Lightweight Clustering Technique for Distributed Data Mining Applications*

Lamine M. Aouad, Nhien-An Le-Khac and Tahar M. Kechadi

School of Computer Science and Informatics
University College Dublin - Ireland
{lamine.aouad,an.le-khac,tahar.kechadi}@ucd.ie

Abstract. Many parallel and distributed clustering algorithms have already been proposed. Most of them are based on the aggregation of local models according to some collected local statistics. In this paper, we propose a lightweight distributed clustering algorithm based on minimum variance increases criterion which requires a very limited communication overhead. We also introduce the notion of distributed perturbation to improve the globally generated clustering. We show that this algorithm improves the quality of the overall clustering and manage to find the real structure and number of clusters of the global dataset.

1 Introduction

Clustering is one of the fundamental technique 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 underlying data. This is a difficult task in unsupervised knowledge discovery and there is already a large amount of literature in the field ranging from models, algorithms, validity and performances studies, etc. However, there is still several open questions in the clustering process including the optimal number of clusters, how to assess the validity of a given clustering, how to allow different shapes and sizes rather than forcing them into balls and shapes related to the distance functions, how to prevent the algorithms initialization and the order in which the features vectors are read in from affecting the clustering output, and how to find which clustering structure in a given dataset, i.e why would a user choose an algorithm instead of another. Most of these issues come from the fact that there is no general definition of what is a cluster. In fact, algorithms have been developed to find several kinds of clusters; spherical, linear, dense, drawnout, etc.

* This study is part of ADMIRE [15], a distributed data mining framework designed and developed at University College Dublin, Ireland.

In distributed environments, clustering algorithms have to deal with the problem of distributed data, computing nodes and domains, plural ownership and users, and scalability. Actually, moving the entire data to a single location for performing a global clustering is not always possible due to different reasons related to policies or technical choices. In addition, the communication efficiency of an algorithm is often more important than the accuracy of its results. In fact, communication issues are the key factors in the implementation of any distributed algorithm. It is obvious that a suitable algorithm for high speed network can be of little use in WAN-based platforms. Generally, it is considered that an efficient distributed algorithm needs to exchange a few data and avoids synchronization as much as possible.

In this paper, we propose a lightweight distributed clustering technique based on a merging of independent local subclusters according to an increasing variance constraint. This improves the overall clustering quality and finds the number of clusters and the global inherent clustering structure in the whole dataset. However, a proper maximum increasing value has to be selected. This can be deduced from the problem domain or found out using various methods. The rest of the paper is organized as follows, the next section surveys some previous parallelization and distribution efforts in the clustering area. Then, section 3 presents our distributed algorithm. Section 4 shows some experimental results and evaluations, and highlights directions for future work and versions. Finally, section 5 concludes the paper.

2 Related Work

This section survey some works in parallel and distributed clustering, and discusses the latest projects and proposals especially regarding grid-based approaches.

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. Details about these algorithms is out of the purpose of this paper, we refer the reader to [8] and [19]. Many parallel clustering versions based on these algorithms have been proposed [3][4][5][6][13][20], etc. In [3] and [13], message-passing versions of the widely used k-means algorithm were proposed. In [4] and [20], the authors dealt with the parallelization of the DBSCAN density based clustering algorithm. In [5] a parallel message passing version of the BIRCH algorithm was presented. In [6], the authors introduced a parallel version of a hierarchical clustering algorithm, called MPC for Message Passing Clustering, which is especially dedicated to Microarray data. Most of the parallel approaches need either multiple synchronization constraints between processes or a global view of the dataset, or both.

The distributed approaches are different, even many of the proposed distributed algorithms are based on algorithms which were developed for parallel systems. Actually, most of them typically act by producing local models followed by the generation of a global model by aggregating the local results. The processes participating to the computation are independent and usually have the same computation level. After this phase, the global clustering is obtained based on only local models, without a global view of the whole dataset. All these algorithms are then based on the global reduction of so-called sufficient statistics, probably followed by a broadcast of the results. Some works are presented in [9][10][11][12][21], mostly related to the k-means algorithm or variants and the DBSCAN density based algorithm.

On the other hand, grid and peer-to-peer systems have emerged as an important area in distributed and parallel computing[1]. In the data mining domain, where massive datasets are collected and need to be stored and performed, the grid can be seen as a new computational and large-scale support, and even as a high performance support in some cases. Some grid or peer-to-peer based projects and frameworks already exist or are being proposed in this area; Knowledge Grid [2], Discovery Net [7], Grid Miner [14], ADMIRE [15], etc. Beyond the architecture design of these systems, the data analysis, integration or placement approaches, the underlying middleware and tools, etc. the grid-based approach needs efficient and well-adapted algorithms. This is the motivation of this work.

3 Algorithm description

This section describes the distributed algorithm and gives some formal definitions. The key idea of this algorithm is to choose a relatively high number of clusters locally (which will be called subclusters in the rest of the paper), or an optimal local number using an approximation technique, and to merge them at the global level according to an increasing variance criterion which require a very limited communication overhead. All local clustering are independent from each other and the global aggregation can be done independently, from and at any initial local process.

3.1 Algorithm foundations

At the local level, the clustering can be done by different clustering algorithms depending on the characteristics of the data. This includes k-means, k-harmonic-means, k-medoids, or their variants, or using the statistical interpretation with the expectation-maximization algorithm which finds clusters by determining a mixture of Gaussian distributions. The merging process of the local subclusters at the global level exploits locality in the feature space, i.e. the most promising candidates to form a global cluster are subclusters that are the closest in the feature space, including subclusters from the same site. Each participating process can perform the merging and subtract the global clusters formation, i.e. which subclusters are subject to form together a global cluster.

Before describing the algorithm itself, we first give developments on some used notions. A global cluster border represents local subclusters at its border.

[1] The designation 'parallel' is used here to highlight the fact that the computing tasks are interdependent, which is not necessarily the case in distributed computing.

These are susceptible to be isolated and added to another global cluster in order to contribute to an improvement of the clustering output. These subclusters are referred to as perturbation candidates. Actually, the initial merging order may affect the clustering output, as well as the presence of non well-separated global clusters, this action is intended to reduce the input order impact. The global clusters are then updated. The border is collected by computing the common Euclidean distance measure. The b farthest subclusters are then the perturbation candidates, where b is a user predefined number which depends on the chosen local number of clusters. Furthermore, multi-attributed subclusters are naturally concerned by this process.

The aggregation part of the algorithm starts with $\sum_{i \in s} k_i$ subclusters, where s is the number of sites involved and k_i, for $i = 1, ..., s$, are the local numbers of clusters in each site. Each process has the possibility to generate a global merging. An important point here is that the merging is logical, i.e each local process can generate correspondences, i.e. labeling, between local subclusters, without necessarily constructing the overall clustering output. That is because the only bookkeeping needed from the other sites are centers, sizes and variances. The aggregation is then defined as a labeling process between local subclusters in each participating site. On the other hand, the perturbation process is activated if the merging action is no longer applied. b candidates are collected for each global cluster from its border, which is proportional to the overall size composition as quoted before. Then, this process moves these candidates by trying the closest ones and with respect to the gain in the variance criterion when moving them from the neighboring global clusters. In the next section we will formally define the problem, notions and criterions.

3.2 Definitions and notations

This section formalizes the clustering problem and the notions described in the previous section. Let $X = \{x_1, x_2, ..., x_N\}$ be a dataset of N elements in the p-dimensional metric space. The problem is to find a clustering of X in a set of clusters, denoted by $C = \{C_1, C_2, ..., C_M\}$. The most used criterion to quantify the homogeneity inside a cluster is the variance criterion, or sum-of-squared-error criterion:

$$S = \sum_{i=1}^{M} E(C_i)$$

where

$$E(C) = \sum_{x \in C} \parallel x - u(C) \parallel^2$$

and

$$u(C) = \frac{1}{|C|} \sum_{x \in C} x$$

is the cluster mean.

Traditional constraint used to minimize the given criterion is to fix the number of clusters M to an a priori known number, as in the widely used k-means, k-harmonicmeans, k-medoids or its variants like CLARA, CLARANS, etc. [16][19][22]. This constraint is very restrictive since this number is most likely not known in most cases. However, many approximation techniques exist such as the gap statistic which compares the change within cluster dispersion to that expected under an appropriate reference null distribution [17], or the index due to Calinski & Harabasz [1], etc. This can be used locally as quoted before. The imposed constraint here states that the increasing variance of the merging, or union, of two subclusters is below a dynamic limit $\sigma_{i,j}^{max}$. This parameter is defined to be twice the highest individual variance from subclusters C_i and C_j [18].

The border B_i of the global cluster C_i is the set of the b farthest subclusters from the generated global cluster center. Let $SC_i = \{scc_1, scc_2, ..., scc_{n_i}\}$ be the set of the n_i subclusters centers merged into C_i. B_i is defined as:

$$B_i(b) = F(u(C_i), b, C_i, SC_i)$$

where

$F(u(C_i), b, C_i, SC_i) =$

$$\begin{cases} fsc(u(C_i), b, C_i, SC_i) \ \cup \ F(u(C_i), b-1, C_i, SC_i - fsc(u(C_i), b, C_i, SC_i)), \ b > 0 \\ \emptyset, \ b = 0 \end{cases}$$

$fsc(u(C_i), b, C_i, SC_i)$ are the b farthest subclusters centers from $u(C_i)$:

$$fsc(u(C_i), b, C_i, SC_i) = arg \max_{x \in SC_i} Euclidean(x, u(C_i))$$

These sets are then performed once the merging is no longer applied, and as quoted before, the multi-attributed subclusters will belong to it.

3.3 summarized algorithm

According to the previous definitions and formalism, the Algorithm 1 summarize the proposed approach. In the first step, local clustering are performed on each local dataset, the local number of clusters can be different in each site. Then, each local clustering in a site i gives as output k_i subclusters identified by a unique identifier, $cluster_{\{i,number\}}$ for $number = 0, ..., k_i - 1$, and their sizes, centers and variances. At the end of local processes, local statistics are sent (5

Algorithm 1 Variance-based distributed clustering

```
Input: $X_i$ $(i = 1, \dots, s)$ datasets, and $k_i$ the number of subclusters in each site $S_i$
Output: $k_g$ global clusters, i.e the global subclusters distribution labeling
 1: for $i = 1$ to $s$ do
 2:    $LS_i = cluster(X_i, k_i)$
 3: end for
 4: $j = select\_aggr\_site()$
 5: for $i = 1$ to $s$ do
 6:    if $i \neq j$ then
 7:       $send(sizes_i, centers_i, variances_i, j)$
 8:    end if
 9: end for
    at site the aggregation site $j$:
10: while $var(C_i, C_j) < \sigma_{i,j}^{max}$ do
11:    $merge(C_i, C_j)$
12: end while
13: for $i = 1$ to $k_g$ do
14:    $find\_border(b, i)$
15:    $add\_multi\_attributed(i)$
16:    for $x = 1$ to $b$ do
17:       $j = closer\_global(x)$
18:       $var_{new} = var(C_i - C_x, C_j + C_x)$
19:       if $var_{new} < var$ then
20:          $label(x, j)$
21:          $var = var_{new}$
22:       end if
23:    end for
24: end for
```

- 9) to the chosen merging process j at step (4). Then, the subclusters aggregation is done in two phases; merging (10 - 12) and perturbation (13 - 24). In the latter phase, the border $B_i(b)$ is found (14 - 15), with $i \in k_g$, and b is a user defined parameter. For each $x \in B_i(b)$, the closet global cluster j is found and the new variance is computed. The actual perturbation, which still a labeling at the global level, is done if the new global variance is smaller (16 - 23). At the step (11), the new global statistics, namely the size, center and variance, are:

$$N_{new} = N_i + N_j$$

$$c_{new} = \frac{N_i}{N_{new}}\, c_i + \frac{N_j}{N_{new}}\, c_j$$

$$var_{new} = var_i + var_j + inc(i,j), \quad \forall C_i, C_j, i \neq j$$

where

$$inc(i,j) = \frac{N_i \times N_j}{N_i + N_j} \times Euclidean(C_i, C_j)$$

represents the increasing in the variance while merging C_i and C_j.

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 is efficient to detect well separated clusters and distribution with their effective distribution number. Otherwise, a clear definition of a cluster does not exist anymore. This is also an efficient way to improve the output for the k-means clustering and derivatives for example, without an a priori knowledge about the data or an estimation process for the number of clusters.

3.4 Performance analysis

The computational complexity of this distributed algorithm depends on the algorithm used locally, the communication time, which is a gather operation and the merging computing time:

$$T = T_{comp} + T_{comm} + T_{merge}$$

If the local clustering is a k-means, the complexity T_{comp} is of order $O(N_i k_i d)$, where d is the dimension of the dataset. The communication time is the reduction of $3d \sum_{i \in s} k_i$ elements. Actually, the aggregation process gathers local information in order to perform the merging. If t^i_{comm} is the communication cost for one element from site i to the aggregation process j then

$$T_{comm} = 3d \sum_{i \in s, i \neq j} t^i_{comm} k_i$$

Since k_i is much less large than N_i, the generated communication overhead is very small.

The merging process is executed a number of times, say u. This is the number of iterations until the condition $var(C_i, C_j) < \sigma^{max}_{i,j}$ is no longer applied. This cost is then equal to $u \times t_{newStatistcs}$, which corresponds to $O(d)$. This is followed by a perturbation process, which the cost is of order $O(bk_g k_i)$. Actually, since this process computes for each of the b chosen subcluster at the border of C_i, k_i distances for each of the k_g global clusters. The total cost is then:

$$T = O(N_i k_i d) + O(d) + O(bk_g k_i) + T_{comm}, \quad T_{comm} \ll O(N_i k_i d)$$

4 Experiments

In this section, we show the effectiveness of the proposed algorithm with some artificial and real datasets. We give a description of the data, the experimentation details and a discussion. As quoted before, the constraint parameter, i.e the maximum merging variance, is set up as twice the highest individual subcluster variance.

4.1 Data description

The first dataset is a generated random Gaussian distributions with 6000 samples. Figure 1 displays this dataset with an initial clustering using k-harmonicmeans

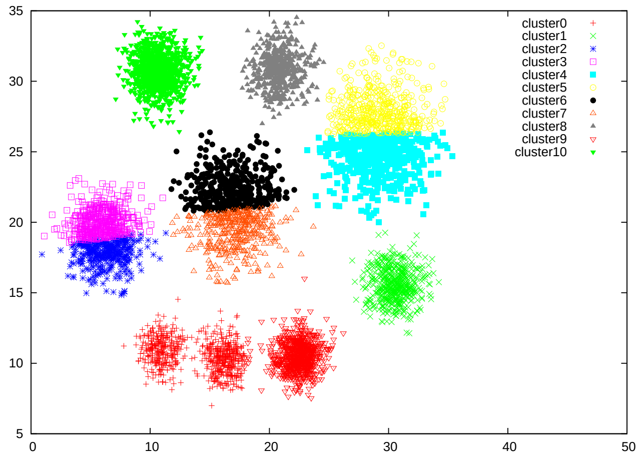


Fig. 1. Global k-harmonicmeans clustering using the gap statistic to find the optimal k of the dataset, $k = 11$.

and the gap statistic. The data was distributed in three sets as shown in Figure 2.

The second set is the well-known Iris dataset. It consists in three classes of irises (Iris setosa, Iris versicolor and Iris virginica) each characterized by 8 attributes and there is 150 instances. The set was randomly distributed as shown in Figure 4 (presented by the attributes "sepal area" and "petal area"). This figure shows also the initial local clustering using k-harmonicmeans with $k = 5$.

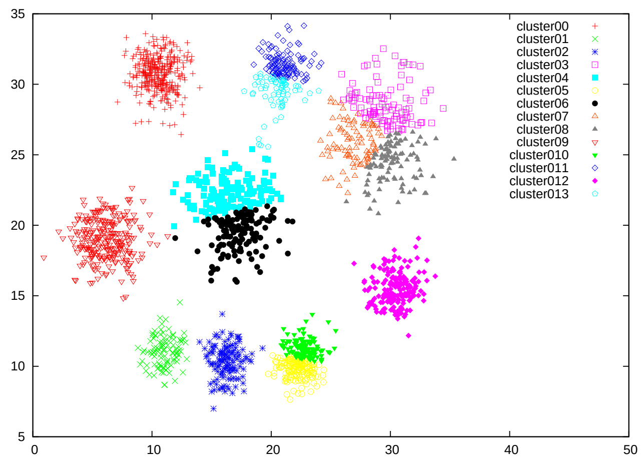
The last dataset is a census data available from the UC Irvine KDD Archive. It is derived from the one percent sample of the PUMS (Public Use Microdata Samples) person records from the full 1990 census sample. Our tests use a discretized version of this set. There are 68 attributes[2]. The set originally contains 2458285 records reduced to 1998492 after elimination of doubled records. The data is distributed over 7 processes.

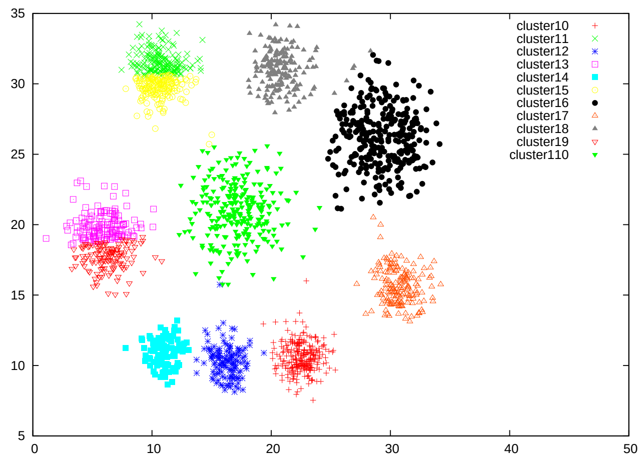
4.2 Evaluations and discussion

The merging output of the first dataset is shown in Figure 3. This result finds the right number of clusters and their distribution independently of the local used clustering algorithm and the number of subclusters. The local number of clusters found using the gap statistic is 14 for the first set and 11 for the two other sets (cf. Figure 2). The gap statistic based implementation of the expectation-maximization algorithm give the same clustering output.

The resulting global clustering for the Iris dataset, and a global k-harmonicmeans clustering using the entire dataset, are given is Figure 5. The algorithm manages

[2] The caseid is ignored during analysis. The list of attributes and the coding for the values can be found at `http://kdd.ics.uci.edu/`





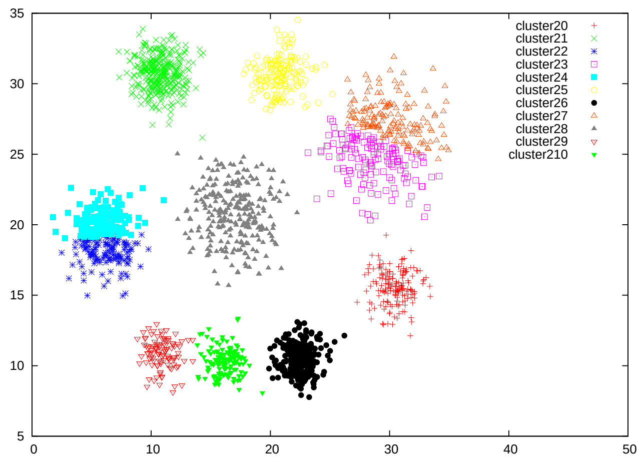


Fig. 2. Local k-harmonicmeans clustering in each process using the gap statistic to find the optimal number of clusters, $k_1 = 14,\ k_2 = 11,\ and\ k_3 = 11$.

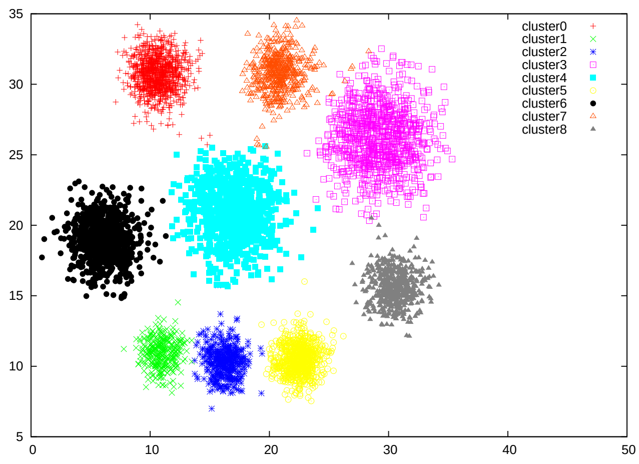


Fig. 3. Generated distributed clustering.

to find the class distribution of the Iris dataset, leading to 3 classes based on 5 or 7 local subclusters. However, because the k-harmonicmeans does not impose a variance constraint it could find a lower sum-of-squared-error which is the case here. These two examples show the independence from the nature and size of the initial clustering. Actually, if there is a real structure in the dataset then true clusters are found and joined together.

For the census dataset, the algorithm leads to 3 clusters based on 20 subclusters locally on 7 processes, and using all the attributes. The local clustering uses the k-means algorithm. This version is based on multiple k-means (user defined parameter) and keep the best output. Firstly, the Figure 6 shows the rank of the values of 9 attributes among the 68 for the whole dataset. The values distribution of two generated global clusters is given in Figure 7. Note that a global sequential clustering is not possible due to the memory restriction related to the fact that the whole data must fit in main memory. Most of the widely used clustering algorithms are concerned with this scalability issue. Beyond the signification of such clustering especially for this dataset and using the entire set of the categorical variables, this experiment shows the scalability of the proposed algorithm. Indeed, specific measurements on the dataset will take into account a specific set of variables. However, the Figure 7 shows, by the selected sorted attributes, different characteristics of these two global clusters concerning age or income for example. Still, the visualization mode does not allow to show the real measurement related to these attributes since they are sorted, which means that there is no true initial observations thereon.

In contrast to many other distributed algorithms, the presented one uses a simple global constraint, a very limited communication overhead, and does not need to know the data structure a priori. This algorithm is effective in finding proper clustering. However, future versions will take into account some other facts as considering the perturbation process during the merging operations and inside subclusters, or of whether or not multi-attributed clusters are present to consider a different approach at this level. Also, varying the constraint criterion could be considered as well as adding other similarity functions.

In fact, the merging process could perform a distance between the distributed subclusters. Each one could be described by additional objects and metrics, as the covariance matrices for example. A general definition of such a distance measure can be $d(x_i, x_j) = (c_j - c_i)^T \; A \; (c_j - c_i)$, where the inclusion of A results in weighting according to statistical properties of the features. Other distances or similarity measures include; Euclidean, Manhattan, Canberra, Cosine, etc. The general form of some of these distances is $d_{i,j} = [\ \sum_K |x_{ki} - x_{kj}|^N\ ]^{\frac{1}{N}}$, and depending on N, the enclosed region takes different shapes. That is to say that the merging process could take into account one or different proximity (i.e similarity or dissimilarity) function to improve the quality of the resulting clustering. This will be considered in future versions. However, the key issue is the selection of an appropriate function, especially, *which kind of measures for which kind of data?* Actually, general observations recommend some distances for some type of data, Euclidean-based for example for dense, continuous data. Still, no true rules exist and the user needs to be familiar and expertise his data.

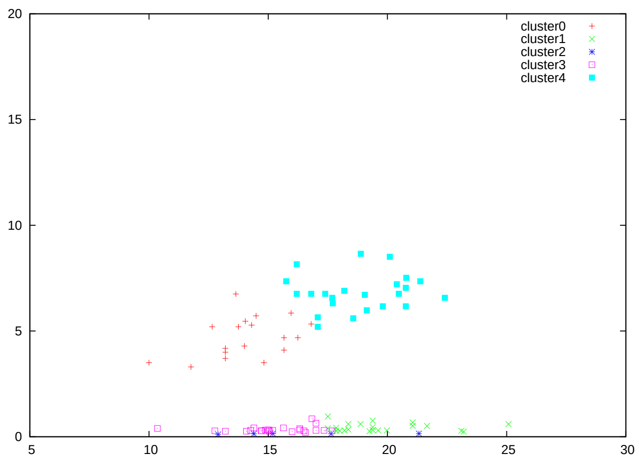
5 Conclusion

In this paper, we evoked the need of efficient distributed and grid-based clustering algorithms. Actually, a huge effort has been made in sequential clustering but there is only few algorithms which tackle the distribution problem especially in loosely coupled environments such as the grid. We proposed a lightweight distributed algorithm based on an increasing variance constraint. It clusters the data locally and independently from each other and only limited statistics about the local clustering are transmitted to the aggregation process which carries out the global clustering, defined as labeling between subclusters. This is done by means of a merging and a perturbation processes. The global model can then be broadcasted to all participating processes if needed, which will use it to label their subclusters.

The algorithm gives good performances at identifying well separated clusters and the real structure of the dataset. In fact, when data are not well separated, the notion of cluster is very confused and does not even exist in the literature. The number of clusters is also automatically found, this resolves the problem of estimating the number of clusters a priori. Furthermore, in addition to classical constraints in distributed clustering, related to the usually infeasible data centralization due to technical, security reasons or local policies, this algorithm can also tackle large and high dimensional datasets that cannot fit in memory since most of the clustering algorithms in literature require the whole data in the main memory and also tend to scale poorly as the size and dimension grow. Nevertheless, open issues could be considered as in the merging process or the choice of the possible better local models and algorithms, in addition to those described in the previous section.

References

1. R. B. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communication in statistics*, 3, 1974.



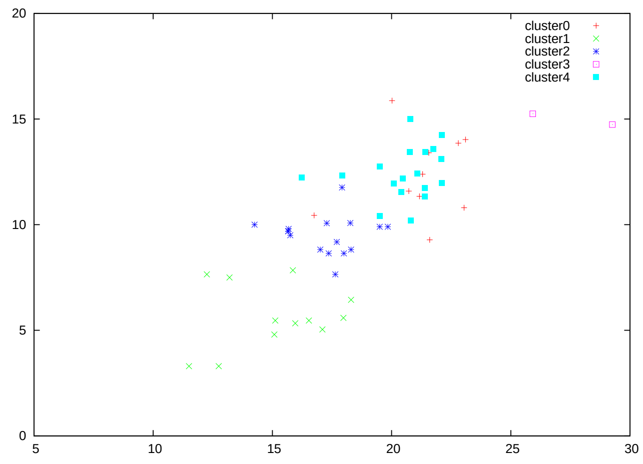
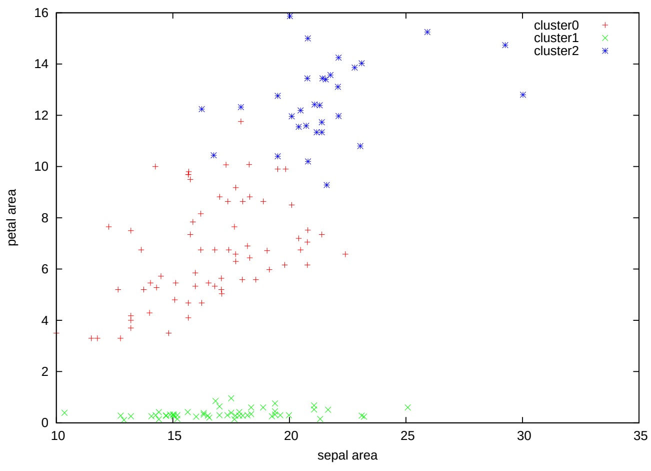
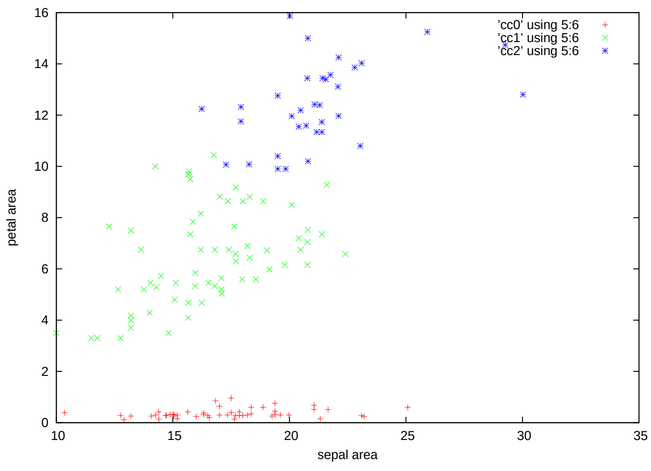


Fig. 4. Iris sub-sets and local clustering using k-harmonicmeans, $k_i = 5, \;\; i = 0, 1$.

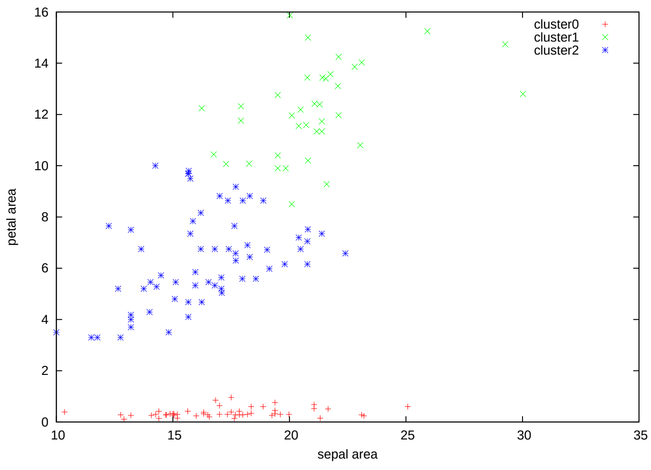
2. M. Cannataro, A. Congiusta, A. Pugliese, D. Talia, and P. Trunfio. Distributed Data Mining on Grids: Services, Tools, and Applications. *IEEE Transaction on System, Man, and Cybernetics*, 34(6), Dec 2004.
3. I. S. Dhillon and D. Modha. A Data-Clustering Algorithm on Distributed Memory Multiprocessors. In *Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*, 1999.
4. M. Ester, H.-P Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 1996.
5. A. Garg, A. Mangla, V. Bhatnagar, and N. Gupta. PBIRCH : A Scalable Parallel Clustering algorithm for Incremental Data. In *10th International Database Engineering and Applications Symposium, IDEAS'06*, 2006.



(a)



(b)



(c)

Fig. 5. The output using 5 `(a)` and 7 `(b)` subclusters, and a global clustering using k-harmonicmeans in `(c)`.

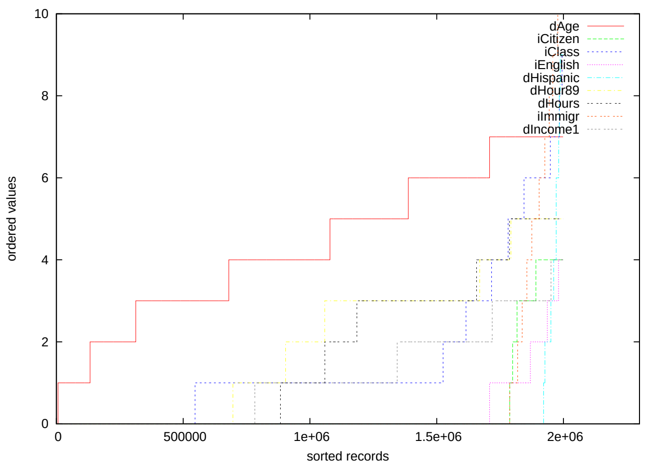
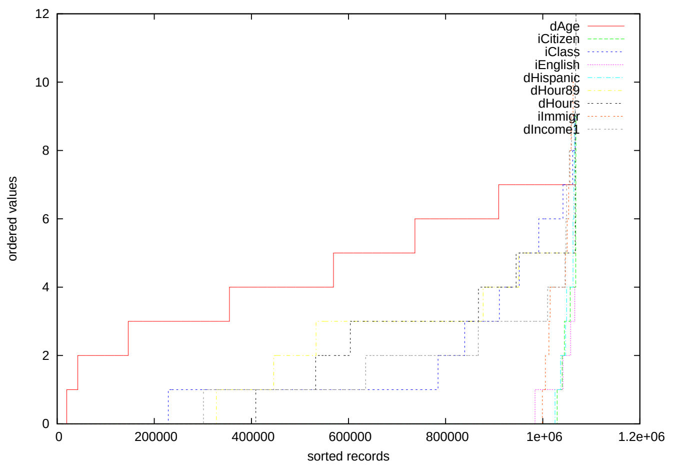


Fig. 6. Rank of the values of 9 attributes of the census database.

6. H. Geng, X. Deng, and H. Ali. A New Clustering Algorithm Using Message Passing and its Applications in Analyzing Microarray Data. In *ICMLA '05: Proceedings of the Fourth International Conference on Machine Learning and Applications (ICMLA'05)*, pages 145–150. IEEE Computer Society, 2005.
7. V. M. Ghanem, Y. M. Kohler, A. J. Sayed, and P. Wendel. Discovery Net: Towards a Grid of Knowledge Discovery. In *Eight Int. Conf. on Knowledge Discovery and Data Mining*, 2002.
8. A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, Sep 1999.
9. E. Januzaj, H-P. Kriegel, and M. Pfeifle. Towards Effective and Efficient Distributed Clustering. In *Int. Workshop on Clustering Large Data Sets, 3rd Int. Conf. on Data Mining, ICDM*, 2003.
10. E. Januzaj, H-P. Kriegel, and M. Pfeifle. DBDC: Density-Based Distributed Clustering. In *9th Int. Conf. on Extending Database Technology, EDBT*, 2004.
11. E. Januzaj, H-P. Kriegel, and M. Pfeifle. Scalable Density-Based Distributed Clustering. In *8th European Conference on Principles and Practice Discovery in Databases PKDD.*, 2004.
12. R. Jin, A. Goswani, and G. Agrawal. Fast and Exact Out-of-Core and Distributed K-Means Clustering. *Knowledge and Information Systems*, 10, July 2006.
13. M. N. Joshi. Parallel K-Means Algorithm on Distributed Memory Multiprocessors. Technical report, University of Minnesota, 2003.
14. G. Kickinger, J. Hofer, P. Brezany, and A. M. Tjoa. Grid Knowledge Discovery Processes and an Architecture for their Composition. *Parallel and Distributed Computing and Networks*, 2004.
15. N-A. Le-Khac, M. T. Kechadi, and J. Carthy. ADMIRE framework: Distributed Data Mining on Data Grid platforms. In *first Int. Conf. on Software and Data Technologies, ICSOFT*, 2006.
16. R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *VLDB, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, 1994.



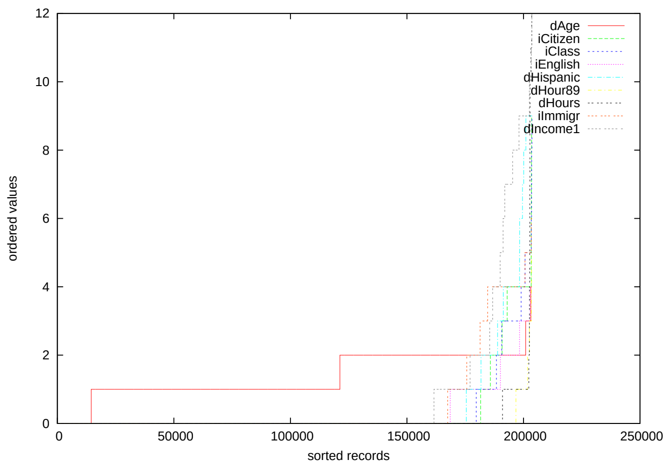


Fig. 7. Values distribution of two generated global clusters.

17. 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.
18. C. J. Veenman, M. J. Reinders, and E. Backer. A Maximum Variance Cluster Algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 24(9), Sep 2002.
19. R. Xu and D. Wunsch. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16, May 2005.
20. X. Xu, J. Jager, and H.-P. Kriegel. A Fast Parallel Clustering Algorithm for Large Spatial Databases. *Journal of Data Mining and Knowledge Discovery*, 3, 1999.
21. B. Zhang and G. Forman. Distributed Data Clustering Can be Efficient and Exact. Technical report, HP Labs, 2000.
22. B. Zhang, M. Hsu, and U. Dayal. K-Harmonic Means - A Data Clustering Algorithm. Technical report, HP Labs, 1999.