it can be directly used in this experiment. The specific characteristics of the two datasets are shown in Figure 4.

The two datasets are mined using the serial FPG algorithm and the parallel SFPG algorithm, and the time required to mine the two algorithms at different support levels is compared by setting different support level thresholds (min_Support). To reduce the influence of the hardware environment or some eventualities on the result data, the experiment is repeated 10 times in the same environment, and the result is averaged from the 10 results. Figure 4 shows the mining time results of the FPG and SFPG algorithms on the mushroom dataset when the min_Support values are 0.4, 0.35, 0.3, 0.5, and 0.25, respectively; Figure 4 shows the mining time results of the FPG and SFPG algorithms on the accidents dataset when the min_Support values are 0.6, 0.55, and 0.45, respectively. The parallel SFPG algorithm based on Spark is significantly faster than the serial algorithm in terms of mining time for both the mushroom and accidents datasets, indicating that the parallel SFPG algorithm can effectively improve the mining efficiency of the standalone algorithm. Moreover, the mining time fluctuation of the SFPG algorithm under the support variation is smaller than that of the serial algorithm, which also indicates that the SFPG algorithm has a better stability.

The execution time of both the MRFPG algorithm and SFPG algorithm decreases as the number of nodes increases. From observing the comparison of the two algorithms mining at the same number of nodes, the Spark-based SFPG algorithm mines faster than the MapReduce-based MRFPG algorithm at any number of nodes from 1 to 5. This is consistent with the theoretical reasons analyzed in the previous section because it can repeatedly use the intermediate result RDDs, so it is not necessary to load the dataset in each iteration, which can effectively reduce the network and I/O consumption in the case of repeated iterations of the algorithm. Therefore, parallel mining of frequent itemset using the Spark platform can help reduce the execution time of mining algorithms to a certain extent [20]. Also set the maximum number of iterations to 10, use the dataset $D1$ to conduct experiments, the privacy budget ε is gradually increased from 0.5, and observe the $F1$ value of the three methods. The experiments on the complete dataset web docs also show that the traditional FPG algorithm causes memory overflow and cannot be executed properly in a standalone environment. This also shows that the parallelization-based algorithm can effectively solve the memory overflow problem of the traditional algorithm.

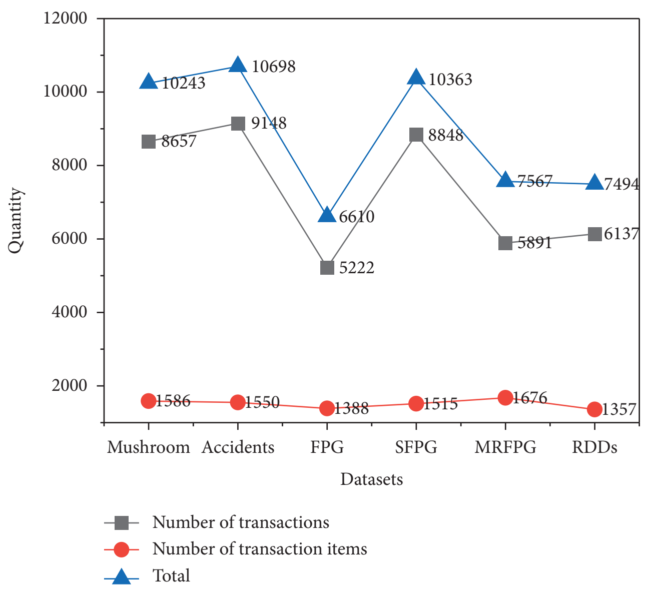


Figure 4: Dataset characteristics.

In the process of frequent itemset mining, each mining task is computationally intensive, so the allocation of items to groups and the choice of parallelism strategy directly affect the memory usage, the load on each node, and the communication overhead. A load of a node depends on the number of recursions required for the item it is responsible for. Therefore, the lower the support of an item, the higher the height of the conditional FP tree constructed from that item, the more the iterations required, the higher the load, and the more the time and the space required for mining.

4. Analysis of Results

4.1. Performance Results of the Hybrid Attribute Feature Clustering Algorithm under Spark. The experiments are repeated 10 times on $D2\_1$, $D2\_2$, $D2\_3$, $D2\_4$ datasets constructed by randomly extracting 20%, 40%, 60%, 80% data from the D2 dataset, and the results are shown in Figure 5.

From the figure, the mining time required by the improved Opt-SFPG algorithm decreases gradually with the increase in the number of computational nodes for processing the same size of data, which is following the expected results. Further, from the analysis of different data sizes, the vertical execution time difference on the four datasets becomes smaller and smaller as the number of computational nodes increases. From this analysis, the improved algorithm can show better mining performance when mining large-scale datasets.

From the results in the figure, first the speedup ratios of the six algorithms, along with the increase in the number of experimental computational nodes, basically increase linearly. This is because the more computational nodes there are, the fewer tasks will be divided equally among individual nodes for the same number of tasks, the more memory will be available to store data, and the faster processing of each task will be. This also reflects the effectiveness of the above

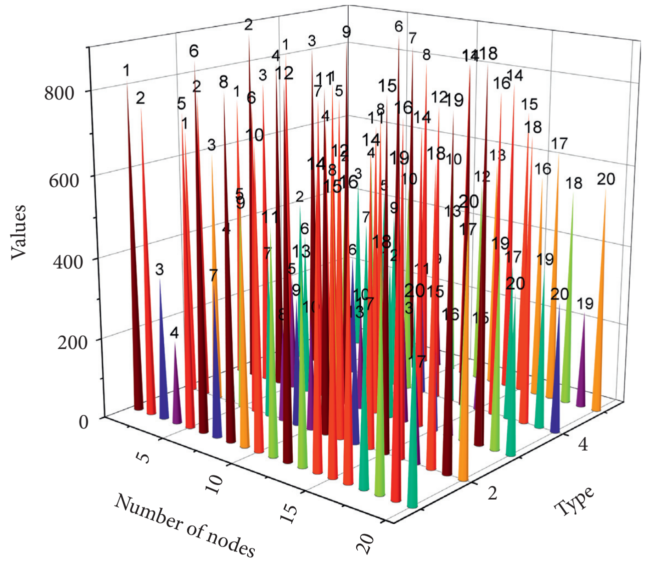


FIGURE 5: Algorithm execution time.

six parallel algorithms. Secondly, from another point of view, the increase in the number of nodes leads to a trend of increasing vertical distance between the algorithms, which is evident in the figure between the optimized algorithm Opt-SFPG and the traditional algorithm SFPG.

It can be analyzed that the Opt-SFPG algorithm based on the dual optimization of item header table and item set grouping is more effective in mining than other algorithms. When the number of nodes is increased from 1 to 5, the average acceleration ratio of the Opt-SFPG algorithm based on the smaller dataset $D1$ is 0.216 and that of the SFPG algorithm is 0.102, while the average acceleration ratio of the Opt-SFPG algorithm based on the larger dataset $D2$ is 1.282 and that of the SFPG algorithm is 0.644. The parallel performance of the optimization algorithm is better when the dataset is larger. The parallel performance of the optimization algorithm is better. When the values are 0.4, 0.35, 0.3, and 0.25, respectively, the mining time results of the FPG algorithm and the SFPG algorithm are on the mushroom dataset; Figure 5 shows that when the min_Support value is 0.6, 0.55, 0.5, and 0.45, respectively, the FPG algorithm and the SFPG algorithm are in the accidents mining time results on the dataset.

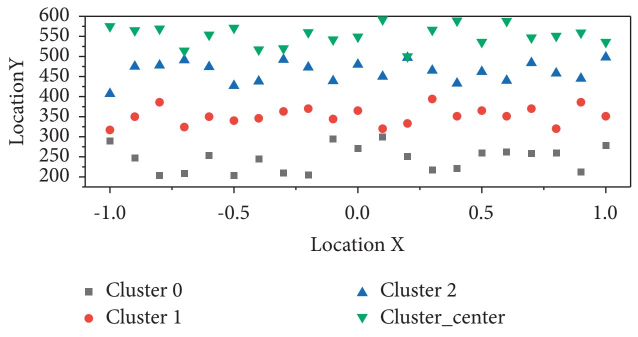
In the case of changing from only 1 compute node to 2 compute nodes, the mining time required by the Opt-SFPG algorithm is directly reduced by about a factor of 1, and the rate of reduction is significantly higher than that of the other five algorithms. Moreover, the mining time of the Opt-SFPG algorithm is the least in all node cases. In terms of the mining time required and the decrease rate of mining time with increasing nodes, the Eq-SFPG algorithm and the Ht-SFPG algorithm are almost equal in the $D1$ dataset; however, in the larger dataset $D2$, especially when the number of nodes increases from 2 to 4, the mining time of the Eq-SFPG algorithm is significantly less than that of the Ht-SFPG algorithm, which indicates that in the case of mining large datasets, the improved node balancing load scheme plays a more important role in the optimization algorithm at the node count level.

The accuracy of the experimental results and the squared K-means cost of the algorithm is shown in Figure 6. The clustering accuracy of the improved algorithm is significantly higher than that of the original K-means algorithm while the squared K-means cost is lower than that of the improved algorithm, which is analyzed because the K-means++ method based on holdout validation can yield the K-value that best matches the true distribution of the data, which can effectively improve the accuracy of the clustering results, and because the determination of the optimal value reduces the number of iterations of the algorithm, thus reducing the squared K-mean cost. The average speedup rate of the Opt-SFPG algorithm is 0.216, and that of the SFPG algorithm is 0.102, while the average speedup rate of the Opt-SFPG algorithm based on the large-scale dataset D2 mining is 1.282; the SFPG algorithm is 0.644.

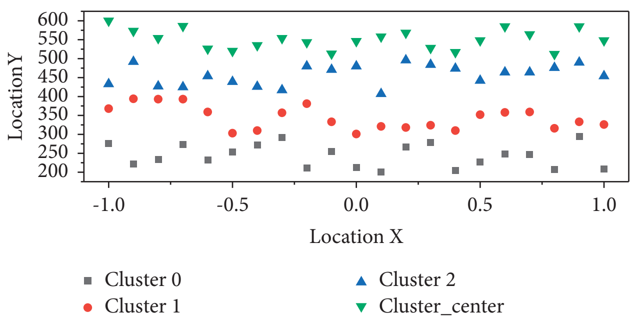
The original DPC algorithm does not divide the sample set into 3 equal parts well, while the CDPC-KNN algorithm gives more satisfactory results for the dataset, and the DBSCAN algorithm divides the dataset into 4 class clusters that do not match the real number of class clusters. The Abalone dataset contains 4177 datasets and 3 class clusters, and for such a dataset with many samples and a complex distribution, the DPC algorithm has more deviations from the real number of class clusters. The clustering effect of the DBSCAN algorithm deviates from the actual one and divides the dataset into more class clusters. DPC algorithm can correctly identify the number of class clusters, but the accuracy is poor, and the clustering result of the CDPC-KNN algorithm on the Abalone dataset is significantly better than the previous two algorithms.

As shown in Figure 6, all three algorithms can correctly classify the dataset. The R15 dataset has more sample points and class clusters than the Spiral dataset, and the CDPC-KNN algorithm and the DPC algorithm both have better clustering results with 99.7% accuracy, while the DBSCAN algorithm has 77.5% accuracy. The D31 dataset has the most 31 class clusters, and the CDPC-KNN algorithm has 94.3% accuracy, which is slightly higher than the 93% accuracy of the DPC algorithm and the 82.6% accuracy of the DBSCAN algorithm. The accuracy of the CDPC-KNN algorithm is 94.3%, which is slightly higher than that of the DPC algorithm (93%) and that of the DBSCAN algorithm (82.6%). The accuracy of the DBSCAN algorithm is 67%.

4.2. Experimental Analysis Results of the Feature Large Dataset. To make the CDPC-KNN clustering algorithm in the local clustering stage each computational node can be clustered independently on the corresponding data partition, the data partitioning stage divides the dataset into several groups of overlapping data partitions, as shown in Figure 7, each partition contains internal points and boundary points, the internal points are the high-density data in the middle space of the partition, and the boundary points are the points distributed on the edges of the partition. The boundary points are points distributed on the edges of the partition, and the boundary points may also be extension points of adjacent partitions; i.e., the boundary



(a)



(b)

FIGURE 6: Clustering results of user data.

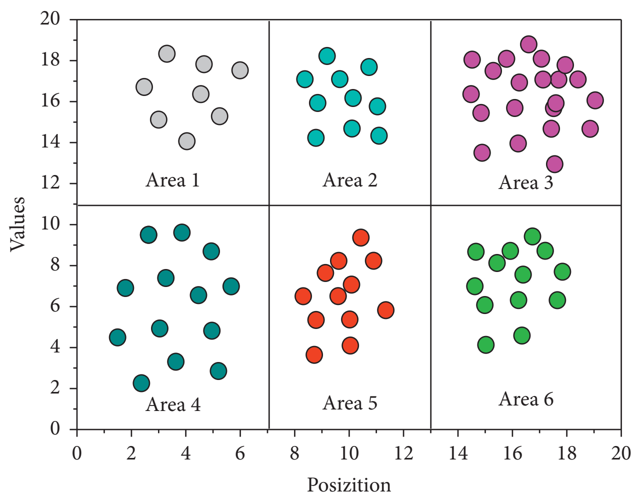


FIGURE 7: Results of the division of the dataset.

points of two adjacent partitions belong to a global class cluster. These overlapping partitions contain some common data objects within each other. In the local clustering merging phase, the algorithm can identify all the local clusters to be merged by evaluating the characteristics of these common data objects (i.e., boundary points and extension points).

Let point q be the sample point where two slices overlap; this point must be the boundary point of the slice where it is located, and the local class cluster belonging to slice 1 is $C1$ and the class cluster belonging to slice 2 is $C2$, and in both $C1$ and $C2$ are core points, i.e., points with large local density; then in the merging process, it is necessary to merge local class clusters $C1$ and $C2$ into global class clusters. Because the merging of class clusters is performed by boundary points, the boundary points of one slice are the extension points of the adjacent slice, as shown in Figure 7.

The Broadcast global broadcast provided by Spark shares the external variables among all nodes' memory, making it possible to keep one copy of the global variables in each Executor, avoiding the need to store a copy of the external variables at each task execution, thus reducing the number of variable copies in the cluster and reducing the memory overhead of the Executor. The average speedup rate of the Opt-SFPG algorithm is 0.216, and that of the SFPG algorithm is 0.102, while the average speedup rate of the Opt-SFPG algorithm based on the large-scale dataset $D2$ mining is 1.282; the SFPG algorithm is 0.644. In the clustering and merging phase of this parallelization experiment, a dictionary of labels is used as a read-only broadcast variable to be used in the relabeling process.

The parallelized execution of the Spark-based DBSCAN algorithm involves many RDD objects conversion and storage, node communication, and data transmission, which increases the cost of running the algorithm to a certain extent compared with the standalone serialization scheme. In the experiments, the parallelization algorithm under the Spark platform is tuned according to the above optimization ideas by combining the characteristics of the Spark platform and practical experience, to further improve the parallel efficiency of the algorithm by taking advantage of the platform.

When the data volume is small, the advantage of cluster computing is small, and the acceleration ratio always rises slowly around 1 for different worker numbers, because the current data volume is much smaller than the data size that the cluster is suitable to process, and coupled with the cluster scheduling and data communication consumption between cluster nodes, the advantage of parallelized cluster operation compared to the standalone case cannot be observed, and the parallel efficiency improvement is not satisfactory. However, as the data volume increases, the speedup ratio exhibits a linear increase with the number of nodes. The maximum speedup ratio reaches 3.3 times that of the standalone operation when the cluster nodes are fully opened, which fully demonstrates that the parallelized LAPO-DBSCAN algorithm proposed in this paper greatly improves the operation efficiency of the algorithm under large-scale datasets.

5. Conclusion

This paper implements the parallelization operation of the LAPO-DBSCAN algorithm under the Spark platform. A Spark on Yarn distributed cluster experimental environment is built, and then a parallelized clustering study is conducted in this distributed environment to realize the parallelization of the LAPO-DBSCAN algorithm, and the parallel algorithm optimization ideas in the Spark environment are summarized with practical experience. Finally, experiments are designed, and the results show that the algorithm has the same clustering effect in standalone and distributed environments, and the execution efficiency of the parallelized LAPO-DBSCAN algorithm in the cluster environment is greatly improved than that in the standalone environment when the data scale is larger. By analyzing the dichotomous method, we find that the privacy budget is consumed too fast and propose the privacy budget allocation AST scheme using the trichotomous method combined with the differential privacy K-means algorithm with equal difference series. This method provides a larger privacy budget in the early iterative stage to facilitate faster convergence of the algorithm and ensures that the center of mass is not distorted by the introduced random noise in the later stage. We build a Spark cluster environment and complete the parallelization improvement of the improved *K*-means algorithm and CDPC-KNN algorithm on the Spark platform and conduct the comparison experiments of the improved K-means algorithm and CDPC-KNN algorithm in the serial-parallel environment with network data, respectively. The experimental results show that the parallelized algorithms have more powerful data processing ability than the serial algorithm and can better meet the needs of current large data clustering analysis. Multiview mapping clustering algorithm is based on coupled DNA-GA population P-system. A new multiview consistent clustering algorithm based on KNN and graph ideas is proposed. This multiview mapping clustering process is done in the coupled DNA-GA population P-system according to specific rule requirements, and the great parallelism of the system can further improve the operational efficiency of the algorithm.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by School of Information Engineering, Chengdu College of Arts and Sciences.

References

[1] J. Wen, J. Yang, and B. Jiang, "Big data driven marine environment information forecasting: a time series prediction network," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 1, pp. 4–18, 2020.

[2] D. Xia, F. Ning, and W. He, "Research on parallel adaptive canopy-K-means clustering algorithm for big data mining based on cloud platform," *Journal of Grid Computing*, vol. 18, no. 2, pp. 263–273, 2020.

[3] S. M. Mujeeb, R. P. Sam, and K. Madhavi, "An empirical study of the big data classification methodologies," *International Journal of Bioinformatics Research and Applications*, vol. 16, no. 2, pp. 195–215, 2020.

[4] J. Yuan, "An anomaly data mining method for mass sensor networks using improved PSO algorithm based on spark parallel framework," *Journal of Grid Computing*, vol. 18, no. 2, pp. 251–261, 2020.

[5] W Ding, C. T Lin, and Z Cao, "Shared nearest-neighbor quantum game-based attribute reduction with hierarchical coevolutionary spark and its application in consistent segmentation of neonatal cerebral cortical surfaces," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2013–2027, 2018.

[6] J. M. Luna-Romera, J. García-Gutiérrez, M. Martínez-Ballesteros, and J. C. Riquelme Santos, "An approach to validity indices for clustering techniques in Big Data," *Progress in Artificial Intelligence*, vol. 7, no. 2, pp. 81–94, 2018.

[7] V. Ravuri and S. Vasundra, "Moth-flame optimization-bat optimization: map-reduce framework for big data clustering using the moth-flame bat optimization and sparse fuzzy C-means," *Big Data*, vol. 8, no. 3, pp. 203–217, 2020.

[8] S. N. Mighan and M. Kahani, "A novel scalable intrusion detection system based on deep learning," *International Journal of Information Security*, vol. 20, no. 3, pp. 387–403, 2021.

[9] J. Lei, T. Jin, and J. Hao, "Short-term load forecasting with clustering–regression model in distributed cluster," *Cluster Computing*, vol. 22, no. 4, Article ID 10163, 2019.

[10] A. Belhassena and H. Wang, "Trajectory big data processing based on frequent activity," *Tsinghua Science and Technology*, vol. 24, no. 3, pp. 317–332, 2019.

[11] V. Talasila, K Madhubabu, K. Madhubabu, M. Mahadasyam, N. Atchala, and L. Kande, "The prediction of diseases using rough set theory with recurrent neural network in big data analytics," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 5, pp. 10–18, 2020.

[12] T. R. Rao, P. Mitra, R. Bhatt, and A. Goswami, "The big data system, components, tools, and technologies: a survey," *Knowledge and Information Systems*, vol. 60, no. 3, pp. 1165–1245, 2019.

[13] J. Wang, Y. Yang, and T. Wang, "Big data service architecture: a survey," *Journal of Internet Technology*, vol. 21, no. 2, pp. 393–405, 2020.

[14] Z. Chelly Dagdia, C. Zarges, G. Beck, and M. Lebbah, "A scalable and effective rough set theory-based approach for big data pre-processing," *Knowledge and Information Systems*, vol. 62, no. 8, pp. 3321–3386, 2020.

[15] C. Valliyammai and R. Devakunchari, "Distributed and scalable Sybil identification based on nearest neighbour approximation using big data analysis techniques," *Cluster Computing*, vol. 22, no. 6, Article ID 14461, 2019.

[16] R. A. Hasan, R. A. I. Alhayali, N. D. Zaki, and A. H. Ali, "An adaptive clustering and classification algorithm for Twitter data streaming in Apache Spark," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 6, pp. 3086–3099, 2019.

[17] A. Mohamed, M. K. Najafabadi, Y. B. Wah, E. A. K. Zaman, and R. Maskat, "The state of the art and taxonomy of big data analytics: view from new big data framework," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 989–1037, 2020.

[18] C. Xu, L. Xu, and Y. Lu, "E-government recommendation algorithm based on probabilistic semantic cluster analysis in combination of improved collaborative filtering in big-data environment of government affairs," *Personal and Ubiquitous Computing*, vol. 23, no. 3, pp. 475–485, 2019.

[19] S. K. Hussin, Y. M. Omar, S. M. Abdelmageid, and M. I. Marie, "Traditional machine learning and big data analytics in virtual screening: a comparative study," *International Journal of Advanced Computer Research*, vol. 10, no. 47, pp. 72–88, 2020.

[20] D. Kılınç, "A spark-based big data analysis framework for real-time sentiment prediction on streaming data," *Software: Practice and Experience*, vol. 49, no. 9, pp. 1352–1364, 2019.