Abstract — Distributed data mining techniques and mainly distributed clustering are widely used in last decade because they deal with very large and heterogeneous datasets which cannot be gathered centrally. Current distributed clustering approaches are normally generating global models by aggregating local results that are obtained on each site. While this approach analyses the datasets on their locations the aggregation phase is complex, time consuming and may produce incorrect and ambiguous global clusters and therefore incorrect knowledge. In this paper we propose a new clustering approach for very large spatial datasets that are heterogeneous and distributed. The approach is based on K-means Algorithm but it generates the number of global clusters dynamically. It is not necessary to fix the number of clusters. Moreover, this approach uses a very sophisticated aggregation phase. The aggregation phase is designed in such away that the final clusters are compact and accurate while the overall process is efficient in time and memory allocation. Preliminary results show that the proposed approach scales up well in terms of running time, and result quality, we also compared it to two other clustering algorithms BIRCH and CURE and we show clearly this approach is much more efficient than the two algorithms.

Keywords — Spatial data, clustering, distributed mining, data analysis, k-means.

I. INTRODUCTION

Across a wide variety of fields, datasets are being collected and accumulated at a dramatic pace and massive amounts of data that are being gathered are stored in different sites. In this context, data mining (DM) techniques have become necessary for extracting useful knowledge from the rapidly growing large and multi-dimensional datasets [U. Fayyad-96]. In order to cope with scalability of data, researchers have developed parallel versions of the sequential DM algorithms [A. Freitas-07]. Because of the resource limitations to process large amounts of datasets, parallel data mining (PDM) can be deployed on distributed platforms such as clusters of computers or Grid [I. Foster-99]. In PDM algorithms the calculations are performed on each node to create local results. These local results will be aggregated to build global ones. However, in the case of very large datasets, the aggregation phase will create significant communication overheads. In addition, this can only be possible if the datasets have already been distributed over the nodes which is currently the case as massive amounts of data are stored on different sites as they were produced. In this context, distributed data mining (DDM) techniques have become necessary for analysing these large and multi-dimensional datasets. DDM is more appropriate for large-scale distributed platforms where datasets are often geographically distributed and owned by different organisations. Many DDM tasks such as distributed association rules and distributed classification [T.G.Dietterich-00, H.Kargupta-00, R.Agrawal-96, L.Aouad-09, L.Aouad6-10, N.Le-Khac-10] have been proposed and developed in the last few years. However, only a few research concerns distributed clustering for analysing large, heterogeneous and distributed datasets. Recent researches [E.Januzaj-04, N.A.Le-Khac-07, L.Aouad1-07] have proposed a distributed clustering model that is based on two main steps: perform partial analysis on local data at individual sites and then send them to a central site to generate global models by aggregating the local results. Even though, the existing distributed clustering do not scale well and one of the biggest issues of these systems is the generation of good global models, because local models do not contain enough information for the merging process.

In this paper, we propose a new approach of distributed clustering. In our approach, centralise clustering is also performed at each site to build local models. These models are sent to the servers where clusters will be regenerated based on local models features. Further analysis will be carried out to create global models [J.F.Laloux-11].

The rest of this paper is organised as follows: In the next section we will give an overview of distributed data mining and discuss the limitations of traditional techniques. Then we will present and discuss our approach in Section 3. Section 4 presents the implementation of the approach and we discuss experimental results in Section 5. Finally, we conclude on Section 6.

II. RELATED WORK

While massive amounts of data are being collected and stored from not only science fields but also industry and commerce fields, efficient mining and management of useful
information of this data is becoming a scientific challenge and a massive economic need. This is even more critical when the collected data is located on different sites and are owned by different organisations [E. Jiawei-06]. This led to the development of DDM techniques to deal with huge, multi-dimensional, and heterogeneous datasets, which are distributed over a large number of nodes. Existing DDM techniques consist of two main phases: 1) performing partial analysis on local data at individual sites and 2) generating global models by aggregating the local results. These two steps are not independent since naive approaches to local analysis may produce incorrect and ambiguous global data models. In order to take the advantage of mined knowledge at different locations, DDM should have a view of the knowledge that not only facilitates their integration but also minimises the effect of the local results on the global models. Briefly, an efficient management of distributed knowledge is one of the key factors affecting the outputs of these techniques.

Moreover, the data that will be collected in different locations using different instruments may have different formats, features, and quality. Traditional centralised data mining techniques do not consider all the issues of data-driven applications such as scalability in both response time and accuracy of solutions, distribution and heterogeneity [L.Aouad-09, M.Bertolotto-07]. Some distributed DM approaches are based on ensemble learning, which uses various techniques to aggregate the results [Le-Khac-07], among the most cited in the literature: majority voting, weighted voting, and stacking [P.Chan-95, Reeves-93]. Recently, very interesting distributed approaches were introduced. The first phase consists of generating local knowledge based on local datasets. The second phase involves integrating local knowledge to generate global results or decisions. Some approaches are well suited to be performed on distributed platforms. For instance, the incremental algorithms for discovering spatio-temporal patterns by decomposing the search space into a hierarchical structure, addressing its application to multi-granular spatial data can be very easily optimised on hierarchical distributed system topology. From the literature, two categories of techniques are used: parallel techniques that often require dedicated machines and tools for communication between parallel processes which are very expensive, and techniques based on aggregation, which proceed with a purely distributed, either on the data based models or on the execution platforms [L.Aouad3-07, L.Aouad4-09]. However, the amount of data continues to increase in recent years, as such, the majority of existing data mining techniques are not performing well as they suffer from scalability. This becomes a very critical issue in recent years. Many solutions have been proposed so far. They are generally based on improvement of data mining algorithms so that they fit with the amounts of data at hand.

Clustering is one of the fundamental techniques in data mining. It groups data objects based on information found in the data that describes the objects and their relationships. The goal is to optimize similarity within a cluster and the dissimilarities between clusters in order to identify interesting structures in the data [L.Aouad3-07]. Clustering algorithms can be divided into two main categories, namely partitioning and hierarchical. Different elaborated taxonomies of existing clustering algorithms are given in the literature. Many parallel clustering versions based on these algorithms have been proposed [L.Aouad3-07, L.Dhillon-99, M.Ester-96, Garg-06, H.Geng-05, Inderjit-00, X.Xu-99], etc. These algorithms are further classified into two sub-categories. The first consists of methods requiring multiple rounds of message passing. They require a significant amount of synchronization. The second sub-category consists of methods that build local clustering models and send them to a central site to build global models [J.F.Laloux-11]. In [L.Dhillon-99] and [Inderjit-00], message-passing versions of the widely used k-means algorithm were proposed. In [M.Ester-96] and [X.Xu-99], the authors dealt with the parallelization of the DBSCAN density based clustering algorithm. In [Garg-06] a parallel message passing version of the BIRCH algorithm was presented. A parallel version of a hierarchical clustering algorithm, called MPC for Message Passing Clustering, which is especially dedicated to Microarray data was introduced in [H.Geng-05]. Most of the parallel approaches need either multiple synchronization constraints between processes or a global view of the dataset, or both [L.Aouad3-07].

Another approach presented in [L.Aouad1-07] also applied a merging of local models to create the global models, current approaches only focus on either merging local models or mining a set of local models to build global ones. If the local models cannot effectively represent local datasets then global model accuracy will be very poor [J.F.Laloux-11].

Both partitioning and hierarchical categories suffer from some drawbacks. For the partitioning class, we have k-means algorithm which needs the number of clusters fixed in advance, while in the majority of cases K is not known, furthermore hierarchical clustering algorithms have overcome this limit, they no not need to provide the cluster number as an input parameter, but they must define the stopping conditions for clustering decomposition, which are not easy.

III. DISTRIBUTED CLUSTERING

The proposed distributed approach is for clustering spatial datasets. It is based to two main steps; 1) it generates local clusters on each dataset assigned to a given processing node. The local clustering algorithm is chosen to be K-Means executed with a given (K) which can be different for each node (see Figure 1). K should be chosen to be big enough to identify all clusters of local dataset.
The boundaries of the clusters represents the new dataset, and they are much more smaller than the original datasets. So the boundaries of the clusters will become the local results at each node in the system. These local results are sent to the leaders through the tree topology of nodes till we reach the root node in which global results will be generated and stored.

IV. IMPLEMENTATION

A. Summarised Algorithm.

This section summarises the proposed approach. In the first step, local clustering is performed on each local dataset using K-means algorithm in parallel, the local number of clusters can be different in each site. Then, each local clustering in a node i produces $K_i$ sub clusters. In the next step, each node applies an algorithm for calculating the contour of each cluster generating locally. These contours will represent new local data for each node. At the end of local processes, each node compares its clusters to its neighbour’s clusters to find out if there is overlapping between the contours. Therefore each node will belong to a group, which represent its neighbours, so each node will communicate only with its neighbours, which optimises the communication in the system.

In the third step local contours are sent to the chosen merging leaders, these leaders are elected among nodes of each group, the leaders proceed to merge the overlapping contours of its group. Therefore, each leader generates new contours (new local dataset). We repeat second and third steps till we reach root node. The sub-clusters aggregation is done under tree structure and the global results are located in the last level of the tree (root node).

As in all clustering algorithms, the expected large variability in clusters shapes and densities is an issue. However, as we will show in the experiments section, the algorithm used for generating the cluster’s contour is efficient to detect well-separated clusters with different shapes. On the other hand our approach determines dynamically the number of the clusters without a priori knowledge about the data or an estimation process for the number of clusters.

- **Pseudo-code algorithm**: Distributed Clustering Algorithm based on K-means:

The system has a tree topology and the nodes execute local calculations in parallel.

1) Data Distribution: Initially data is divided among all nodes in the systems.
2) The basic nodes (tree leaves) will execute in parallel K-means algorithm with a $K_i$ different for each node that can be larger than the number of clusters actually present in these local datasets.
3) Neighbouring nodes must share their clusters to form even larger clusters using the overlay technique.
4) The results must reside in the father node (called ancestor).
5) Repeat 3 and 4 until reaching the root (heap) node.

In the following we give a pseudo-code that is executed in parallel in each node of the system:

---

**Fig 1. Overview of the proposed Approach**

After generating local results, nodes compare their local clusters with their neighbours nodes’ clusters, then a nodes called leaders will be elected to merge local clusters to form larger clusters using the overlay technique. These leaders are elected according to some conditions such as the capacity of node, etc. The process of merging clusters will continue until we reach the root node. The root node will contain the global clusters (models).

During the second phase communicating the local clusters to the heads may generate huge overhead. Therefore, the objective is to minimise the data communication and computational time, while getting accurate global results. Note that our approach while it is based on the same principle as existing approaches, it minimises the communication overheads due to the data exchange. Therefore instead of exchanging the whole data (whole clusters) between nodes (local nodes and leaders), we first proceed by reducing the data that represent a cluster. The size of this new data cluster is much smaller that the initial one. This process is carried out on each local node.

There are many data reduction techniques proposed in the literature. Many of them are focusing only in dataset size i.e., they try to reduce the storage of the data without paying attention to their knowledge behind this data. In [N-A-10], an efficient reduction technique has been proposed; it is based on density-based clustering algorithms. In this approach, we choose representatives of each cluster. However, selecting representatives is still a challenge in terms of quality and size of this set. We can choose, for instance, medoids points, core points, or even specific core points [E.Januzaj-04] as representatives [J.F.Laloux-11].

In our approach, we focus on the shape and the density of clusters created. The shape of a cluster is represented by its boundary points (called contour) (see Fig 2). Many algorithms for extracting the boundaries from a cluster are cited in the literature [M.J.Fadilia-04, A.Ray-97, M.Melkemi-00, Edelsbrunner-83, A.Moreira-07]. In our approach we used the algorithm proposed in [M. Dukchama-08] which is based in Triangulation to generate the boundary of the cluster, it is an efficient algorithm for constructing a possibly non-convex boundary. The algorithm is able to accurately characterise the shape of a wide range of different point distributions and densities with a reasonable complexity of $O(n \log n)$.

The expected large density with a reasonable complexity of $O(n \log n)$.
Algorithm 1 shows the pseudo code for the Node, in the system.

Algorithm 1: Distributed clustering algorithm based on K-Means
Input: \( X_i \): Dataset Fragment, \( K_i \): Number of sub-clusters for Node, \( D \): tree degree.
Output: \( K_{ig} \): Global Clusters (global results)

level = treeheight;
1) K-means(\( X_i, K_i \));
2) Node, joins a group G of D elements;
   // Node, joins his neighbourhood.
3) Compare cluster of Node, to other Node’s clusters in the same group;
   // look for overlapping between Clusters
4) j= elect leader Node();
   // elect a node which will merge the overlapping clusters
if (i <> j) then
   Send(contour i, j);
else
   if( level > 0 ) then
      level - - ;
      Repeat 2, 3, and 4 until level=1;
   else
      return (\( K_{ig} \): Node,’s clusters);

B. Example of execution
We suppose that the system contains \( N = 5 \) Nodes, each Node executes K-Means algorithm with different \( K_i \), as it is shown in the Fig 2. \( Node1 \) execute K-Means algorithm with \( K=30 \), \( Node2 \) with \( K=60 \), \( Node3 \) with \( K=90 \), \( Node4 \) with \( k=120 \), and \( Node5 \) with \( K= 150 \). Therefore each node in the system generate its local clusters, afterwards, local clusters, which are overlapping between each others are merged to generate new clusters and so on, till we obtain the final clusters. As we can see, although we started by different value of \( K \), we generated only five clusters results. (See Fig 2).

V. EXPERIMENTAL RESULTS

In this section, we study the performance of our Algorithm and demonstrate its effectiveness compared to BIRCH and CURE algorithms:

BIRCH: We used the implementation of BIRCH provided by the authors in [Tian-96]. The implementation performs a pre-clustering and then uses a centroid-based hierarchical clustering algorithm with time and space complexity that is quadratic to the number of points after pre-clustering. We set parameter values to the default values suggested in [Tian-96].

CURE: We used the implementation of CURE provided by the authors in [Sudipto-01]. The algorithm uses representative points with shrinking towards the mean. As described at [Sudipto-01], when two clusters are merged in each step of the algorithm, representative points for the new merged cluster are selected from the ones for the two original clusters rather than all points in the merged cluster.

Our Algorithm: Our algorithm is described in Section IV. It executes K-Means algorithm with different \( K \) for each node. The key point is to choose \( K \) as bigger than the number of correct clusters, i.e., the big \( K \) is, the accurate clusters generated are. As described at the end of Section IV, when two clusters are merged in each step of the algorithm, representative points for the new merged cluster are represented by union of the two contours of the two original clusters rather than all points in the merged cluster. This speeds up the execution times without adversely impacting the quality of clusters generated. In addition, our implementation uses the tree topology and heap data structures. Thus, this also improves the complexity of the algorithm.

![Fig 2. Distributed Clustering Algorithm based on K-Means](image-url)
A. Data sets

We experimented with different datasets containing points in two dimensions. As a short paper, we report here the results with only one dataset, as illustrated in TABLE1. The number of points in the dataset is also given in TABLE1. The dataset contains five big elliptic shapes that traditional partitioned and hierarchical clustering algorithms, including BIRCH and CURE, fail to find, we show that our Algorithm not only correctly clusters this dataset, but also its execution times is much quicker than BIRCH and CURE.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Points</th>
<th>Shape of Clusters</th>
<th>Number of Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14000</td>
<td>Big Oval (Egg Shape)</td>
<td>5</td>
</tr>
</tbody>
</table>

B. Quality of Clustering

We run the three algorithms on the same dataset to compare them with respect to the quality of clusters generated. Fig 3 shows the clusters found by the three algorithms for the same dataset. The clusters generated for BIRCH and CURE are identified by colours, each colour represents a cluster.

As expected, since BIRCH uses a centroid-based hierarchical clustering algorithm for clustering the pre-clustered points, it could not find all the clusters correctly. It splits the larger cluster while merging the others. In contrast, the CURE algorithm succeeds to generate majority of clusters but it still fails to discover all the clusters. Our Distributed Clustering Algorithm successfully generates all the clusters with the default parameters setting in section IV, as it is shown in Fig 3, after merging the local clusters, we generated five final clusters.

C. Comparison of DISCBK\(^1\) Execution Time to BIRCH and CURE

The goal of our experiment in this subsection is to demonstrate the impact of using the combination of parallel and distributed architecture to deal with the limited capacity of a node in the system and tree topology to accelerate the speed of computation.

Fig 4. Comparison to BIRCH and CURE.

Fig. 4 illustrates the performance of our algorithm compared to BIRCH and CURE as the number of data points is increased from 100000 to 500000 the number of clusters and their shapes are not altered. Thus, for our algorithm we consider two values for the number of nodes in the system: N=100 and N=200, in order to show the effectiveness of the parallelisation phase of the algorithm. The execution times do not include the time for display the clusters since these are approximately the same for the three algorithms.

As the graphs demonstrate, our algorithm’s execution times are always lower than CURE’s and BIRCH’s. In addition, increasing in number of nodes in the system further improves our running times. Finally, as the number of points increases, our execution times increase slowly. In contrast, the executions times for BIRCH’s pre-clustering algorithm increases much more rapidly with increasing dataset size. This is because BIRCH scans the entire database and uses all the points in the dataset for pre-clustering. Thus, the above results confirm that our distributed clustering algorithm is very efficient compared to both BIRCH and CURE.

D. Scalability

The goal of the scalability experiments is to determine the effects of increasing in number of nodes in the system on execution times. The dataset contains 1 000 000 points. The results for varying number of nodes are shown in Fig 5, our algorithm took only a few seconds to cluster 100000 points in a distributed system that contains over 500 nodes. Thus, the algorithm can comfortably handle high-dimensional data because the computations are performed in parallel.

---

\(^1\) DIStributed Clustering Algorithm Based on K-means
In the execution times, we do not include the time for the display phase. In Fig 5, we plot the execution time for the algorithm as the number of nodes increases from 10 to 600. The graphs confirm that the computational complexity of our algorithm is logarithmic, with respect to the dataset size.

VI. CONCLUSIONS

In this paper, we present a new approach for distributed clustering on spatial datasets. In this approach, local models are generated by executing K-Means algorithm in each node, and then the local results are merged to build the global clusters. The local models are extracted from the local datasets so that their sizes are small enough to be exchanged through the network.

Besides, we recover local datasets from their local models and then merge them together to build the new global models. Preliminary results of this algorithm are also presented and discussed. They also show the effectiveness of our approach either on quantity of clusters generated or the execution time compared to BIRCH and CURE algorithms. Furthermore, they demonstrate that the algorithm not only outperforms existing algorithms but also scales well for large databases without sacrificing clustering quality. Our new method is different from current distributed clustering models presented in the literature as it regenerates the global result from mining local dataset using K-means algorithm with different K for each node in the system. Most of the current methods are based on exchanging the whole clusters to build the global one.

A more extensive evaluation is ongoing. In the future we intend to execute our Algorithm with different large real world datasets in order to prove its robustness.

ACKNOWLEDGMENT

The research and the publication were supported by the European Union project IGIT titled as “Integrated geo-spatial information technology and its application to resource and environmental management towards the GEOSS” (grant agreement PIRSES-GA-2009-247608). We would like to thank our researchers for making available their scientific results for this analysis.

REFERENCES


X. J. P. Jiawei Han, Y. Micheline Kamber, Data Mining Concept and Techniques 2nd edition. Morgan Kaufmann, 6 April 2006.


