Ingénierie des Systèmes d'Information
Vol. 29, No. 4, August, 2024, pp. 1297-1304
Journal homepage: http://iieta.org/journals/isi

Distributed K-means Clustering Using Topological Relationships

Zouaoui Guellil*, Nadir Mahammed, Nabil Keskes

LabRI-SBA Lab., Ecole Superieure en Informatique, Sidi Bel Abbes 22000, Algeria

Corresponding Author Email: z.guellil@esi-sba.dz

https://doi.org/10.18280/isi.290405

Received: 10 October 2023
Revised: 9 May 2024
Accepted: 30 May 2024
Available online: 21 August 2024

Keywords:
data mining, clustering, distributed k-means algorithm, number of groups, topological relations

ABSTRACT

Cluster analysis has been widely studied due to its importance and several methods have been developed for this purpose. However, these methods are designed to process on centralized data. In this paper, we present an asynchronous approach based on topological relationship. The proposed approach unfolds on three steps: First, each site searches for clusters (models) of its local data. Secondly, a central site proceeds to the analysis and search for the partition of the whole (the global model). Finally, we proceed with the search for the right number of groups of the global model. We note that each local data has their own number of clusters and it can be different from the number of clusters in the entire data. The experiments have clearly demonstrated the effectiveness of the proposed approach to find the partition closest to that obtained on the data set from its subsets.

1. INTRODUCTION

Data partitioning, also known as cluster analysis, is the process of finding groups in data that are ideally characterised by high internal similarity and significant differences between groups [1]. Today, the rapid growth of acquisition and storage technologies has led to an increase in the amount of data available in various fields [2]. This data is often geographically dispersed across different locations and can be used to extract meaningful insights, dependencies or correlations; traditional centralised clustering algorithms are impractical due to data volume, processing capacity and privacy concerns associated with data aggregation [3].

Distributed clustering algorithms address this by building the overall model without requiring access to data from different locations. Two architectures are used [4]: synchronous, where all nodes build the global model together, requiring extensive data exchange, with communication either direct between pairs of nodes [5], circular between all nodes [6], or following a hierarchical structure [7]. Asynchronous methods, where each node builds a local model and sends it to a central location to be combined into a global model [8]. This method is more privacy friendly as only local models are shared.

K-means [9, 10] is one of the most extensively studied and widely used clustering algorithms, favoured for its conceptual simplicity and computational efficiency for clustering large datasets [11]. Numerous distributed variants of the k-means algorithm have been proposed, they differ in the way they communicate and combine the local results [12]. In order to reduce communication between the different sites, a Peer-to-Peer (P2P) architecture has been proposed by Datta et al. [13] where each node of the network calculates the clusters of local data, and exchanges the centroids with its neighbors. Each neighbor recalculates its centroids, using its local data, and the centroids obtained from its neighbors. However, no proof of convergence nor indication on the speed of convergence of the algorithm was provided.

Ng [14] proposed to perform a two-level classification. The first one on the local data of each node, and the second on the local partitions generated by the previous step by performing another clustering of their centroids.

Jin et al. [15] proposed a distributed fast and exact k-means clustering (DFEKM) that uses data sampling to improve performance. Each site sends a sample of its data to a central site where it is grouped using k-means method. After a complete run through all data, the centers are calculated, and the convergence is determined based on a metric called the confidence radius estimated in the previous step. The disadvantage of the proposed method is that the data set is sampled, the results strongly depend on the samples drawn (its size and content) and that in some cases, groups present in data may not exist in the sample. The best sample is detailed by Broder et al. [16].

In the study of Durut et al. [17], a master/slave architecture is used to distribute the first step of k-means. On each iteration, all nodes perform the assignment step and calculate partial averages using their local data. Then they transmit the results to a master node, which calculates the overall average and outputs it to the other nodes. The same approach was used by Benchara and Youssfi [18] on Cloud micro-services.

Qin et al. [19] proposed a distributed k-means which each step is executed in distributed manner. The initial centroids are chosen by collaborating all sites. After local data are affected to the closest centroid iteratively, the average consensus algorithm is used to calculate the new clusters and the convergence is reached the centroids are unchanged.

The algorithm proposed by Kotary and Nanda [20] solve the distributed clustering problem as a constrained optimization using Lagrange multipliers. It aims to make uniform the number of groups on all nodes by sharing the cluster centers between neighbors. They assumed that the cluster centers are very similar after convergence at each node. No proof of convergence has been provided for the proposed method. The solution may vary if the data is not uniformly distributed.

Xia et al. [21] used the differential privacy technology local differential privacy (LDP) [22] in their distributed k-means solution. At each iteration a central site broadcasts the centroids to all the sites. The latter take care of the data assignment step and send to the central site the perturbed data according to the LDP principle and its membership group. The site recalculates the new centroids until the convergence. This solution is very greedy in term of communication because each site sends all these data at each iteration.

Soliman et al. [3] presented an asynchronous decentralized k-means clustering algorithm for independently and identically distributed (IID) data, which is not the case (Non-IID Data) as defined by Rodríguez-Barroso et al. [23]. The process begins by generating local model by k-means and iteratively share their result, at each node. The received local models are clustered using k-means to find the centers to be combined. Next, a weighted averaging is applied using the estimated cardinalities by HyperLogLog counters.

In recent years, the integration of topology and geometry opens up new perspectives for data mining and allows for more reliable and richer results [24]. These approaches make it possible to analyse complex data and create abstract representations of all the characteristics of the data, which makes it possible to reveal intrinsic structures that are not easily identifiable with traditional statistical techniques [25]. The promising results from the application of these methods across various domains, such as biology and image processing, highlight their potential as powerful tools for data analysis [26].

In the present article, we propose an asynchronous approach considering that distributed clustering is performed in three stages. First, each site searches for the clusters (models) of its local data and transmit them to a central site. Secondly, a central site analyses and searches for the partition of the set (the global model) based on the notion of the topological relationship between the different components of the local models.

Aggregation using topological relationships allows only subgroups that share data to be merged, so clusters that are included are merged first as they represent maximum data sharing and overlap second, while disjoint subgroups are not merged. distance-based methods do not allow this distinction, especially in a distributed context. Finally, we search for the right number of groups for the global model.

Privacy is ensured by the fact that there is no sharing of data samples between nodes. Nodes process data locally and share only relevant information, such as local models. The isolation of data samples contributes to the confidentiality and privacy of the information being processed. This is a significant advantage over other distributed clustering methods that share data across multiple nodes.

The paper is structured as follows: Section 2 presents some preliminary notations necessary for the presentation and better understanding of our work. The proposed approach is detailed in Section 3, followed by a time complexity and convergence analysis in Section 4. The experimental procedure is then described in Section 5. Section 6 is dedicated to the evaluation and discussion of the obtained experimental results. Finally, some concluding remarks are given in Section 7.

2. PRELIMINARIES

In this section, we list some preliminary notations and definitions frequently used in solving the addressed problem. Each site S has its own set of Ns data denoted by Xs, which are grouped using k-means in ks groups $\{C_1, C_2, \dots C_{ks}\}$. A group C_i is represented by a centroid (the average of these members), it is characterized by the number of instances N_i and the radius R_i defined by:

$$C_i = \frac{1}{N_i} \sum_{x \in X_S} X_i \tag{1}$$

$$N_i = \text{card}(C_i) \tag{2}$$

$$R_i = \frac{1}{N_i} \sum_{x \in C_i} d(x, C_i) \tag{3}$$

The adopted reasoning is based on the topological relationships between the different components of groups (i.e. centroid). Topological relations define the most primitive spatial relations that are allowed between entities, such as disjoint, meet, overlap, and inside (inclusion) [27]. We are interested in three situations: Inclusion, overlap and well separated. We define for each situation an operation: replacement, merging, and selection.

2.1 Replacement operation

The replacement operation aims to eliminate the redundancy contained in the raw model. It's possible that two groups have exactly the same representatives (centroid) but some groups can be covered by others and find themselves eliminated (see Figure 1). So, a center C_i only replaces C_j if C_j is included in C_i. Formally:

$$\begin{gathered} \forall C_i, C_j \in C, C_i \subset C_j \\ \text{if } d(C_i, C_j) < R_i \text{ or } d(C_i, C_j) < R_j \text{ and } R_i > R_j \end{gathered} \tag{4}$$

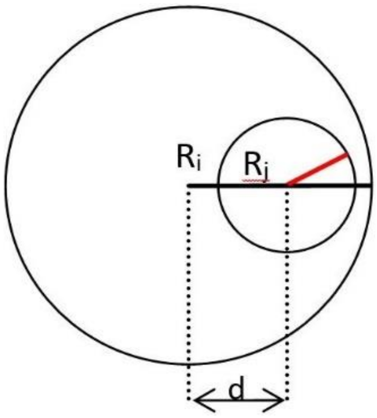

Figure 1. The center j is covered by I

2.2 Merging operation

The merging operation consists in applying a hierarchical grouping on the overlapping subgroups (see Figure 2). Two groups overlap if:

$$d(C_i,C_j) < (R_i + R_j) \text{ and } d(C_i,C_j) > R_i \text{ and } d(C_i,C_j) > R_j \quad (5)$$

Following the merger of two subgroups, the new one is created to replace them, its center is calculated by the weighted average of their centers. Its radius must allow it to cover/include the two subgroups from which it comes. As we do not have access to global data, this radius is estimated geometrically. We notice that the minimum diameter D of the group enveloping two overlapping groups is equal to the difference between the sum of their diameters (D_1, D_2) and the size of the overlap zone OL. Formally:

$$D = D_1 + D_2 - \mathrm{OL} = 2*R_1 + 2*R_2 - \mathrm{OL} \quad (6)$$

$$L = R_1 + R_2 - d(C_1,C_2) \quad (7)$$

$$D = 2*R_1 + 2*R_2 - (R_1 + R_2 - d(C_1,C_2)) = R_1 + R_2 + d(C_1,C_2) \quad (8)$$

$$R = \frac{R_1 + R_2 + d(C_1,C_2)}{2} \quad (9)$$

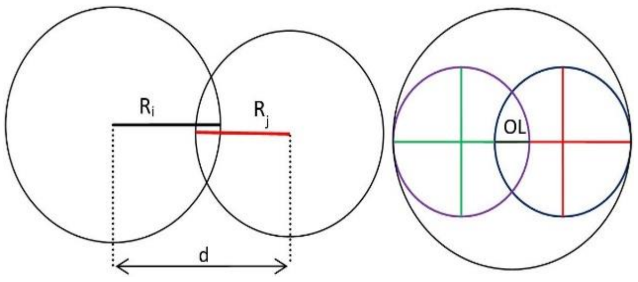

Figure 2. The overlap of two subgroups

2.3 Selection operation

When the replacement and the merging operations do not allow to obtain the expected number of groups, we are in a situation where the subgroups are neither in inclusion nor in overlap, but they are well separated. In this case, we choose the desired number of groups that maximize the inter-group inertia.

Algorithm 1: Maximum separation
Input: Desired number of groups *k*, set of centroids
Output: Partition of *k* groups

```
1   // (first two group)
      Search for $(C_1, C_2) = argmax\ d(C_i,C_j)$
2   while k is not reached do
3       Assign each centroid to the closest available groups
4       Select the one that has the greatest distance from its
          closest group
5       Add this centroid to the selected groups
6       Increase k
7   End while;
```

The groups are chosen one by one (except for the first two groups) as follows: The first two groups are the two farthest subgroups. The other centers are drawn from the set of subgroups where each one is assigned to the set of groups already chosen. The subgroup which has the greatest distance from its closest group will be part of the set of final groups. The algorithm (1) illustrates the principle.

3. DISTRIBUTED DATA CLUSTERING

The proposed approach (see algorithm (2)) is based on an asynchronous model that considers the distributed clustering is achieved in two stages (plus one for the choice of the global number of groups). First, each site realizes the search for the local models of its local data (local partitions), at the end of this step, we retain for each group C_i two characteristics, the number of instances N_i and the radius R_i. These data (groups, number of instances and radius) are sent to a central site that analyzes and searches for the partition of the distributed data (global model).

The union of the local partitions constitutes a first model of the distributed data which the number of groups is equal to the sum of the numbers of local groups ($\mathrm{N\_total} = \sum k_i$). This raw model is characterized by a large number of subgroups that are gradually refined using the three operations mentioned (replacement, fusion, and selection).

The refine algorithm proceeds to eliminate the redundancy by replacing the included groups with those that cover them until no inclusion relationship is found or the fixed number of groups is reached. If the latter is not reached, we use the merging operation where we merge the overlapping groups. We start with the closest, if the number of groups is not obtained, we choose the n centers among the remainder that increase the separability (the selection operation).

Therefore, the aggregation is performed in a hierarchical way using a sophisticated strategy that prioritizes the merging of subgroups based on their data sharing; similarity criteria are also considered to guarantee an efficient merging. As a consequence, subgroups with the highest degree of data sharing (inclusion) and similarity are merged first, followed by overlapping subgroups. Disjoint subgroups are not combined since they are fundamentally distinct and unrelated groups.

Algorithm 2: Refine
Input: Desired number of groups k, set of centroids (local partitions)
Output: Partition of *k* groups

```
1 while there are inclusions and k is not reached do
2     Replace groups included
3 while there are overlaps and k is not reached do
4     Merge the closest
5 if number of groups not reached then
6     Select the desired number of groups;
```

The third step is to determine the overall number of groups. We generate partitions with the number of groups from 2 to N_total (N_total is the sum of the numbers of local groups) and ship them to the other sites where they are evaluated on their local data. The best partition is the one with the best quality indexes average.

The convergence of the algorithm is guaranteed by the design of the algorithm itself. Since the number of subgroups is limited, as well as the number of included and overlapping relations, it is guaranteed that the algorithm will stop after a finite number of iterations, rather than running forever and eventually converging to a solution.

4. TIME COMPLEXITY AND CONVERGENCE

The analysis of the time complexity of the Refine algorithm requires a thorough examination of the computational costs associated with each step.

The algorithm is divided into three main parts: The merging of included groups, this phase is dominated by the search for inclusions in the entire set of elements, which is conducted comparing each pair of subgroups with a complexity of $O(N^2)$. The merging of overlapping groups, like the preceding step, is dominated by the search for overlaps in the set of elements present at this phase, which is $(N - N_{inc})$, so its complexity is $O((N - N_{inc})^2)$. The last step consists of forming groups from the remaining disjoint clusters (N_{disj} disjoint clusters) with a complexity of $O(N_{disj}^2 + k\,N_{disj})$.

Combining the aforementioned complexities results in a total complexity of $O(N^2 + (N - N_{inc})^2 + N_{disj}^2 + K\,N_{disj})$. Since N_{inc} and N_{disj} are subtotals of N, the dominant complexity is still $O(N^2)$. Regarding the small size of the treated subgroups, the complexity is acceptable.

The convergence of the algorithm is guaranteed by the design of the algorithm itself. Since the number of subgroups is limited, as well as the number of included and overlapping relations, it is guaranteed that the algorithm will stop after a finite number of iterations, rather than running forever and eventually converging to a solution.

5. EXPERIMENTATION

In experimental phase, we seek to determine the efficiency of our approach to find the closest partition to the one obtained with the dataset from its subsets. The datasets used in our experiments come from the UCI Machine Learning Repository, a widely used and publicly available collection of benchmark datasets for machine learning research. We chose to use the following datasets: High Time Resolution Universe (HTRU2) [28] and Tamil Nadu Electricity Board hourly readings [29]. These datasets were obtained in their pre-processed form provided by the repository. We also used the image shown in Figure 3.

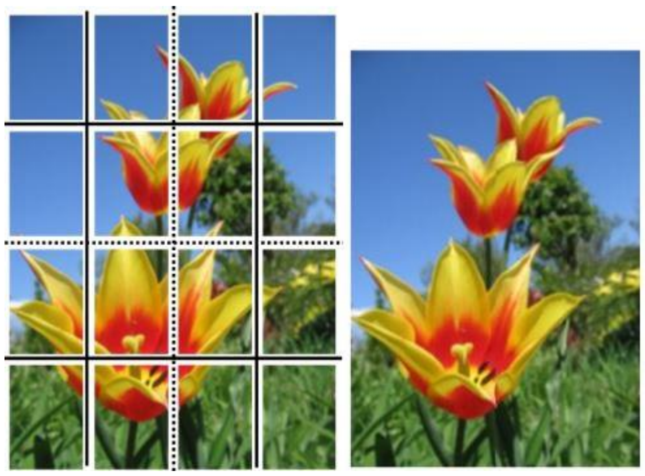

Figure 3. Tulip image divided into 4 and 16 delimited by dotted lines

The HTRU2 is a dataset that describes a sample of candidate pulsars, each one characterized by eight continuous variables and one to designate the membership class. It contains 17898 instances distributed into two classes, 1639 known as true pulsars and 16259 identified by human annotators as noise caused by radio interference.

Tamil Nadu electricity board hourly readings represent the electricity consumption collected in real time every hour in residential, commercial, industrial and agricultural areas around Thanjavur City. This dataset has five attributes (forkva, forkw, type, sector and service). We chose to use three attributes:

- For classification: Forkva and forkw attributes.
- For the label: Sector attribute (integer value from 0 to 4) for the label.

These data are divided and distributed randomly over 2, 4, 8 and 16 sites (except the Tulip image dataset which is divided into 4 and 16 sites). Each site uses k-means clustering for local data clustering. We used an initialization algorithm based on the misclassified instance [30]. The latter has the advantage of ensuring the same expected results and to guarantee good quality by initially maximizing the inter-class inertia compared to a random initialization.

In order to determine the locally optimal number of groups k, we varied k=2, ..., 16. For each value of *k*, we calculated the silhouette index (Si) [31]. The appropriate number of groups is the one that maximizes this value [32].

On the central site, the refine procedure is launched to generate partitions where the number of groups varies from 2 to N_total. In order to determine the optimal partition, we evaluated the generated partitions on each site. The evaluation of the generated partitions is based on the same index of quality silhouette and we used the Mean Square Error (MSE) to compare partitions with the same number of groups (compactness). Then, we compared the chosen partition with the one obtained from the entire dataset to measure their resemblance. For this purpose, we use F-measure [33], Jaccard index [34] and variation of information (VI) [35].

6. RESULTS AND DISCUSSION

This section presents a summary of the obtained results. In Section 6.1, we evaluated the different datasets and their subsets in order to determine the optimal number of groups. Then, we compared the generated partitions and those obtained (with all datasets) in Section 6.2.

6.1 Individual clustering results

Table 1. Number of groups obtained on each dataset

Data set		**Tulip image**	**HTRU2**	**Tamil Nadu**
Number of groups	Global data	5	2	4
	2 sites	/	2 groups for all sites	4 groups for all sites
	4 sites	6, 4, 7, 2		
	8 sites	/		
	16 sites	3, 3, 2, 6, 4, 2, 3, 2, 3, 6, 4, 5, 3, 2, 4, 2		

The number of groups that maximized the silhouette index in each dataset is summarized in Table 1. Note that for HTRU2 and Tamil Nadu, the number of groups in the subsets (local data) is identical to that obtained in the entire data. This finding can be explained by the diversity of the data in the subsets. The case of the Tulip image dataset is a good testing scenario for our approach, as the number of groups is different for each subset, and even greater in some cases, than the number of the overall dataset.

6.2 Evaluation of generated partitions

We compared the partitions produced by our approach in the context of distributed clustering to those produced by a global clustering on the entire dataset using internal index and external index. The number of groups varies from 2 to N_total.

In the case of the Tamil Nadu dataset (see Figure 4), MSE decreases each time the number of groups (2 to 4) is increased to the optimal number (4). Performances are similar at this level, then it continues to decrease for global clustering while it does not vary as quickly for generated partitions.

Likewise, for the curves of the silhouette index (see Figure 5), they are similar before the optimal zone (number of groups = 4) then differ beyond. We have the same phenomenon for HTRU2 dataset results (in Figure 6) because the diversity of local data which contains data that fit different groups in the global dataset.

Indeed, local clustering is at the optimal range (4 groups in Tamil and 2 in HTRU2). It can only produce a partition with a number of groups less or equal to the optimal range which has a similar quality to global clustering.

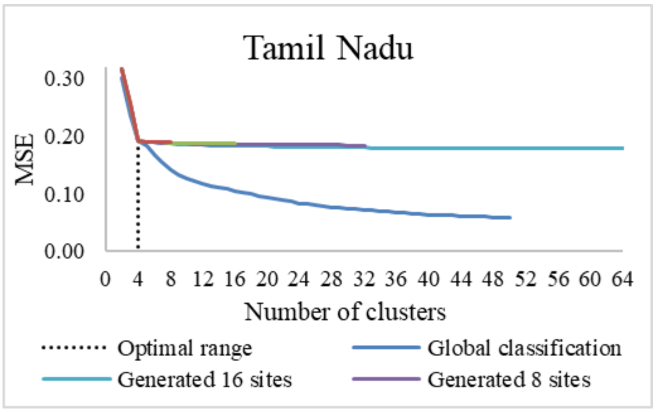

Figure 4. MSE according to the number of groups for Tamil Nadu dataset

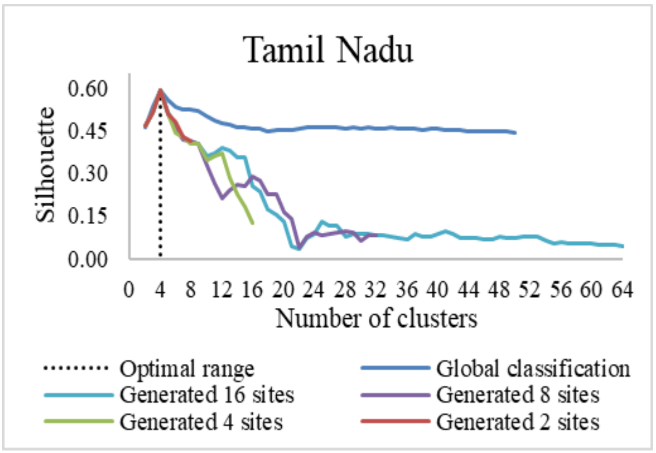

Figure 5. Silhouette index according to the number of groups for Tamil Nadu dataset

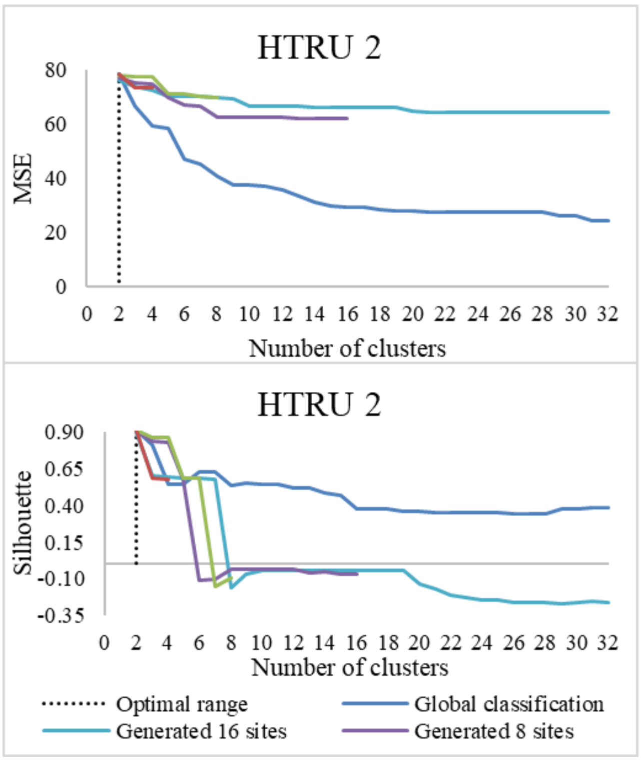

Figure 6. MSE and silhouette index according to the number of groups for HTRU2 dataset

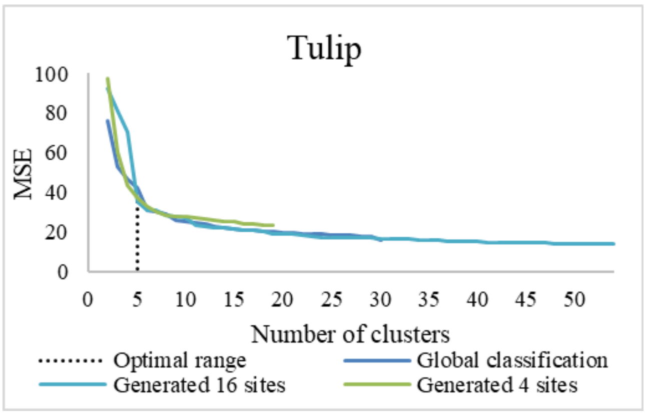

Figure 7. MSE according to the number of groups for the Tulip image

Figure 7 shows the MSE from the Tulip dataset results. The generated partitions are less or equal to 3 groups and have a lower quality (calculate the ratio) compared to the overall partition. From the number of groups = 5, the different partitions come together. MSE values are close except for the partitions generated in 4 sites, which from 10 groups have values greater than the others.

From Figure 8, the silhouette index is similar for the different clustering at the optimal point (5 groups). Beyond this point, the values displayed fluctuate until the value 10 where the global classification takes over with certain stability.

Tables 2 and 3 summarize the obtained results by evaluating the global partition and the generated partitions with the optimal number of groups on the full dataset.

We observe that the generated partitions are similar to those obtained on the dataset with a small improvement:

- MSE from 0.107% to 3.081%
- Silhouette index from 0.001% to 1.585%.

Table 2. Index quality in optimal range for HTRU (2 groups) and Tamil Nadu (4 groups)

		HTRU2			
		MSE	Ratio	Silhouette	Ratio
HTRU2	Global	78.228	-	0.907980	-
	2 Sites	78.145	0.107	0.907487	0.054
	4 Sites	77.828	0.512	0.905870	0.232
	8 Sites	76.898	1.701	0.900164	0.861
	16 Sites	75.818	3.081	0.893588	1.585
Tamil Nadu	Global	0.1915	-	0.588989	-
	2 Sites	0.1915	0%	0.589011	0.004
	4 Sites	0.1915	0%	0.588992	0.001
	8 Sites	0.1915	0%	0.588979	0.002
	16 Sites	0.1915	0%	0.588962	0.005

Table 3. Index quality in optimal range for Tulip image (5 groups)

	Tulip image			
	MSE	Ratio	Silhouette	Ratio
Global	42.1554	-	0.747445	-
4 Sites	37.5698	10.88	0.761358	1.86
16 Sites	35.7468	15.20	0.768436	2.81

Table 4. Average obtained silhouette index on the local data

	NBGR	2 Sites	4 Sites	8 Sites	16 Sites
HTRU2	2	0.906	0.905	0.896	0.891
	3	0.582	0.862	0.835	0.594
	4	0.572	0.856	0.827	0.59
	5	-	0.579	0.573	0.582
	6	-	0.579	-0.097	0.579
	7	-	-0.142	-0.09	0.577
	8	-	-0.091	-0.021	-0.159
	9	-	-	-0.021	-0.07
	10	-	-	-0.023	-0.041
Tamil Nadu	2	0.463	0.463	0.463	0.463
	3	0.512	0.513	0.513	0.513
	4	0.589	0.589	0.588	0.588
	5	0.509	0.506	0.501	0.502
	6	0.481	0.434	0.442	0.432
	7	0.425	0.435	0.405	0.426
	8	0.411	0.398	0.403	0.383
	9	-	0.387	0.369	0.387
	10	-	0.301	0.354	0.345
Tulip image	2		0.37		0.41
	3		0.611		0.454
	4		0.707		0.432
	5		0.754		0.597
	6		0.721		0.591
	7		0.613		0.578
	8		0.614		0.579
	9		0.593		0.55
	10		0.587		0.534

With Tulip image, overall, the quality of the generated partitions is better than the one obtained on the whole dataset (global clustering) with a significant improvement of MSE from 10.88% to 15.20% as well as for silhouette index from 1.86% to 2.81%. The various results show an improvement which can be explained by the fact of processing subsets of data allowing the approach a better detection of the groups. It reduces the influence of outliers (noise). Figure 4 confirms the results shown in Table 4. In Figure 9, (a1) and (a2) provide better separation of parts of images (i.e., green grass at the bottom of the image).

In order to determine the globally optimal number of groups, we evaluated the generated partitions on the local data of the different sites, the one with the best average quality index is chosen. This process is done automatically. Table 4 shows the result of different evaluations by generated partitions in different case. We observe that the best value of silhouette index corresponds to the same number of groups in the global data.

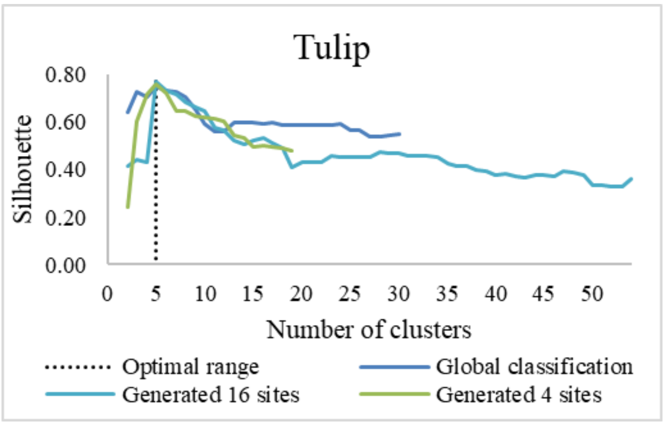

Figure 8. Silhouette index according to the number of groups for the Tulip image

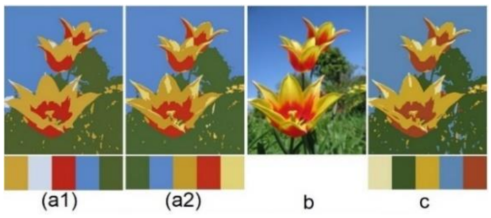

Figure 9. Clustering results using generated partitions: (a1) 4 sites, (a2) 16 sites (b) Original image and (c) Global clustering

Table 5. Similarity measurement between global and generated partition (Gen.)

		Partition		VI	Jaccard	F Measure
HTRU2	2 sites	Global	Gen.	**0.0014**	**0.9999**	**0.9999**
		Gen.	Label	0.5742	0.8032	0.8909
		Global	Label	0.5739	0.8033	0.8909
	4 sites	Global	Gen.	0.0195	0.9976	0.9988
		Gen.	Label	0.5806	0.8016	0.8899
		Global	Label	0.5739	0.8033	0.8909
	8 sites	Global	Gen.	0.0654	0.9892	0.9945
		Gen.	Label	0.6019	0.7951	0.8859
		Global	Label	0.5739	0.8033	0.8909
	16 sites	Global	Gen.	0.1128	0.9781	0.9889
		Gen.	Label	0.6293	0.7869	0.8807
		Global	Label	0.5739	0.8033	0.8909
Tamil Nadu	2 sites	Global	Gen.	0.0321	0.9939	0.9969
		Gen.	Label	0.2281	0.3715	61.14
		Global	Label	0.2281	0.3715	61.14
	4 sites	Global	Gen.	0.031	0.9943	0.9971
		Gen.	Label	2.9142	0.2281	0.3715
		Global	Label	2.9142	0.2281	0.3715
	8 sites	Global	Gen.	0.027	0.995	0.9975
		Gen.	Label	2.9142	0.2281	0.3715
		Global	Label	2.9142	0.2281	0.3715
	16 sites	Global	Gen.	0.0397	0.9924	0.9962
		Gen.	Label	2.9142	0.2281	0.3715
		Global	Label	2.9142	0.2281	0.3715

Table 5 shows a great similarity between the generated partition and the global partition. By comparing the generated partition with the global partition according to the labels, we noticed that they have the same ability to detect groups, the centralized approach is more efficient for groups detection, the better it is for the distributed one. The results show that the detection capacity is better for HRTU2 than for Tamil Nadu dataset.

Finally, like any other research, this work also has some limitations. The performance analysis of the proposed approach focused only on finding the closest partition to the one obtained on the dataset from its subsets and did not explore the search for a partition with a fixed number of groups with the same quality as clustering on the entire of the dataset. Indeed, the number of groups is found automatically based on the silhouette index for both local clustering or generated partitions, setting a fixed number of groups for the global partition (generated) sometimes leads to bad solutions (see Figures 4 to 8), it is difficult to apply it on local sites due to the nature and specificity of local data.

A key limitation of this method is that its goal is to find the closest partition to that obtained on the full dataset from its subsets, so its performance depends on the ability of k-means to cluster the full dataset correctly. If k-means fails on the full dataset, for any reasons, then even exploiting the topological relationships between local groups the distributed approach will fail to find a good global partition.

7. CONCLUSIONS

This paper presents a distributed clustering algorithm based on k-means that employs a centralized architecture. The algorithm exploits the topological relationships between local groups (inclusion, overlapping, and disjoint) to generate a global clustering, unlike traditional distance-based methods that merge clusters based on proximity and can merge disjoint clusters because they are closest. Such arbitrary merges can severely compromise the resulting clusters.

The key contribution of the proposed approach is its ability to identify the partition that is closest to the partition obtained on the dataset from its subsets without relying on local data. Furthermore, the optimal number of groups for the global clustering is dynamically determined by evaluating the generated solution on local data, taking into account the fact that each local clustering may have a different optimal number of groups. This was validated through experiments conducted in various scenarios.

Nevertheless, the study also identified certain limitations. The performance analysis was conducted with the objective of identifying the partition that was most similar to the partition obtained from the dataset's subsets. This analysis did not aim to identify a partition with a fixed number of groups. Consequently, when the predefined number of groups does not correspond to the optimal number for the dataset, the algorithm encounters difficulties in accurately capturing the underlying data structure. Consequently, the partition generated may not be as similar to the centralized as would be optimal. This is due to the nature and specificity of the local data, as illustrated in Figures 4 to 8.

In summary, this work presents a promising distributed clustering algorithm that addresses key challenges in knowledge discovery across distributed data sources. It does so by leveraging topological relationships to prioritize merging clusters that share data, while disjoint subgroups are not merged. Further research is required to extend the applicability of this approach to centroid-based clustering algorithms.

REFERENCES

[1] Larose, D.T., Larose, C.D. (2014). Hierarchical and k-means clustering. In: Discovering Knowledge in Data: An Introduction to Data Mining. John Wiley & Sons, pp. 209-227. https://doi.org/10.1002/9781118874059.ch10

[2] Kargupta, H., Han, J., Philip, S.Y., Motwani, R., Kumar, V. (2008). High-Performance Distributed Data Mining. In: Next Generation of Data Mining. CRC Press, New York, pp. 152-168. https://doi.org/10.1201/9781420085877

[3] Soliman, A., Girdzijauskas, S., Bouguelia, M.R., Pashami, S., Nowaczyk, S. (2020). Decentralized and adaptive k-means clustering for non-iid data using hyperloglog counters. In Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, pp. 343-355. https://doi.org/10.1007/978-3-030-47426-3_27

[4] Laloux, J.F., Le-Khac, N.A., Kechadi, M.T. (2011). Efficient distributed approach for density-based clustering. In 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Paris, France, pp. 145-150. https://doi.org/10.1109/WETICE.2011.27

[5] Forero, P.A., Cano, A., Giannakis, G.B. (2008). Consensus-based distributed expectation-maximization algorithm for density estimation and classification using wireless sensor networks. In 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, NV, USA, pp. 1989-1992. https://doi.org/10.1109/ICASSP.2008.4518028

[6] Weng, Y., Xiao, W., Xie, L. (2011). Diffusion-based EM algorithm for distributed estimation of Gaussian mixtures in wireless sensor networks. Sensors, 11(6): 6297-6316. https://doi.org/10.3390/s110606297

[7] Bendechache, M., Kechadi, M.T. (2015). Distributed clustering algorithm for spatial data mining. In 2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM), Fuzhou, China, pp. 60-65. https://doi.org/10.1109/ICSDM.2015.7298026

[8] Januzaj, E., Kriegel, H.P., Pfeifle, M. (2003). Towards effective and efficient distributed clustering. In Workshop on Clustering Large Data Sets (ICDM2003), 60. https://www.dbs.ifi.lmu.de/Publikationen/Papers/DBDC.pdf.

[9] Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. University of California Press. https://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf.

[10] Selmi, A.T.E., Zerarka, M.F., Cheriet, A. (2024). Enhancing K-means clustering with post-redistribution. Ingénierie des Systèmes d’Information, 29(2): 429-436. https://doi.org/10.18280/isi.290204

[11] Tan, P.N., Steinbach, M., Kumar, V. (2018). Cluster analysis: Basic concepts and algorithms. In Introduction

to Data Mining, 2nd ed. Pearson Education, India, pp. 525-603. https://www.pearsonhighered.com/assets/preface/0/1/3/3/0133128903.pdf.

[12] Ahmed, M., Seraj, R., Islam, S.M.S. (2020). The k-means algorithm: A comprehensive survey and performance evaluation. Electronics, 9(8): 1295. https://doi.org/10.3390/electronics9081295

[13] Datta, S., Giannella, C., Kargupta, H. (2008). Approximate distributed k-means clustering over a peer-to-peer network. IEEE Transactions on Knowledge and Data Engineering, 21(10): 1372-1388. https://doi.org/10.1109/TKDE.2008.222

[14] Ng, M.K. (2000). K-means-type algorithms on distributed memory computer. International Journal of High Speed Computing, 11(2): 75-91. https://doi.org/10.1142/S0129053300000096

[15] Jin, R., Goswami, A., Agrawal, G. (2006). Fast and exact out-of-core and distributed k-means clustering. Knowledge and Information Systems, 10: 17-40. https://doi.org/10.1007/s10115-005-0210-0

[16] Broder, A., Garcia-Pueyo, L., Josifovski, V., Vassilvitskii, S., Venkatesan, S. (2014). Scalable k-means by ranked retrieval. In WSDM '14: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, New York, USA, pp. 233-242. https://doi.org/10.1145/2556195.2556260

[17] Durut, M., Patra, B., Rossi, F. (2012). A discussion on parallelization schemes for stochastic vector quantization algorithms. arXiv preprint arXiv:1205.2282. https://doi.org/10.48550/arXiv.1205.2282

[18] Benchara, F.Z., Youssfi, M. (2021). A new scalable distributed k-means algorithm based on Cloud micro-services for High-performance computing. Parallel Computing, 101: 102736. https://doi.org/10.1016/j.parco.2020.102736

[19] Qin, J., Fu, W., Gao, H., Zheng, W.X. (2016). Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory. IEEE Transactions on Cybernetics, 47(3): 772-783. https://doi.org/10.1109/TCYB.2016.2526683

[20] Kotary, D.K., Nanda, S.J. (2019). Automatic determination of K in distributed K-means clustering. Procedia Computer Science, 165: 556-564. https://doi.org/10.1016/j.procs.2020.01.050

[21] Xia, C., Hua, J., Tong, W., Zhong, S. (2020). Distributed K-Means clustering guaranteeing local differential privacy. Computers & Security, 90: 101699. https://doi.org/10.1016/j.cose.2019.101699

[22] Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A. (2011). What can we learn privately? SIAM Journal on Computing, 40(3): 793-826. https://doi.org/10.1137/090756090

[23] Rodríguez-Barroso, N., Stipcich, G., Jiménez-López, D., Ruiz-Millán, J.A., Martínez-Cámara, E., González-Seco, G., Luzón, M.V., Veganzones, M.A., Herrera, F. (2020). Federated learning and differential privacy: Software tools analysis, the Sherpa. ai FL framework and methodological guidelines for preserving data privacy. Information Fusion, 64: 270-292. https://doi.org/10.1016/j.inffus.2020.07.009

[24] Snášel, V., Nowaková, J., Xhafa, F., Barolli, L. (2017). Geometrical and topological approaches to Big Data. Future Generation Computer Systems, 67: 286-296. https://doi.org/10.1016/j.future.2016.06.005

[25] Alomari, H.W., Al-Badarneh, A.F., Al-Alaj, A., Khamaiseh, S.Y. (2023). Enhanced approach for agglomerative clustering using topological relations. IEEE Access, 11: 21945-21967. https://doi.org/10.1109/ACCESS.2023.3252374

[26] Almgren, K., Kim, M., Lee, J. (2017). Mining social media data using topological data analysis. In 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, USA, pp. 144-153. https://doi.org/10.1109/IRI.2017.41

[27] Felice, P.D., Clementini, E. (2009). Topological Relationships. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_432

[28] Robert, L. (2016). High Time Resolution Universe Survey South (HTRU2). Figshare. https://doi.org/10.6084/m9.figshare.3080389.v1

[29] Kalyani, K. (2013). Tamilnadu electricity board hourly readings. UCI Machine Learning Repository. https://doi.org/10.24432/C5KP4K

[30] Guellil, Z., Zaoui, L. (2009). Proposition d'une Solution au Problème d'Initialisation Cas du K-means. In Proceedings of the 2nd Conférence Internationale sur l'Informatique et ses Applications (CIIA'09), CEUR Workshop Proceedings, Saida, Algeria, p. 547. http://ceur-ws.org/Vol-547/37.pdf.

[31] Rousseeuw, P.J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20: 53-65. https://doi.org/10.1016/0377-0427(87)90125-7

[32] Pollard, K.S., Van Der Laan, M.J. (2002). A Method to Identify Significant Clusters in Gene Expression Data. In U.C. Berkeley Division of Biostatistics Working Paper Series. Berkeley Electronic Press: Berkeley, pp. 107. https://biostats.bepress.com/ucbbiostat/paper107.

[33] Zhang, E., Zhang, Y. (2009). F-Measure. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA, pp. 50. https://doi.org/10.1007/978-0-387-39940-9_483

[34] Leskovec, J., Rajaraman, A., Ullman, J.D. (2020). Chapter 3. Finding similar items. In Mining of Massive Data Sets, 3rd ed. Cambridge University Press, pp. 76-77. http://infolab.stanford.edu/~ullman/mmds/ch3.pdf.

[35] Meilă, M. (2003). Comparing clusterings by the variation of information. In Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA. Springer, Berlin, Heidelberg, pp. 173-187. https://doi.org/10.1007/978-3-540-45167-9_14