New Approach for Distributed Clustering

Souhila Ghanem[#1], Tahar Kechadi[*2], A.Kamel Tari[#]

[#]*Department of Computer Science, University of Béjaia*
Targa Ouzemour, Bejaia, Algeria
[1]souhila.ghanem@gmail.com

[*]*School of Computer Science and Informatics*
University College Dublin, Belfield, Dublin 04, Ireland
[2]Tahar.kechadi@ucd.ie

***Abstract*— Nowadays the data collections are huge and in most cases do not reside in a centralised location. The latter complicates the task of traditional data mining techniques, as datasets are distributed and often heterogeneous. In this paper we propose a distributed approach based on the aggregation of models produced locally. The datasets will be processed locally on each node to produce clusters from local data then, we construct global clusters hierarchically. The aim of this approach is to minimise the communications, maximise the parallelism and load balance the work among different nodes of the system, and reduce the overhead due to extra processing while executing the hierarchical clustering. This technique is evaluated and compared to the sequential version using benchmark datasets and the results are very promising.**

***Keywords*— Data Mining, Distributed Data Mining, Clustering, OPTICS**

I. INTRODUCTION

The large amounts of data produced nowadays pose the problem of storage, preservation, and analysis. For instance, many companies have databases in the order of terabytes. However, given the size of these collections, centralised data mining techniques have major difficulty in treating them, because they require a high capacity of storage, large computing speeds, and these data are often distributed in nature. The sequential data mining methods generally assume that all datasets are available in one centralised data centre. In a distributed environment this assumption is not tenable. The development of parallel and distributed techniques becomes essential to ensure accurate solutions at a very reasonable time.

The traditional distributed algorithms require that a large number of data be transferred to a central site, which leads to very high costs of communication. This has motivated the development of new distributed approaches, such as distributed multi-stage clustering approaches [1], which limit the costs of communications by exchanging only the representatives of clusters processed locally in each node. This method uses approximation techniques to determine automatically the number of local clusters [2-5]. In this paper, we propose distributed approach for density-based clustering algorithms, such as OPTICS.

Our approach tries to inherit the advantages of the previous distributed algorithms and provides solutions to the problems from which they suffer. We keep the concept of aggregating local models, which avoids the exchange of large amounts of data between sites. Our approach takes the responsibility of choosing the number of clusters, which is not straightforward, and it does not require approximation techniques to determine it. Each node determines the clusters based on its local data sets and calculates their representatives. The representatives of each cluster are then exchanged to calculate global clusters. Global clusters are determined by testing the overlap between clusters of different sites.

II. THE CLUSTERING

Clustering is a method of extracting knowledge from data collections. Most of the clustering algorithms require input parameters, which are difficult to determine, but have a significant effect on the result of clustering. For example, partitioning algorithms that are used to divide data into k clusters, k is an input parameter. To determine k we need certain knowledge of the domain, which is not available for many applications. Unlike partitioning algorithms, hierarchical algorithms do not need k as an input. However, a termination condition must be defined by indicating when the process of aggregating or division must be stopped. One solution for choosing appropriate values for these parameters is to test a range of values, but the number of possible values may be infinite [11]. This is still one of the major problems of the existing clustering algorithms. A solution is usually specific to each algorithm and application.

III. THE DISTRIBUTED CLUSTERING

This research area attracted significant interest, but few works have been dedicated to distribute clustering [7-8]. Among these works, one can site the distributed multi-stage approach [1] and a distributed version of DBSCAN [9].

In this paper we propose a distributed approach for the OPTICS algorithm and it consists of three important steps:
(1) The local clustering.
(2) Determination of the representatives of local clusters.
(3) Determination of the global models based on the local models.

A. The Local Clustering

The local clusters are generated by executing the OPTICS algorithm on local datasets of each node. The algorithm has the following characteristics: first, the reachability-plot provided by the algorithm is insensitive to input parameters (ε and MinPts). The values of these parameters must be large enough to get reasonably good results. Secondly, the obtained order is independent of the data dimensions. In the following we describe the OPTICS algorithm.

1) OPTICS Algorithm

The OPTICS (Ordering Points To Identify Clustering Structure [6]) algorithm constitutes an extension of the DBSCAN algorithm [12]. It establishes an order in which objects are treated. The information, which will be used to generate this ordering, consists of two values for each object: the *core-distance* and the *reachability-distance*. They are describes in the following:

The *core distance* of an object *p* is the distance ε' between the point *p* and its MinPts closer neighbours if they do not exceed ε, otherwise it is undefined.

The *reachability-distance* of an object *p* with respect to another object *o* is the smallest distance such that *p* is directly density reachable from *o* if *o* is a core object.

The OPTICS algorithm generates an ordering of all data: the points are ordered according to their density with respect to both parameters ε and MinPts. The structure of clustering can be visualised by plotting the values of the reachability distance r_i according to the orderly points o_i. Figure 1 shows a graphical representation of the order provided by the OPTICS algorithm. From this graph we can easily identify the parameters, in particular, the number of clusters is equal to five in this example.

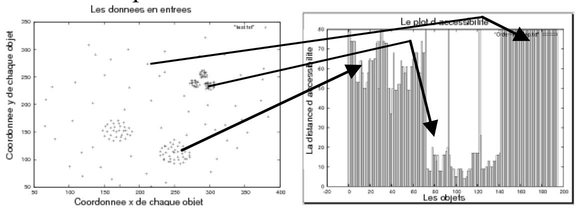

Fig. 1 Reachability-plots for a data set with clusters of different sizes, densities

The method proposed in [6] to determine clusters from the reachability-plot consists of determining the up and down areas. The first step of this method assumes that any combination of up and down areas forms a cluster and the second step tries to filter clusters with respect to the value of MIB[1] which leaves many groups which are not really clusters especially in the case of the reachability-plot with several peaks, (Fig. 2)

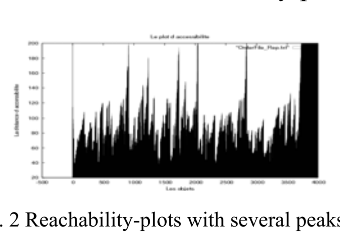

Fig. 2 Reachability-plots with several peaks

2) New Method of Calculation of Clusters

Our method also consists in calculating the up and down areas [6], but we define a threshold equal to 80 % of the biggest up or down areas. It consists of determining a right height equal to 80 % of the biggest up or down areas. Any combination of areas which begins with a down area and ends by an up area and whose height is greater or equal to the threshold is considered as a cluster. The points, which are between a down and up areas and which verify the condition of minimum threshold belong certainly to the cluster. Therefore, we will select among these points those belonging to the cluster.

To automate this process we need to define a height that can constitute the minimum threshold. By experiment a point of a down area belongs to the cluster if the double of the height of its successor is less than RDFPD (Reachability Distance of First Point of Down area). In other words, a point x_i of an up area belongs to the cluster if $2*x_i$ is lower than RDEPU (Reachability Distance of the first point after the End Point of Up area). The algorithm is given in the following:

SetSteepArea: contains the beginning of each down area and the end of each up area. **SetFirstEnd:** contains the values of reachability distance of the beginning of each down area and the first point after the end of each up area, which forms the cluster. Let **n** be the number of objects in the dataset.

Step 1:

```
index =0, SetSteepArea = $\varnothing$, SetFirstEnd = $\varnothing$ ;
r (p): the reachability-distance of the point p;
While index < n do.
  (1) If a new steep down area starts at index, add it to
        SetSteepArea,
        First =r (index); continue to the right of it.
  (2) If a new steep up area starts at index, combine it with
        every steep down area in SetSteepArea, check each
        combination for fulfilling the cluster conditions [6].
        If the conditions are checked, End = r(index+1) ;
          Add the value of First and End to SetFirstEnd and
          continue to the right of this steep up region.
  (3) Otherwise, increment index.
```

Step 2:

max = 80% of the largest peak in SetFirstEnd.

Comparison of values in SetFirstEnd to max and deletion of the values lower than max and also remove up and down areas corresponding in SetSteepArea.

After filtering SetSteepArea, The remaining values really correspond to points belonging to up and down areas which form clusters.

Etape 3:

Add points between two areas down and up in SetSteepArea to clusters. The filtering is done as follows:

```
For all point $x_i$ of an up area in SetSteepArea do
If $2*x_i \leq$ RDFPU then add $x_i$ to cluster
For all point $x_i$ of a down area in SetSteepArea do
If $2*x_{i+1} \leq$ RDEPD then add the point $x_i$ to the cluster
```

We implemented the OPTICS algorithm on several sets of data by using the new method based on the definition of the threshold to determine clusters from the reachability-plot. Our approach gives very good results and here are some examples.

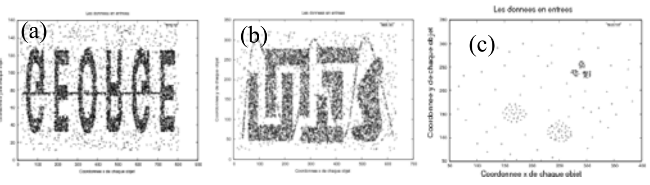

Fig. 3 (a) (b) (c): representation of initial data

[1]MIB: (maximum in between values) is the maximum value between the current index in the order of clusters and the last two peaks [6].

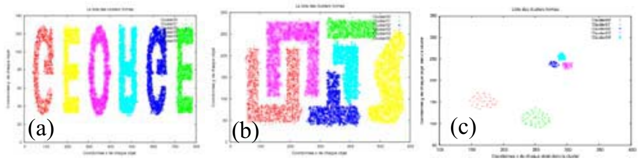

Fig. 4 (a) (b) (c): Clusters formed after execution of the algorithm OPTICS.

B. Local models

After performing clustering on each site, we need representatives witch describe the result of local clustering. We must find a compromise between the two following constraints:

- Having a small number of representatives
- Having a clear description of each local cluster.

Our solution consists of calculating the borders of each local cluster. The borders can serve as very good representatives.

C. Determination of Borders

The search for the borders of clusters is a difficult problem, because clusters may have concave forms with holes, heterogeneous densities and also the data can have several dimensions. Clusters represented in the Figure 7, contain regions with heterogeneous densities (presence of holes). To avoid these problems, we opted for the method described in [10]. This method is based on the creation of the balance vector.

Balance Vector: Let i be the point for which we want to find the balance vector. Let j be the set of all the points in the neighbourhood of i. The balance vector is defined as the mean of all the vectors for the points in j to i.

$$\vec{v_i} = \frac{\sum_{p_j \in N_\epsilon(p_i)} p_i - p_j}{|N_\epsilon(p_i)|}$$

If a point is a border point, there should be no points in its surroundings in the direction of the balance vector. This feature permits to separate border points and internal points. The arrow in red shows the balance vector.

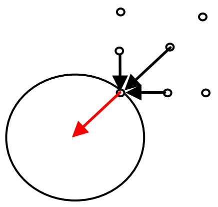

Fig.6: Representation of the surrounding in the direction of the balance vector

Let C be the cluster and B(C) be the border points of the cluster C, $\rho>0$ a positive parameter and n_p the norm of the balance vector of the point p.

Pseudo code (1)

```
for all points i in C do
i' = i + ρn_i
  for all points j ≠ i in C do
    if the dist(i'; j) < ρ then
      Discard point i from B(C)
    end if
  end for
end for
```

Pseudo code (2nd Pass) (2)

```
for all points i in B(C) do
  for all points j ≠ i in B(C) do
    if the dist(i; j) < ρ then
      Discard point j from B(C)
    end if
  end for
end for
```

The algorithm (1) gives good result even for clusters with regions of different densities and holes, but the borders are denser when the initial datasets are denser. For this we apply the second part of the algorithm (2), which can eliminate some points in dense regions. Figure 7 shows the clusters and Figure 8 shows the borders of the clusters calculated with this method.

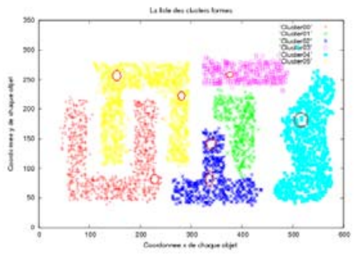

Fig. 7: Clusters with holes

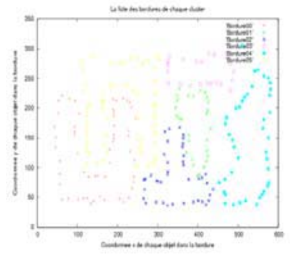

Fig. 8: Borders for every cluster.

D. Regeneration of Clusters

The obtained borders are sent to the sites of aggregates. From these borders we will regenerate the clusters. For this we need the sign of the balance vector of each point border and the average of the distances between points in the cluster (ADP), which is used as distance between two regenerated points. This distance can be adapted to the status of each cluster (if it is dense or not). The sign of the balance vector of a point i is calculated with respect to a new landmark. If the vector is oriented to the inside of i then the sign is negative otherwise it is positive (see Figure 10b).

The regeneration method presented in [13] consists of selecting all the points borders received for which the sign of the balance vector is negative. From each of these points, other points will be regenerated. This method does not often give good results because we regenerate the clusters only with the borders points received, the number of points constituting

the border is small compared to the number of real points at the border, this allowed to minimise the communications, but the regeneration from these points does not give regenerated clusters with the same density as the original clusters, because we ignore points which are between the border points and from which we must also regenerate other points. On the other hand the end of regeneration should meet a positive point and only if we have not all the positive points in the border, else this can regenerate points, which are not part of the cluster and the process may not end. Figure 10c shows this difficulty.
To solve this problem, we first need to regenerate all the points of the border and then start the procedure of regenerating the clusters. The distance between a border point and a regenerated point or between other regenerated points is equal to ADP (the average of the distances between points in a cluster (see Figure 10d).

E. Regeneration of Borders

To have a complete border, which allows regenerating similar clusters as close as possible to the original clusters we precede as follows:

```
Regeneration (B(C) ) // B(C) : Borders of the cluster C,
T[i] : Table indicating whether the objects in the cluster are
treated or not, initialised to 0, i={1,.. , nb}, nb : number of points
borders in B(C).
While ∃ a point in B(C) untreated
  Selected an untreated point p in B(C) (ie T[p]=0)
  T[p]=1 ;
  Search for the closest point of p untreated noted p1 (p ≠p1)
  While (T[p1] !=2) do
      if p1 is (-) then
          Regeneration negative points between (p, p1) at a distance ADP
      else if p1 is (+)
          Regeneration positive points between (p, p1) at a distance ADP
      if (T[p1] =0)
          p=p1 ;
          Search for the closest point of p noted p1 (T[p1] ≠2)
          T[p]=2
      else if (T[p1] ==1)
          T[p1]=2
      End if
  End while
End while
```

Fig. 9 Algorithm for regeneration borders.

To regenerate a cluster similar to the original cluster from the new borders, we select all the border points whose sign of the balance vector is negative. From each of these points, other points will be regenerated until we meet positive points in the neighbourhood. The distance between a border point and a regenerated point or between other regenerated points is equal to ADP. During the regeneration we regenerate points that belong to the inside of the cluster (see Figure 10 e).

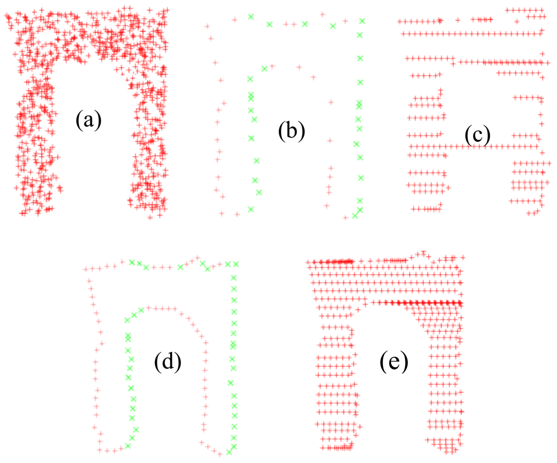

Fig.10. (a) Original cluster (b) Borders. (c) Regeneration of the cluster from the borders of the Figure 10 (b). (d) Regeneration borders. (e) Regeneration of the cluster from the borders of Figure 10(d).

In Figures10 (b) and (d): (+) the sign of the balance vector is negative. (x) The sign of the balance vector is positive.

F. Distribution model

Each node terminating the sequential execution of the algorithm on local data must be able to decide with which other sites it will make aggregations. For this, we defined a topology for the nodes of the system. The nodes are arranged in a ring in which each node has two neighbours: one on the left and the other on its right. Each node must have information from the left hand side. This means that every node in the left must send its representatives (borders) to its right neighbour. When a node receives the representatives of clusters of its left neighbour, it looks if there is an overlap with its clusters. If this is the case, the new representatives of clusters will be formed and will be sent to the right neighbour, otherwise it send all the representatives witch are in its left to its right neighbour.

G. Global models

Once the representatives are transferred to the aggregate nodes, we test whether there are clusters that overlap at the same time as we regenerate the clusters. The overlapping clusters are merged. We say that two clusters overlap if there is a covering between the two clusters or if there are more than MinPts border points of a cluster at a distance less than ADP to another cluster. The borders of new clusters formed are calculated during the test of overlapping procedure. The number of border points formed will be high and to minimise the communications we will send only the border points (Section III (3)), which represent well the new cluster in another node. The average of the core distances that correspond to the newly formed cluster will be equal to the average of the core distances of the merged clusters. The average of the distances between points for the new cluster is equal to the average of the averages of the distances between the points of the merged clusters.

In the following, we perform aggregation between clusters of node S_1 and those of the node S_2. We regenerate the borders of the clusters and we test the overlap between them.

S_1 and S_2 are two nodes, $Cluster_1$: cluster of the site S_1, $Cluster_2$: a cluster of the site S_2.

```
Aggregation (Representatives of S_1, Representatives of S_2)
for any cluster noted Cluster_1 of the site S_1 do
    Regeneration (Borders of Cluster_1)
    for any cluster noted Cluster_2 of the site S_2 do
        Regeneration (Borders of Cluster_2)
        if Cluster_1 is in the left of the Cluster_2 then
            Overlap-test (Borders-Cluster_1, Border-Cluster_2)
        else
            Overlap-test (Borders-Cluster_2, Border-Cluster_1)
        end if
    end for
end for
```

Fig. 11 Aggregation procedure

The procedure **Overlap-test** () allows to regenerate clusters, to test the overlap between them and form the new borders of the new cluster.

B1, B2: Borders of the clusters C1 and C2. NC: new cluster, NB: new borders, ADP(NC): the average distances between points in the cluster NC: ACD(NC): average of cores distance of the cluster in NC.

```
Overlap-test (B1, B2)
merge = 0 ;
for all point i in the borders of the cluster C1 do
    variable = true; nb=0 ;
    while (variable=true) then
        search neighbours of i in B2
        if all neighbours noted j of i in B2 ≠∅ do
            if (sign(i) ='+') and (sign (j) ='-') then //Testing border
                nb=nb+1 ;
                save i and j in tab1.
            else //--------------------- overlap ---------------------
                NB= B1+B2. // NB: the new borders formed
                variable=false; merge=1;
                if ((sign (i) ='-') and (sign (j) ='-'))
                    Eliminate points of intersection j in NB.
                    Finding neighbours of j in B2 which have
                            the sign (+)
                    while the neighbours of j =∅ Then
                        Search the neighbours of j noted l in B1
                        if the neighbours of l ≠ ∅ then
                            Delete the point l in the NB;
                        else Regeneration of a point j
                            Search the neighbours of j in B2
                    end while
                    if all neighbours k of j ≠ ∅ then
                        Eliminate the neighbours of k in NB
                    end if
            end si
        else  if sign(i='-')
            Regeneration of a point i
            Search the neighbors of i in B1 which have the sign (+)
            if (neighbor (i)=∅) then variable=true.
            else Variable=false.
        end if
        if (sign(i)='+') then variable=false;
    end while
end for
if ((nb≥ MinPts) and (merge=0)) then // merge the two clusters
    NB= B1+B2, merge=1;
    Eliminate elements of tab1 in NB
end if
if (merge==1)
    ADP(NC)=ADP(C1)+ ADP(C2)/2
    ACD(NC)=ACD(C1)+ ACD(C2)/2
    Filtering (NB)// apply the part 2 of the algorithm of calculates
        border (section III (3))
end if
```

Fig.12 The procedure **Overlap-test**.

IV. Evaluation and Experimentation

We evaluated our approach on several datasets on two dimensions. We used a distributed system that consists of three machines, and we run two experiments:

In the following we have two sets of data; each is distributed on three sites. The Figure 13 represents the initial data: points in red belong to the node 1, the points in blue are hosted on node 2 and finally the points in green stored on the node 3. Figures 14 and 15 represent clusters formed in each of the three nodes for the first and second datasets respectively. In Figure 16 we can see the results of the aggregation of clusters for both datasets. We can notice that the results are the same for both execution models; sequential and distributed.

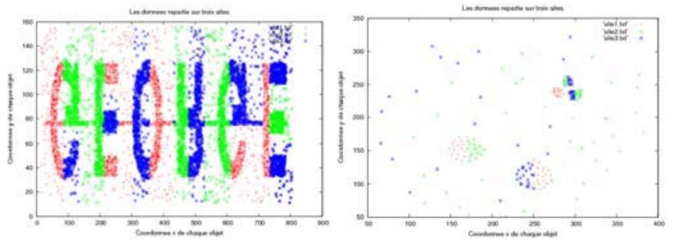

Fig. 13 Two sets of data, each is distributed over three machines

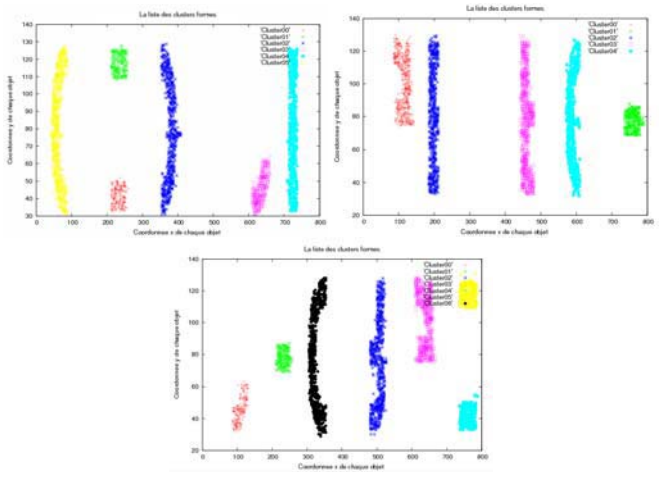

Fig.14 The clusters formed in each site for the first dataset

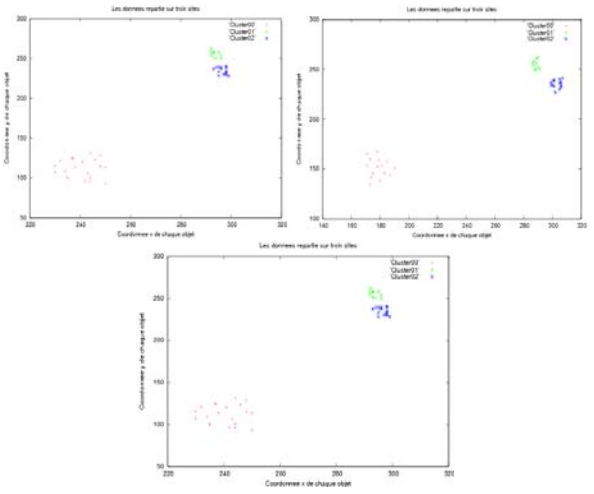

Fig.15 Clusters formed in each site for the second dataset

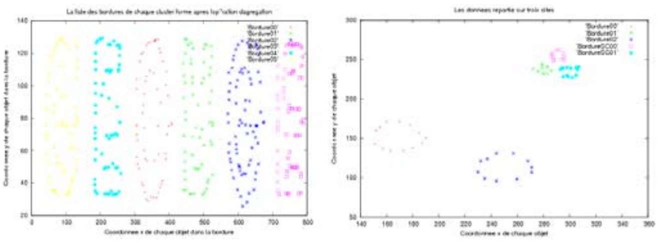

Fig. 16 Results of aggregation on three nodes for two datasets.

Generally, the sequential OPTICS algorithm gives good results if the extraction method of clusters from the reachability-plot was efficient. In some particular datasets the proposed method can give results that do not correspond to the expected results. Our approach overcomes this problem because to do the aggregation, we test the overlap between clusters and there is an overlap between any two clusters, the two clusters will be merged.

Our approach also allows gaining much in response time compared to the sequential algorithm. The response time of OPTICS is heavily dominated by the execution time of the ε-neighbourhood queries, which must be performed for each object in the dataset, i.e. the execution time is given by O(*n** execution time of an ε-neighbourhood query). In our distributed approach the number of objects is distributed among the system nodes. This reduces considerably the execution time of a ε-neighbourhood request and the calculations are performed in parallel. After the reconstruction of borders, the execution time of the aggregation algorithm is negligible compared to that of the algorithm OPTICS and that of the calculation of the borders because the number of borders in a cluster represents hardly 3 % of the total number of objects in a cluster [10].

V. Conclusions

In this paper we proposed and study an efficient distributed OPTICS algorithm and most of the existing parallel and distributed versions are based on sequential version that generates a lot of communications and thus affect their performance. The OPTICS algorithm has great difficulty in choosing the right number of clusters. Our approach automatically determines this number. It consists of performing a local clustering, selecting the representatives of each cluster and sending them to aggregate nodes. At each node we regenerate the borders and clusters and test overlap between them.

In our experimental evaluation, we showed that the new distributed approach gives the same results as the sequential execution on the whole dataset. In the near future we will try to extend this approach to design distributed versions of other types of data mining clustering techniques.

References

[1] L.M. Aouad, N-A. Le-Khac and M-T. Kechadi, A Multi-Stage Clustering Algorithm for Distributed Data Mining Environments, School of Computer Science and Informatics University College Dublin –Ireland, , 2008

[2] Y Mingjin and Y. Keying. Determining the number of clusters using the weighted gap statistic. Biometrics, 63(4), 2007.

[3] Y Mingjin and Y. Keying2. Determining the number of clusters using the weighted gap statistic. Biometrics, 63(4), 2007

[4] R. Tibshirani, G.Walther, and T. Hastie. Estimating the number of clusters in a dataset via the Gap statistic. Technical report, Stanford University, March 2000.

[5] R. Tibshirani, G. Walther, and T. Hastie, Estimating the Number of Clusters In a Data Set Via the Gap Statistic , Journal of Royal Statistical Society, 2001

[6] M. Ankerst, M.M. Breunig, H-P. Kriegel, J. Sander, OPTICS: Ordering Points To Identify the Clustering Structure, Institute for Computer Science, University of Munich Oettingenstr. 67, D-80538 Munich, Germany, 1999.

[7] L.M. Aouad · N-A. Le-Khac and M-T. Kechadi, Performance study of distributed Apriori-like frequent itemsets, 2009

[8] L.M. Aouad, N-A. Le-Khac and M-T. Kechadi, Distributed Frequent Itemsets Mining in Heterogeneous Platforms, Journal of Engineering, Computing and Architecture 2007.

[9] E. Januzaj, H-P. Kriegel, M. Pfeifle, Towards Effective and Efficient Distributed Clustering, Institute for Computer Science University of Munich Germany, 2003.

[10] A. Piras and M-T. Kechadi, Distributed Clustering – Hierarchical Clustering on Grids, Journal of Parallel Computing, (in press) 2010.

[11] J-P Nakache, J Conais, Approche pragmatique de la classification, Editions Technip, Paris, 2005.

[12] M Ester., H.-P Kriegel, J.Sander, X Xu.: A Density- Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR, AAAI Press, 1996, pp.226-231.

[13] S. Ghanem, M.T. Kechadi, A.K. Tari, Nouvelle approche pour le clustering distribué, 2010