Journal of Network Intelligence ©2024 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **9**, Number **1**, February **2024**

Network Data Resource Recommendation Based on Spark and Fast Spectral Clustering Algorithms

An-Qing Zhu*
Management School
South China Business College Guangdong, University of Foreign Studies
Guangzhou 510000, P. R. China
204932@gwng.edu.cn

Hong-Yuan Li
School of Architectural Engineering
Guangzhou City Construction College
Guangzhou 510925, P. R. China
87651566@qq.com

Rui-Hui Wu
Faculty of Entrepreneurship and Business
Universiti Malaysia Kelantan
Kuantan 16100, Malaysia
a20e0276f@siswa.umk.edu.my

*Corresponding author: An-Qing Zhu

Received June 9, 2023, revised August 13, 2023, accepted October 23, 2023.

Abstract. *: Spectral clustering is widely used due to its excellent performance in non-linearly separable data. Therefore, how to integrate spectral clustering algorithms applied to network data resource recommendation are of great research value. However, due to the high time complexity and space complexity, spectral clustering requires a large amount of time to complete the clustering task when facing data with high dimensionality and certain sparsity. To address this problem, this work proposes a Spark-based fast spectral clustering algorithm. Firstly, the morphological similarity distance is used instead of Euclidean distance as the similarity measure to improve the clustering accuracy. Secondly, the local optimum problem caused by the improper selection of the initial clustering centre is improved by the Max-distince criterion. In addition, the KL scatter is used as the distance metric for clustering, and the information provided by the elements in the dataset is fully utilised to measure the interrelationship of different datasets and guide the clustering of data, which improves the problem of sparse data distribution to a certain extent. Finally, the proposed fast spectral clustering algorithm is implemented in the Spark distributed data processing framework. The experimental results show that the proposed fast spectral clustering algorithm has higher accuracy and coverage compared with other commonly used source recommendation algorithms, and is more adaptable in real-time recommendation of massive network data resources, obtaining higher quality and efficiency of clustering.*

Keywords: Big data; Parallel computing; Resource recommendation; Spectral clustering; Spark

1. **Introduction.**

The number and size of available data sets in the world is growing rapidly with the use of more and more mobile devices, sensors. According to the IDC report, global network data resources will grow exponentially between 2013 and 2020. In the context of today's big data, how to better utilise and process network data resources has become an important and difficult issue [1,2,3].

Cluster analysis, also known as cluster analysis, is a technique that is widely used for data mining analysis in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics [4,5]. Clustering is the process of dividing similar objects into different groups or more subsets by means of static classification, so that the member objects in the same subset all have some similar properties, and generally classifies data clustering as a kind of unsupervised learning [6,7]. Clustering analysis can be classified in various ways from different perspectives; on the one hand, data clustering algorithms can be divided into structural and decentralised algorithms. Structural algorithms use previously successfully used clusters to classify, while decentralised algorithms determine all classifications at once [8,9]. Structural algorithms can be computed in both directions from top to bottom or bottom to top [10]. Bottom-up algorithms start with each object as a separate classification and continuously fuse similar objects within it. Top-down algorithms, on the other hand, classify all objects as a whole and then gradually subdivide them into smaller ones [11,12]. Spectral clustering algorithms were first used for clustering images in image segmentation. Subsequently, methods based on spectral clustering algorithms have also been used in areas such as text clustering, social network analysis, and biological data analysis. In contrast to traditional algorithms such as K-means and hierarchical clustering, spectral clustering algorithms can handle arbitrarily shaped clusters and do not need to assume a particular form of data distribution. In addition, spectral clustering algorithms can use different similarity measures, such as Euclidean distance and cosine similarity. Spectral clustering algorithms are more stable, less sensitive to initial values and noise, and more accurate clustering results than traditional clustering algorithms.

However, spectral clustering algorithms have more obvious drawbacks in terms of running time. Since the time complexity of computing the similarity matrix is $O(N^2)$, the time overhead of the spectral clustering algorithm to compute the similarity matrix is large for large-scale datasets [13]. For high-dimensional data, spectral clustering algorithms need to be pre-processed using dimensionality reduction algorithms, which themselves consume a large amount of computational resources [14,15]. As a result, spectral clustering algorithms take longer to run on large-scale datasets, cannot handle super-large-scale datasets, and require careful selection of parameters and eigenvalue decomposition methods to obtain better clustering results. Over the years, there have been a number of improved big data spectral clustering algorithms that seek to reduce the time and quality of big data spectral clustering. However, due to the high time complexity and space complexity, spectral clustering requires a lot of time to complete the clustering task when facing data with high dimensionality and certain sparsity. Therefore, how to achieve high quality spectral clustering of big data in a faster time has become an important research topic.

Traditional serial spectral clustering algorithms are effective in handling small amounts of data, but are not feasible for large data [16], so we need to implement them on distributed computing frameworks and process them on computer clusters. Currently, distributed computing frameworks [17] are divided into three types of frameworks, batch processing framework, stream processing framework and hybrid processing framework, depending on the form of the processed data. The main objective of this work is to achieve high efficiency and high precision spectral clustering under the Spark distributed data processing framework, so as to solve the big data spectral clustering problem.

1.1. **Related Work.** Clustering is widely used in machine learning and data mining tasks, with spectral clustering being widely used in clustering tasks due to its good performance. However, spectral clustering requires $O(N^2)$ time complexity and space complexity to compute and store the Laplacian graph between data. Therefore, spectral clustering requires a lot of memory and time for big data tasks, which is difficult to implement on resource-limited machines, so speeding up and reducing memory requirements for spectral clustering turns out to be a very important research topic.

In the past decades, spectral clustering of big data has been a hot research topic at home and abroad, and many spectral clustering improvement algorithms have been proposed to achieve certain improvements in space complexity and time complexity.Wang et al. [18] used the K-means algorithm to select a number of columns from the $N \times N$ Laplacian graph, and then combined it with the Laplacian graph to construct a low-rank approximation matrix and perform eigen decomposition, and finally approximated by matrix multiplication to obtain the eigenvectors of N data. Wang et al. [19] transformed the Laplacian eigendecomposition problem into a weighted clustering problem by approximating the solution, which significantly reduced the eigendecomposition time complexity. Based on this, Ahmadi et al. [20] weighted only some of the data points and used the obtained cluster cores to classify the remaining data points, further reducing the time required.

Although the above algorithm significantly reduces the time required for Laplacian feature decomposition, the above spectral clustering algorithm still requires a significant amount of time to complete the clustering task if the data is of very high dimensionality and exhibits considerable sparsity. This is because traditional spectral clustering algorithms, consider more the distance between common items and ignore the information that may be embedded between non-common items. In information theory, KL scatter is used to measure the information loss incurred when one distribution is fitted to another. The use of KL scatter allows for the introduction of probability distributions from statistics, taking into account the overall distributional properties of the data and the information provided by the different data in the dataset.

Spark has attracted a lot of attention as the emerging and most widely used open source framework for big data processing. Heidari et al. [21] implemented the K-means algorithm using MapReduce and experimentally demonstrated that placing cluster heads with some minimal distance is better than placing cluster heads randomly. Mbyamm et al. [22] used Hash sampling function and MapReduce computing framework to process data clustering and improve the execution efficiency of clustering algorithms. Compared with Hadoop, Spark is designed to be memory-based from the beginning. Such a distributed framework can store the intermediate data processing results in memory, and there is no need to read HDFS data repeatedly for each iteration of data processing, which reduces the I/O load. At the same time, Spark-based clustering algorithm design based on Spark DAG task scheduling execution mechanism, better than the Hadoop MapReduce iterative execution mechanism.

1.2. **Motivation and contribution.** Based on the above analysis, a fast Spark-based spectral clustering algorithm is proposed in this work to improve the accuracy of clustering and the execution efficiency of the algorithm in the face of clustering tasks for datasets with sparsity and high-dimensional attributes.

The main innovations and contributions of this work include:

(1) Using morphological similarity distance instead of Euclidean distance as a similarity measure to improve the accuracy of spectral clustering, and improving the local optimum

problem caused by improper selection of initial clustering centres through the maximum-distince criterion;

(2) With the help of KL scatter as a distance indicator for spectral clustering, the information provided by the elements in the data set is fully utilized to measure the interrelationship of different data sets and guide the clustering of data, which improves the problem of sparsity of data distribution to a certain extent;

(3) The spectral clustering algorithm is optimised and implemented under the Spark parallel computing framework to improve the processing capability of massive data and significantly reduce the time required for clustering of large data.

2. Analysis of the relevant principles.

2.1. Principle of the spectral clustering algorithm.

Spectral clustering algorithms are derived from spectral graph theory. The standard spectral clustering algorithm divides the data by a Laplacian graph of similarity relations between them, so that the similarity within clusters is high and the similarity between clusters is low. Spectral clustering consists of two main parts: the construction of the Laplacian graph and the cutting of the Laplacian graph.

The Laplacian diagram is constructed mainly by first constructing the corresponding adjacency matrix, where w_{ij} represents the elements of the i-th row and j-th column of the adjacency matrix. It is important to note that this element also represents the similarity of the i-th data to the j-th data. The adjacency matrix is then normalised to obtain a Laplacian graph, and the algorithms for constructing the adjacency matrix are generally classified as e-neighbour, k-nearest neighbour and fully connected. Spectral clustering achieves clustering category delineation through the Laplace partitioning method, and the number of partitioned subsets is equal to the number of clustering categories. Spectral clustering requires solving for the vertex similarity and the eigenvalues of the partitioned subsets. Let the Laplacian graph $G = (V, E)$ contain a total of n vertices. The set of edge relations formed by all vertices is $E = \{e_{ij} = \langle v_i, v_j\rangle | v_i, v_j \in V\}$, where the degree of similarity between the vertex v_i and the vertex v_j is w_{ij}.

$$w_{ij} = \begin{cases} \exp(\frac{-d(v_i,v_j)^2}{\sigma^2}), e_{ij} \in E \\ 0, otherwise \end{cases} \tag{1}$$

where σ is a constant and $d(v_i, v_j)$ is the distance between two vertices.

The calculation of the distance value is related to the dimension chosen and the common method is the grey scale value. The similarity between all vertices is calculated to form the similarity weights W.

$$W(i,j) = \begin{cases} w_{ij}, e_{ij} \in E \\ 0, otherwise \end{cases} \tag{2}$$

Let the similarity of the i-th vertex in the Laplacian graph G to all other vertices be D_{ii}, which is calculated as shown below:

$$D_{ii} = \sum_j w_{ij} \tag{3}$$

The similarity of all vertices can be calculated from the above equation. The total similarity of all vertices in G can be constructed by combining multiple similarities D.

Assume that the non-normalized Laplace coefficient is L.

$$L = D - W \tag{4}$$

The normalized Laplace coefficient is L_s, and the transfer coefficient is L_r.

$$L_{\mathrm{s}} = 1 - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \tag{5}$$

$$L_{\mathrm{r}} = D^{-1} L = 1 - D^{-1} W \tag{6}$$

In spectral clustering, clustering accuracy is closely related to the partitioning scheme of the Laplacian graph, in addition to the need to focus on the similarity relationships between vertices.

Let the undirected graph have k connected subsets $A_1, A_2, \ldots, A_k$, then the cut set between connected subsets is as follows

$$cut(A_1, A_2, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A_i}) \tag{7}$$

There are more ways to derive cut-set solutions, and the normalised cut-set method is chosen for this work.

$$Ncut(A_1, A_2, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{W(A_i, \bar{A_i})}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A_i})}{vol(A_i)} \tag{8}$$

$$vol(A_i) = \sum_{v_i \in A_i} d_i \tag{9}$$

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{10}$$

Let the normalised cut set divide the graph G into the k categories $A_1, A_2, \ldots, A_k$. The calculation of the k categories is represented by the set $h_j = \{h_{1j}, h_{2j}, \ldots, h_{nj}\}$.

$$h_{ij} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}}, v_i \in A_j \\ 0, otherwise \end{cases} \tag{11}$$

Finally, we construct the set of categories as a subset $H = \{h_1, h_2, \ldots, h_k\}$. For the k categories, the normalised cut set can be optimised to the features corresponding to the k eigenvalues of L_s to obtain the features corresponding to L_r.

2.2. **Spark architecture.**

This work chooses Spark to implement the spectral clustering algorithm instead of MapReduce, mainly because Spark extends the MapReduce computational model while retaining the advantages of MapReduce. Compared with MapReduce, the intermediate results of SparkJob computation can be stored in the node's memory without the need to read data from HDFS or disk. The system architecture of Spark is shown in Figure 1.

The Spark platform provides for one master node and all other nodes are slaves. When a new slave node joins the Spark platform, it must first register with the master node. When the user submits a clustering task to the master node, the master node decomposes the task and sends it to the slave nodes, and gives the user feedback on the address of the assigned slave node. Based on the feedback, the user registers with the slave node and establishes a connection with the slave node. When the slave node loads the task, the user side submits the calculation results to the user. All node operations are implemented in Resilient Distributed Datasets (RDD), which improves data access efficiency and thus enhances clustering in real time.

3. **Fast spectral clustering algorithm.**

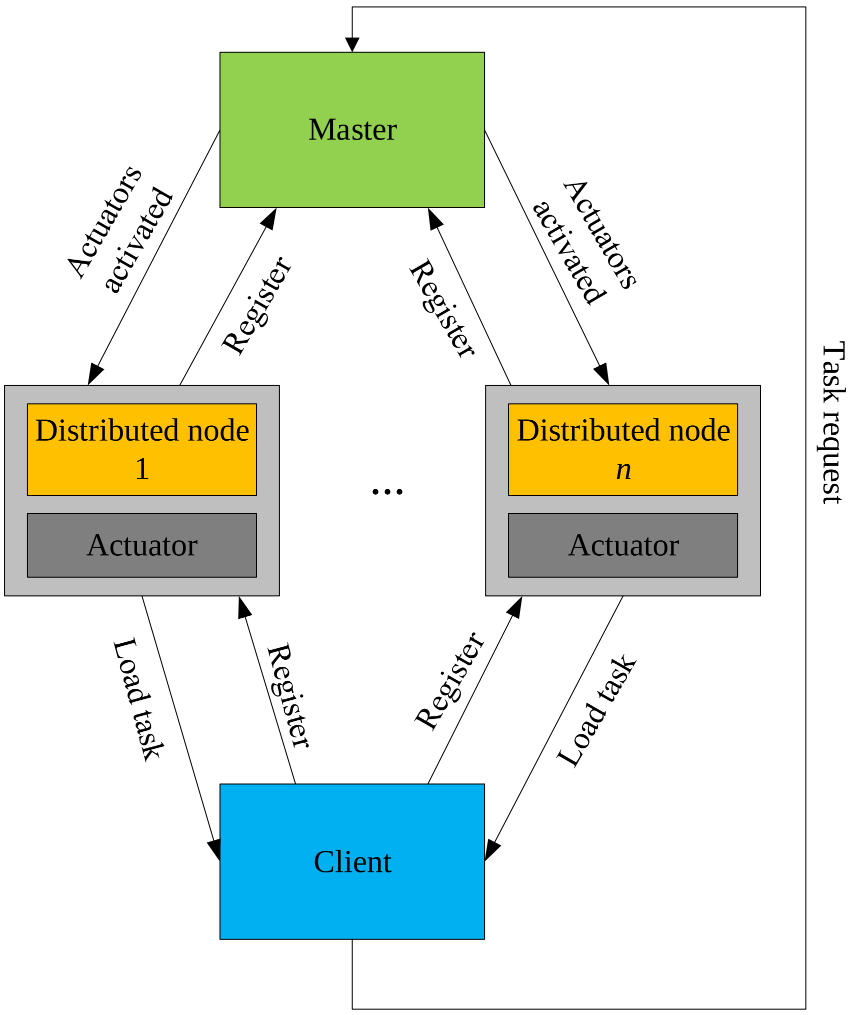

Figure 1. The System Framework for Spark

3.1. **Morphological similarity distances.** This work optimises the standard spectral clustering algorithm by using morphological similarity distances [23] and the maximum distance distance criterion to establish the location of clustering centres in relation to data points. The similarity measure is generally derived from the Minkowski distance and the main applications are the Euclidean distance and the Manhattan distance.

The morphological similarity distance D_{MSD} is calculated as shown below:

$$D_{MSD} = (1 - D/SAD) \times ED \tag{12}$$

$$D = \left| \sum_{i=1}^{m} (X_{ji} - Y_{qi}) \right| \tag{13}$$

where SAD denotes Manhattan Distance, ED denotes Euclidean Distance, X_{ji} denotes the i-th attribute value of variable j, Y_{qi} is the i-th attribute value of variable q, and m denotes the feature dimension.

An example of the calculation of distances between variables is shown in Table 1. It can be seen that if $D/ASD = 1$, then the morphological similarity distance D_{MSD} is the Euclidean distance.

Table 1. Example of distance calculation between variables

Feature difference	**ED**	**SAD**	**D**	**MSD**
(-1,3,-2)	3.74	6	6	7.48

3.2. **Max-distance criterion and fast localisation.** The Max-distance algorithm is an improvement of the maximum-minimum distance algorithm [24], which selects the initial cluster prime according to its maximum distance principle.

Firstly, a point near the boundary (not a boundary point) is selected as the cluster centre C_1 and C is defined as the set of cluster centres. Then, the data point furthest from the cluster center C_1 is selected as the second cluster center C_2. Second, the distances between the other data points and C_1 and C_2 are calculated, and the smallest of them is calculated.

$$d_{xy} = \left\| n_x - n_y \right\|, y = 1, 2 \tag{14}$$

$$d_x = \min\left[d_{x1}, d_{x2}\right], x = 1, 2, \ldots, n \tag{15}$$

$$d_l = max_x[\min[d_{x1}, d_{x2}]] > \theta.n_1 - n_2 \tag{16}$$

where θ is the specified parameter.

If there are k $(k! = K)$ cluster centres, calculate the distance from each sample point to the cluster centre of mass d_{xy} and d_l.

$$d_l = max_x[\min[d_{xy}, d_{zz}]] > \theta \cdot n_y - n_z \tag{17}$$

where $y! = z$ and $y, z \in C$.

To reduce the amount of redundant computation in the spectral clustering algorithm, the spatial location of data points in relation to the cluster centres is introduced. For any point in the initial data set, if its spatial location in relation to the cluster centre of mass is known, it is possible to determine which is the nearest cluster centre of mass, so that there is no need for multiple calculations and the data point is simply assigned to the appropriate class cluster.

For an arbitrary data point $s(x, y, z)$ in the space-rectangular coordinate system. Assuming that the maximum value of the dimension in which x is located is max$x$, the minimum value is min$x$, and the number of segments of the grid in this dimension is xNum. Assuming that the maximum value of the dimension in which y is located is max$y$, the minimum value is min$y$, and the number of segments of the grid in this dimension is yNum. Assuming that the maximum value of the dimension in which z is located is max$z$, the minimum value is min$z$, and the number of segments of the grid in this dimension is zNum. Based on the coordinates of the points in the dataset, the grid location of point s can be quickly located at (x', y', z'), thus effectively reducing the computational effort between point s and the cluster centre of mass.

$$x' = \frac{x - minx}{maxx - minx + \alpha} \cdot xNum \tag{18}$$

$$y' = \frac{y - miny}{maxy - miny + \alpha} \cdot yNum \tag{19}$$

$$z' = \frac{z - minz}{maxz - minz + \alpha} \cdot zNum \tag{20}$$

where α is a positive decimal.

3.3. **Characterization based on KL scatter.** KL scatter belongs to relative entropy. Let $P(x)$ and $Q(x)$ be two probability probability distributions for the values taken by X. The relative entropy is shown as follow

$$D(P||Q) = \sum P(x) \log \frac{P(\ x\ )}{Q(\ x\ )} \tag{21}$$

The outstanding feature of relative entropy is that it is non-symmetric, i.e. $D(P\|Q)$ and $D(Q\|P)$ are not equal. However, both represent the distance between P and Q.

Assume that the data to be clustered is $X = \{X_1, X_2, ..., X_{n-1}, X_n\}$, N denotes the total number of samples, M denotes the total number of sample attributes, V denotes the total number of non-empty samples and K denotes the sparsity of the sample set.

$$K = \frac{V}{MN} \times 100\% \tag{22}$$

Typically, when the K value is under 5 %, these type of information sets can be categorized as sparse. The basic distribution of spatial data sets is shown in Table 2.

Table 2. Basic distribution of spatial data sets

X	x				
	1	2	...	m-1	m
X_1	X_{11}	X_{12}	...	$X_{1(m-1)}$	X_{1m}
X_2	X_{21}	X_{22}	...	$X_{2(m-1)}$	X_{2m}
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$
X_n	X_{n1}	X_{n2}	...	$X_{n(m-1)}$	X_{nm}

The spectral clustering process is divided into a process based on the formation of a probability matrix, a process based on the formation of a distance matrix and a circular iterative process. The probability matrix formation process generates a probability matrix of $n \times k$.

$$\mathbf{P}_{\mathrm{M}} = \begin{bmatrix} P_{11} & \cdots & P_{1k} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nk} \end{bmatrix} \tag{23}$$

Based on the probability matrix above, the KL distances between any row in the matrix and any other row are calculated separately to form the KL matrix on the overall data set. As also discussed in the introduction to KL scatter, KL has an asymmetric nature and the average of the calculated distances between any two rows from each other is used here as the actual KL value. The remainder is filled with zeros, resulting in the upper triangular probability matrix shown below:

$$\mathrm{DM} = \begin{bmatrix} 0 & \mathrm{DM}_{12} & \cdots & \mathrm{DM}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{24}$$

With the KL matrix formed, the two rows of data represented by values less than a certain precision in the distance matrix are merged to form a new data set. For the merged data set, a new probability matrix and distance matrix are formed according to the above process, and the merging of data is completed again. The clustering of all the data sets is completed by repeating the process until the K matrix fails to achieve the accuracy of the merged data.

4. Parallelization and implementation of Spark-based fast spectral clustering algorithm.

4.1. Implementation of morphological similarity distances. The parallel computing framework for big data is shown in Figure 2.

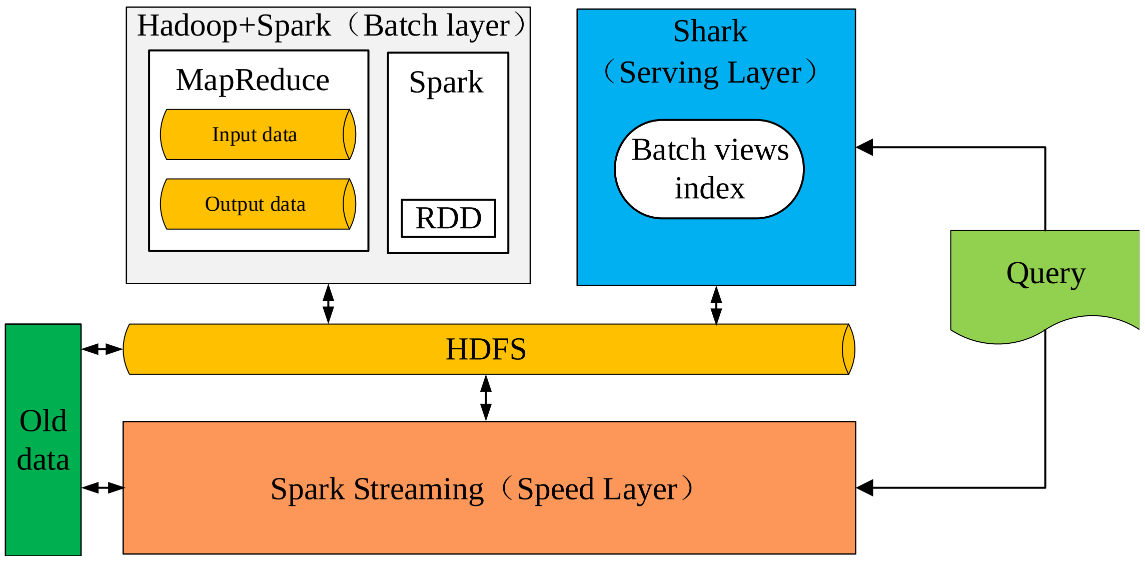

Figure 2. Parallel computing framework for big data

Based on a parallel computing framework for big data, the proposed fast spectral clustering algorithm uses morphological similarity distances instead of Euclidean distances. The key Java pseudo-code for computing the morphological similarity distance between two specific points is shown in Algorithm 1.

Algorithm 1 Pseudo-code of morphological similarity distance between two points

```
1: public static Double distance(Point a, Point b)
2: // Calculate the morphological similarity distance between two points
3: double sum = 0.0
4: for (int i = 0; i < a.alsize(); i++) do
5:     // Each dimension is subtracted to take the absolute value and then summed
6:     sum += Math.abs(a.alget(i) - b.alget(i))
7: end for
8: return sum
```

4.2. Implementation on Spark. The data point furthest away from the known center of mass is selected each time when picking the remaining initial center of mass. The idea of the algorithm for selecting the point furthest from the selected centre of mass for each iteration is as follows:

(1) Loop to find the remaining k-1 initial points.

(2) Calculate and record the minimum distance between the data point and the centre of mass, and output ¡minimum distance, data point> key-value pairs.

(3) The above key-value pairs are arranged in reverse order so that the maximum distance is at the top.

The key Java pseudo-code for implementing the fast spectral clustering algorithm on Spark is shown in Algorithm 2.

Algorithm 2 Parallelization of Spark-based fast spectral clustering algorithms

```
1: JavaPairRDD< Double, String > data distance center = data.mapToPair((
2:      (PairFunction< String, Double, String >)p→{
3:      Point data point = Point.StringToPoint(p).
4:      List< String > center_data= init_center_broadcast.value().
5:      double min_distance = Double.MAX_VALUE.
6: for String s : center data do
7:     Point center = Point.StringToPoint(s).
8:     double distance = Point.distance_paradigm(data_point center).
9:     if min_distance> distance then
10:        min_distance = distance.
11:    end if
12: end for
13:     return new Tuple2<>(min_distance p).
14:     })
15: // Sorted by minimum distance
16: JavaPairRDD< Double, String > distance_order=data_distance_center.sortByKey(false)
17: String new center = distance_order.take(1).get(0).
```

5. Experimental results and analysis.

5.1. Experimental data and environment.

5.1. Experimental data and environment. To validate the performance of the fast spectral clustering algorithm for online learning resource recommendation under the Spark platform, performance simulations were performed on the common 4-class public network dataset in Table 3. The experimental environment consists of five virtual machines, including a Master master node, which is responsible for the operation and management of the driver. The remaining four Worker slave nodes act as execution nodes.

The experiments were run on the following hardware environment: 64G RAM, 2T hard disk, 40 Intel Xeon(R) SSilver4114 CPUs at 2.20GHz. software environment: Ubuntu 18.04.6LTS, JDK1.8.0311, Spark-2.4.8-bin-hadoop2.7. Firstly, validate the fast spectral clustering algorithm's clustering performance for different samples. Secondly, resource recommendation is performed using stand-alone spectral clustering and fast spectral clustering algorithms under Spark platform respectively. Finally, the performance of the proposed fast spectral clustering algorithm is compared with the commonly used resource recommendation algorithms.

Table 3. Data sets

Data set name	Number of samples	Number of users	Number of resources
Goodbooks-10k	860214	44647	13333
Mooc	414443	85868	4635
CiteULike	230318	8884	20313
MOOPer	256585	50077	8483

5.2. Network resource recommendation performance. Fast spectral clustering of the above four classes of network data resources using a single node, and their clustering performance is shown in Table 4. It can be seen that the accuracy of the spectral clustering performance is high for all four classes of datasets, with the highest accuracy of 0.7791 in the MOOPer set, and 0.7629 even in the lowest accuracy Goodbooks-10k

Table 4. Clustering Performance Analysis

Data sets	**Accuracy**	**AUC**	**Clustering time/s**
Goodbooks-10k	0.7629	0.5877	185.2578
Mooc	0.765	0.579	146.1038
CiteULike	0.7724	0.6493	137.6618
MOOPer	0.7791	0.6628	140.7928

dataset, indicating that the proposed fast spectral clustering has a high degree of clustering adaptation for the four classes of network data samples. In terms of AUC, the AUCs of all four data sets exceeded 0.58, indicating that fast spectral clustering has a high degree of aggregation for network data resources. In terms of clustering time, it took 185.2578 s on the Goodbooks-10k set and 137.6618 s on the CiteULike set. The reason for the large time difference between the two may be due to the different number of sample sets involved in clustering, with the Goodbooks-10k set having a significantly larger sample size than the CiteULike set.

After obtaining the clustering results, resource recommendations need to be made according to the categories and the results are as shown in Table 5. It can be seen that the

Table 5. Accuracy and coverage of resource recommendations

Data sets	**Accuracy**	**Coverage**
Goodbooks-10k	0.8026	0.8111
Mooc	0.8073	0.8232
CiteULike	0.8141	0.8193
MOOPer	0.8214	0.8282

accuracy of the recommendations for all four sample sets is higher than 0.80, while the coverage rate is higher than 0.81. Therefore, the proportion of resources accepted by users for all recommended resources is up to more than 80%. At the same time, the proportion of users using the recommended resources versus actually using all resources reached over 81%, which indicates that the performance of network data resource recommendations has been significantly improved through spectral clustering.

5.3. **Efficiency improvement of Spark for fast spectral clustering.** To verify the extent of Spark platform's impact on recommendation efficiency, the single-node recommendation time T_1 and Spark multi-node recommendation time T_2 were solved with differential settings of the number of Spark nodes, and the speedup ratio $K = \frac{T_1}{T_2}$ was calculated as shown in Table 6. It can be seen that for the four types of network data sample sets, the recommendation efficiency is significantly improved after Spark parallel computing. The speedup ratios for the Goodbooks-10k set, Mooc set, CiteULike set and MOOPer set were 40.833, 28.183, 26.845 and 36.885 respectively when the number of nodes was 7. This indicates that the time required by the fast spectral clustering algorithm was significantly reduced by the Spark platform. The parallel computing framework of Spark has a significant impact on the recommendation efficiency of the clustering algorithm.

5.4. **Recommendation performance of different algorithms.** Collaborative filtering [26], standard spectral clustering [27], multiview spectral clustering [28] and fast

Table 6. Spark Acceleration Ratio

Data sets	**Number of nodes**	***K***
Goodbooks-10k	1	3.177
	4	20.507
	7	40.833
Mooc	1	2.89
	4	16.769
	7	28.183
CiteULike	1	2.98
	4	18.629
	7	26.845
MOOPer	1	3.242
	4	23.443
	7	36.885

spectral clustering were used for network data resource recommendation simulations respectively. To eliminate the effect of Spark parallel computing framework, all four recommendation algorithms were done on a single machine and their recommendation accuracy is shown in Figure 3, Figure 4, Figure 5 and Figure 6.

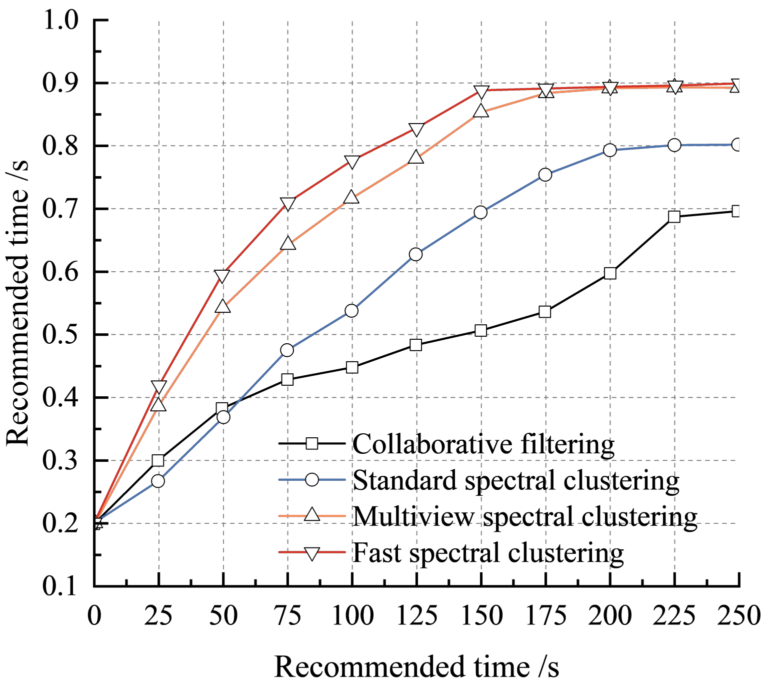

Figure 3. Goodbooks-10k

It can be seen that for all four datasets, the collaborative filtering recommendation algorithm has the lowest recommendation accuracy and the proposed fast spectral clustering algorithm has the highest recommendation accuracy. In the Goodbooks-10k set, CiteULike set and MOOPer set, the accuracy of Multiview spectral clustering and the fast algorithm are very close, and the standard spectral clustering is slightly inferior to these two algorithms. In the Mooc set, the recommendation accuracy of the proposed fast spectral clustering algorithm is significantly better. In terms of recommendation time,

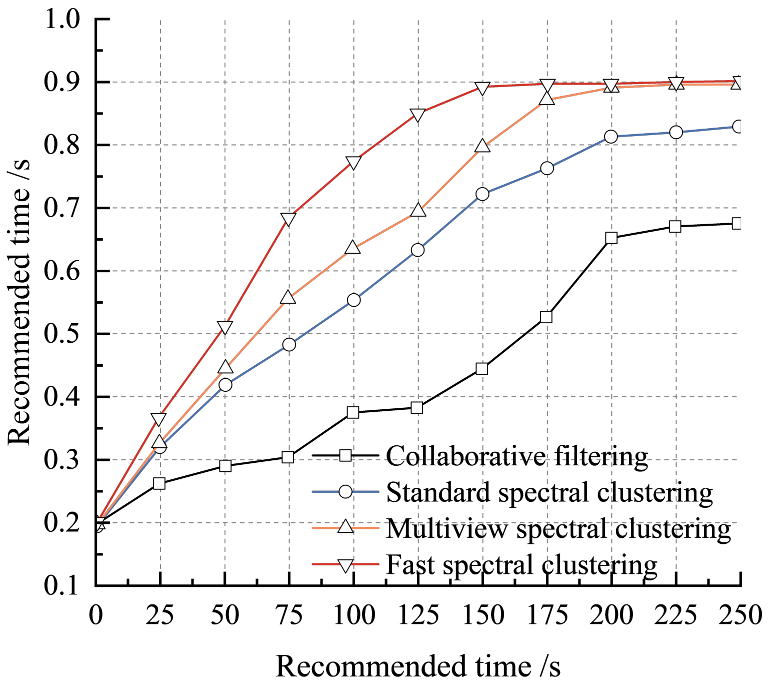

Figure 4. Mooc

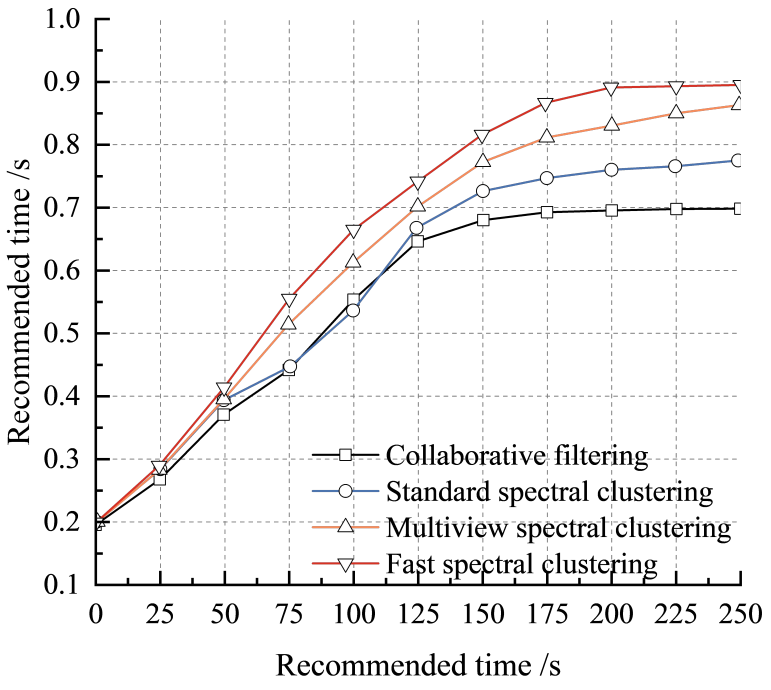

Figure 5. CiteULike

the collaborative filtering algorithm takes the shortest time and the proposed fast spectral clustering algorithm takes the second shortest time. However, the standard spectral clustering and Multiview spectral clustering are the most time consuming because the time complexity and space complexity of spectral clustering are high when dealing with data with high dimensionality and some sparsity, thus requiring a large amount of time to complete the clustering task.

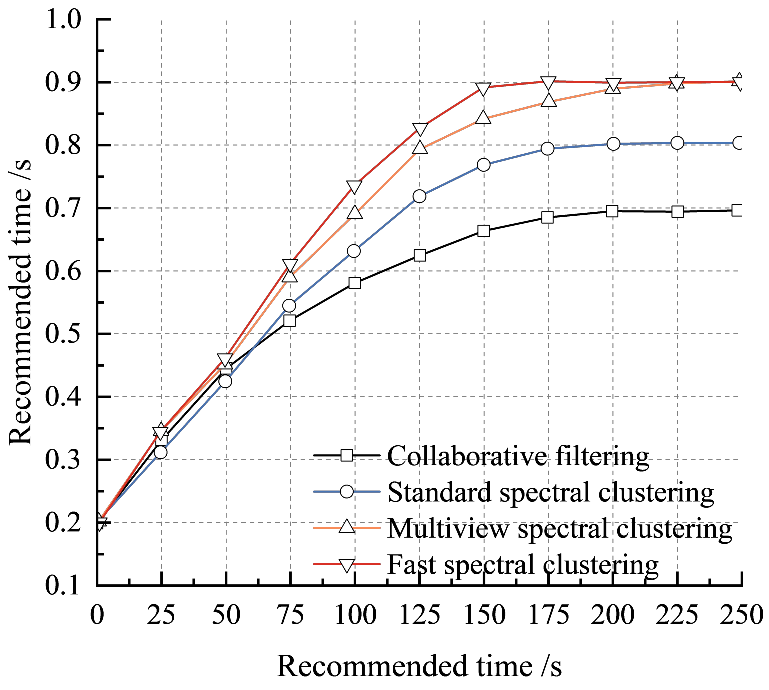

Figure 6. MOOPer

To further compare the recommendation performance, the four algorithms were implemented on a Spark platform with a node count of 5 and the recommendation coverage values were calculated and the results are shown in Table 7.

Table 7. Recommended coverage and recommended times

Data sets	**Spark-based parallelization algorithms**	**Coverage**	**Recommended time/s**
Goodbooks-10k set	Collaborative filtering	0.6287	5.9939
	Standard Spectral Clustering	0.7396	10.3389
	Multiview spectral clustering	0.7508	10.5699
	Fast spectral clustering	0.7751	9.4579
Mooc set	Collaborative filtering	0.6308	5.4029
	Standard Spectral Clustering	0.7536	9.8499
	Multiview spectral clustering	0.7722	9.7839
	Fast spectral clustering	0.784	9.2439
CiteULike Collection	Collaborative filtering	0.6661	4.8949
	Standard Spectral Clustering	0.7593	8.2879
	Multiview spectral clustering	0.777	8.4959
	Fast spectral clustering	0.7833	7.7639
MOOPer Collection	Collaborative filtering	0.6682	3.9519
	Standard Spectrum Clustering	0.7631	7.1379
	Multiview spectral clustering	0.7836	7.6139
	Fast spectral clustering	0.7922	6.2009

It can be seen that the performance of different algorithms varies significantly for the recommendation coverage of the four types of sample sets, with fast spectral clustering having the highest coverage and collaborative filtering having the lowest coverage. After using Spark acceleration with five nodes, the recommendation efficiency of all four algorithms improved significantly compared to the recommendation time of a single node, so the addition of the Spark platform can effectively improve the recommendation efficiency for these four types of recommendation algorithms when dealing with massive network data resources.

6. **Conclusion.** In this work, a Spark-based fast spectral clustering algorithm was proposed for solving the problem of recommending massive network data resources. Morphological similarity distance is used instead of Euclidean distance as the similarity measure to improve the clustering accuracy. Secondly, the Max-distince criterion is used to improve the local optimum problem caused by the improper selection of initial clustering centres. In addition, the KL scatter is used as a distance metric for clustering, which makes full use of the information provided by the elements in the dataset to measure the interrelationship of different datasets and guide the clustering of data, improving the problem of sparse data distribution to a certain extent. In the clustering process, parallel computation of similarity and feature vectors is achieved with the help of Spark platform to improve the efficiency of spectral clustering. Follow-up research will try to construct feature values using different cut-set methods to verify the performance of the spectral clustering algorithm for network data resource recommendation under different cut-set methods, and also optimise the feature point clustering method for spectral clustering to further improve the adaptability of the spectral clustering algorithm in network data recommendation applications.

Acknowledgement. This work supported by project of department of education of Guangdong province (No. 2022WZJD013), and Department of Education of Guangdong Province: Research team on integration of data and reality and management innovation (No. 2023WCXTD023).

REFERENCES

[1] J. Yang, Y. Li, Q. Liu, L. Li, A. Feng, T. Wang, S. Zheng, A. Xu, and J. Lyu, "Brief introduction of medical database and data mining technology in big data era," *Journal of Evidence-Based Medicine*, vol. 13, no. 1, pp. 57-69, 2020.

[2] C.-M. Chen, L. Chen, W. Gan, L. Qiu, and W. Ding, "Discovering high utility-occupancy patterns from uncertain data," *Information Sciences*, vol. 546, pp. 1208-1229, Feb. 2021.

[3] W. Gan, L. Chen, S. Wan, J. Chen, and C.-M. Chen, "Anomaly Rule Detection in Sequence Data," *IEEE Transactions on Knowledge and Data Engineering*, 2022, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9665277.

[4] W. Haoxiang, and S. Smys, "Big data analysis and perturbation using data mining algorithm," *Journal of Soft Computing Paradigm (JSCP)*, vol. 3, no. 01, pp. 19-28, 2021.

[5] V. Plotnikova, M. Dumas, and F. Milani, "Adaptations of data mining methodologies: a systematic literature review," *PeerJ Computer Science*, vol. 6, e267, 2020.

[6] Z. S. Ageed, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, H. S. Yahia, M. R. Mahmood, and I. M. Ibrahim, "Comprehensive survey of big data mining approaches in cloud systems," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 29-38, 2021.

[7] T.-Y. Wu, J. Lin, Y. Zhang, and C.-H. Chen, "A Grid-Based Swarm Intelligence Algorithm for Privacy-Preserving Data Mining," *Applied Sciences*, vol. 9, no. 4, 774, 2019.

[8] T.-Y. Wu, J. C.-W. Lin, U. Yun, C.-H. Chen, G. Srivastava, and X. Lv, "An efficient algorithm for fuzzy frequent itemset mining," *Journal of Intelligent and Fuzzy Systems*, vol. 38, no. 5, pp. 5787-5797, 2020.

[9] F. Zhang, T.-Y. Wu, Y. Wang, R. Xiong, G. Ding, P. Mei, and L. Liu, "Application of Quantum Genetic Optimization of LVQ Neural Network in Smart City Traffic Network Prediction," *IEEE Access*, vol. 8, pp. 104555-104564, 2020.

[10] F. Martínez-Plumed, L. Contreras-Ochando, C. Ferri, J. Hernández-Orallo, M. Kull, N. Lachiche, M. J. Ramirez-Quintana, and P. Flach. "CRISP-DM twenty years later: from data mining processes to data science trajectories," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 8, pp. 3048-3061, Aug. 2019.

[11] H. Thakkar, V. Shah, H. Yagnik, and M. Shah, "Comparative anatomization of data mining and fuzzy logic techniques used in diabetes prognosis," *Clinical EHealth*, vol. 4, pp. 12-23, 2021.

[12] K. G. Al-Hashedi, and P. Magalingam, "Financial fraud detection applying data mining techniques: a comprehensive review from 2009 to 2019," *Computer Science Review*, vol. 40, 100402, 2021.

[13] F. E. Bock, R. C. Aydin, C. J. Cyron, N. Huber, S. R. Kalidindi, and B. Klusemann, "A review of the application of machine learning and data mining approaches in continuum materials mechanics," *Frontiers in Materials*, vol. 6, 110, 2019.

[14] Y. Yun, D. Ma, and M. Yang, "Human-computer interaction-based decision support system with applications in data mining," *Future Generation Computer Systems*, vol. 114, pp. 285-289, 2021.

[15] G. Sun, Y. Cong, J. Dong, Y. Liu, Z. Ding, and H. Yu, "What and how: generalized lifelong spectral clustering via dual memory. " *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3895-3908, 2021.

[16] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering. " *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1212-1226, 2019.

[17] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning. " *ACM Computing Surveys*, vol. 53, no. 2, Article 33, 2020.

[18] J. Wang, J.-X. Liu, C.-H. Zheng, C.-H. Lu, L.-Y. Dai, and X.-Z. Kong, "Block-constraint laplacian-regularized low-rank representation and its application for cancer sample clustering based on integrated TCGA data," *Complexity*, vol. 2020, Article 13, 2020.

[19] Y. Wang, T. Li, L. Chen, G. Xu, J. Zhou, and C. P. Chen, "Random Fourier feature-based fuzzy clustering with p-Laplacian regularization ," *Applied Soft Computing*, vol. 111, 107724, 2021.

[20] M. Ahmadi, A. Taghavirashidizadeh, D. Javaheri, A. Masoumian, S. J. Ghoushchi, and Y. Pourasad, "DQRE-SCnet: a novel hybrid approach for selecting users in federated learning with deep-Q-reinforcement learning based on spectral clustering," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 9, pp. 7445-7458, 2022.

[21] S. Heidari, M. Alborzi, R. Radfar, M. A. Afsharkazemi, and A. Rajabzadeh Ghatari, "Big data clustering with varied density based on MapReduce," *Journal of Big Data*, vol. 6, pp. 1-16, 2019.

[22] M. J. Mbyamm, J. Zhang, and B. A. Kouassi, "MapReduce FCM clustering set algorithm," *Cluster Computing*, vol. 24, pp. 489-500, 2021.

[23] X. Kong, X. Zhao, C. Liu, Q. Li, D. Dong, and Y. Li, "Electricity theft detection in low-voltage stations based on similarity measure and DT-KSVM," *International Journal of Electrical Power & Energy Systems*, vol. 125, 106544, 2021.

[24] E. Carrillo, S. Yeotikar, S. Nayak, M. K. M. Jaffar, S. Azarm, J. W. Herrmann, M. Otte, and H. Xu, "Communication-aware multi-agent metareasoning for decentralized task allocation," *IEEE Access*, vol. 9, pp. 98712-98730, 2021.

[25] X. Yang, X. Yang, J. Yang, Q. Ming, W. Wang, Q. Tian, and J. Yan, "Learning high-precision bounding box for rotated object detection via kullback-leibler divergence," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18381-18394, 2021.

[26] M. Srifi, A. Oussous, A. Ait Lahcen, and S. Mouline, "Recommender systems based on collaborative filtering using review texts -a survey," *Information*, vol. 11, no. 6, 317, 2020.

[27] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting trajectory and behavior of road- agents using spectral clustering in graph-lstms," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4882-4890, 2020.

[28] Z. Kang, G. Shi, S. Huang, W. Chen, X. Pu, J. T. Zhou, and Z. Xu, "Multi-graph fusion for multi-view spectral clustering," *Knowledge-Based Systems*, vol. 189, pp. 105102, 2020.