D-GridMST: Clustering Large Distributed Spatial Databases

Ji Zhang
Department of Computer Science
University of Toronto
Toronto, Ontario, M5S 3G4, Canada
Email: jzhang@cs.toronto.edu

Abstract:
In this paper, we will propose a distributable clustering algorithm, called Distributed-GridMST (D-GridMST), which deals with large distributed spatial databases. D-GridMST employs the notions of multi-dimensional cube to partition the data space involved and uses density criteria to extract representative points from spatial databases, based on which a global MST of representatives is constructed. Such a MST is partitioned according to users' clustering specification and used to label data points in the respective distributed spatial database thereafter. Since only the compact information of the distributed spatial databases is transferred via network, D-GridMST is characterized by small network transferring overhead. Experimental results show that D-GridMST is effective since it is able to produce exactly the same clustering result as that produced in centralized paradigm, making D-GridMST a promising tool for clustering large distributed spatial databases.

1. Introduction

With rapid development of techniques in data acquisition and storage, spatial databases store an increasing amount of space-related data such as satellite maps, remotely sensed images and medical images. These data, if analyzed, can reveal useful patterns and knowledge to human users. Clustering is a process whereby a set of objects is divided into several clusters in which each of the members in some way similar and is different from the members of other clusters [11]. Spatial data clustering, aiming to identify clusters, or densely populated regions in a large and multi-dimensional spatial dataset, serves as an important task of spatial data mining. Though a large number of spatial clustering algorithms have been proposed in literature so far, most of them assume the data to be clustered are locally resident in centralized scenario, making them unable to cluster inherently distributed spatial data sources.

[2] presents a distributed data clustering algorithm that is designed based on a classical clustering algorithm PAM and a spanning tree clustering algorithm, called Clusterize. [3] proposes an approach to deal with clustering data emanating from different sites. It operates in three major steps: (1) find the local clusters of data in

each site; (2) find (high) clusters from the union of the distribution data sets at the central site; (3) finally compute the associations between the two sets of clusters. In [6], three classical clustering methods, namely *K*-means, *K*-Harmonic Means (KHM) and Maximum Expectation (EM) are modified to parallelise the clustering of distributed data sources. More recently, a parallel implementation of *K*-means based on the message-passing model is presented [5]. To deal with heterogeneity of data across distributed sites, [9] presents a Collective Hierarchical Clustering (CHC) algorithm for analyzing distributed and heterogeneous data. This method first generates local cluster models and then combines them to generate the global cluster model of the data.

In this paper, we will propose a distributable clustering algorithm, called *Distributed-GridMST (D-GridMST)*, that deals with large distributed spatial databases. D-GridMST employs the notions of multi-dimensional cube and grid to partition the data space involved and uses density criteria to extract representative points from spatial databases, based on which a global MST of representatives is constructed. Such a MST is partitioned according to users' clustering specification and used to label data points in the respective distributed spatial database thereafter. Since only the compact information of the distributed spatial databases is transferred via network, D-GridMST is characterized by small network transferring overhead. Experimental results show that both the centralized (GridMST) and distributed (D-GridMST) versions of our clustering technique are efficient and effective since it is able to produce exactly the same clustering result as that produced in centralized paradigm. These advantages are believed to make D-GridMST a promising tool for clustering large distributed spatial databases.

The remainder of this paper is organized as follows. In Section 2, we will present GridMST, our clustering technique for clustering centralized spatial databases. Section 3 gives the details of D-GridMST. Experimental results are reported in Section 4. The final section concludes this paper.

2. GridMST

GridMST is a new approach that aims to address two specific needs in the clustering of dynamic spatial databases, namely multi-resolution clustering and incremental clustering. GridMST is fast, scalable, robust to noise, and effective in accomplishing multi-resolution and incremental clustering. Figure 1 gives an overview of GridMST. It consists of three major parts. The first part deals with scaling the algorithm for very large spatial databases. The second part deals with extracting the necessary information to build a summary structure for multi-resolution clustering and incremental clustering. The final part deals how the multi-resolution clustering and/or incremental clustering make use of the summary structures to perform analysis, respectively.

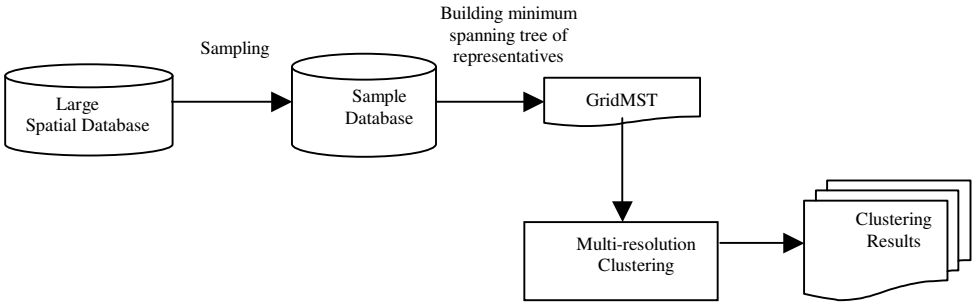


Figure 1. Overview of GridMST

2.1 Sampling Large Spatial Databases

Similar to BIRCH, CURE, and C2P, GridMST handles with the problem of very large spatial databases by sampling the databases. [7] derives a theorem to determine the minimum sample size required to ensure that a fraction of the cluster is always included in the sample with probability δ. That is, for a cluster *u*, if the sample size *s* satisfies

$$s \geq fN + \frac{N}{|u|}\log(\frac{1}{\delta}) + \frac{N}{|u|}\sqrt{(\log(\frac{1}{\delta}))^2 + 2f|u|\log(\frac{1}{\delta})}$$

then the probability that the sample contains fewer than $f|u|$ points belonging to cluster u is less than δ, where *N* is the size of the dataset, $|u|$ is the size of the cluster *u,* $0 \leq f \leq 1, 0 \leq \delta \leq 1$.

GridMST uses this theorem to determine the sample size and performs uniform sampling on the large spatial database to obtain a sample database. We observe that this sample database could still be too large to fit entirely into the main memory. In this case, GridMST will divide the sample database into several smaller partitions, each of which can be loaded into the main memory. The partitions are read in one at a time and processed for their density information. When all the partitions have been scanned, the grid cells occupied by the whole sampling dataset are obtained (the grid structure will be discussed later in this paper). The size of occupied grid cell is small enough to be stored in the main memory. Representative points can now be generated based on the density information of these grid cells. Note that these representative points are for the entire sample database. This makes GridMST flexible and yet effective in handling samples of all sizes.

3.2 Constructing the R-MST

In GridMST, a number of representative points of the database are picked using the density criterion. A minimum spanning tree of these representative points, denoted as R-MST, is built. GridMST constructs R-MST in a number of steps. First, a grid data structure is constructed whereby each point in the dataset is assigned to one and only

one cell in the grid. The density of each grid cell is then computed. If the density of a grid cell exceeds some user-specified threshold, then the cell is considered to be dense and the centroid of the points in the dense cell is selected as the representative point of this cell. Once the representative points have been selected, a graph-theoretic based algorithm is used to build the R-MST. We now introduce some basic definitions that are used in the algorithm to construct the R-MST.

3.2.1 Definitions

Definition 1: Relative Density of a grid cell
Let *g* be some cell in a grid structure G. Let *n* be the number of points in *g* and *avg* be the average number of points in a cell in G. Then, the Relative Density of *g*, denoted as RD(*g*), is defined as the ratio *n / avg*.

A grid cell is a neighbor of some grid cell *g* if it is directly connected to *g*. Hence, a center grid cell will have 8 neighboring grid cells, an edge grid cell will have 5 neighboring grid cells, and a corner grid cell will have only 3 neighboring grid cells.

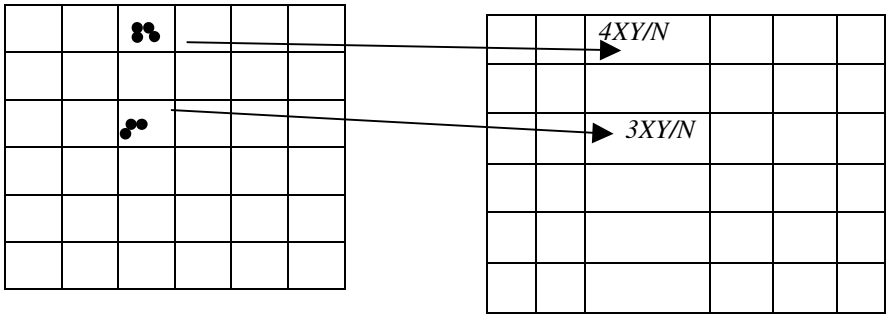


Figure 2. Relative density of grid cells

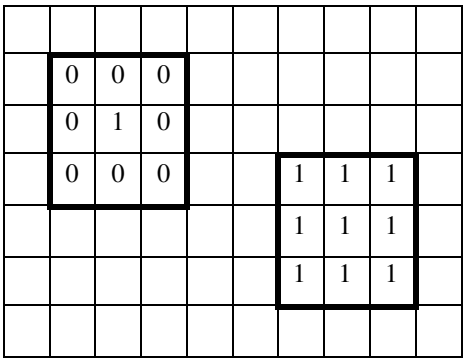


Figure 3. Two grid cells that have the same RD but different ND

Definition 2: Neighborhood Density of a grid cell
Let *g* be some cell in a grid structure G and *Neighbor* be the set of neighboring grid cells of *g*. The Neighborhood Density of *g*, denoted as ND(*g*), is defined as the average of the densities of *g* and its neighboring grid cells. ND(*g*) is given by the following formula:

$$ND(g) = \frac{1}{t} * (RD(g) + \sum_{g_i \in Neighborg_g} RD(g_i))$$

where t =1+ the number of neighboring grid cells of *g*. Specifically,

$$t = \begin{cases} 6 & \text{if } g \text{ is an edge grid cell} \\ 4 & \text{if } g \text{ is a corner grid cell} \\ 9 & \text{otherwise} \end{cases}$$

Definition 3: Dense vs. Non-dense grid cells
Let *g* be some cell in a grid structure G. *g* is a dense grid cell if ND(*g*) is greater than or equal to some user specified density threshold, *Td*, otherwise *g* is a non-dense grid cell.

Suppose *X* and *Y* are the horizontal and vertical number of grid cells in grid structure and *N* is the size of the dataset. Then the average number of points in the cells of the grid is given by *N/XY*. Figure 2 shows the densities of the non-empty grid cells in the grid structure. Note that from Definition 3, a grid cell is considered dense if its neighborhood density, ND, exceeds some user-specified threshold. The reason for using ND rather than RD to determine the denseness of a grid cell is that ND measures not only its own density, but it also reflects the density of its neighboring area. This actually compensates for the effect of outliers.

Figure 3 highlights two grids. Although the center grid cells of these two grids have the same RD of *XY/N*, these cells have different ND values. The ND of the left center grid (which is most likely an outlier) is *XY*/5*N*, while the ND of the right center grid is *XY/N*. This example clearly shows that ND of a grid cell is more effective in limiting the effect of an outlier than RD. After the dense grid cells have been identified, we compute the centroid of data points falling into each dense cell. These centroids will form the set of representative points that well reflects the approximate distribution of the data points in the entire dataset.

Definition 4: R-MST
Suppose *A* = <*V, E*> is an undirected graph, where *V* is the set of vertices denoting the set of representative points and *E* is the set of edges connecting these representative points. R-MST is a connected acyclic subgraph of *A* that has the smallest total cost (or length), which is measured as the sum of the costs of its edges.

3.2.2 Generating Representative Points

To generate the representative points, all the grid cells are examined in order to extract the dense cells. We devise an efficient algorithm to extract the dense cells and generate representative points. The steps of the algorithm are shown in Figure 4. Steps 1-6 perform the grid cell mapping, i.e. assign each data point into one and only one grid cell. Once a point has been assigned, the density of its corresponding cell is incremented by 1. Steps 7-13 extract dense cells based on some pre-specified density threshold, *Td*. A cell is a dense grid cell if its Neighborhood Density is greater than or equal to *Td*, otherwise it is a non-dense grid cell. The representative points are generated using the centroids of all the points in the dense cells and are added to the list of representative points.

```
Procedure Repr_Generation (Dataset D)
Begin
1. For each point p in D Do
2.{
3.    Cell (p)=Map(p);
4.          j = Hash (Cell(p));
5.    Count [j]++;
6. }
7. For each cell i in the hash table do
8. {
9.    ND=Neighborhood_Den(i);
10.   If (ND >=Td) Then {
11.         Cell i is a dense cell;
12              Repr_List=Repr_List ∪ CentroidOfPoints(i);
13.   } }
End
```

Figure 4. Algorithm to extract representative points.

3.2.3 Constructing the R-MST

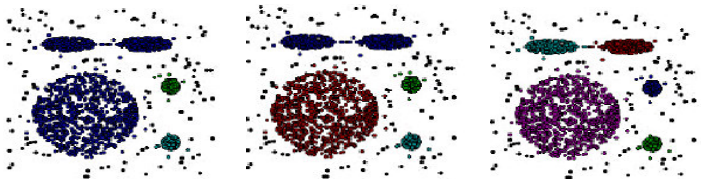
The algorithm for constructing the R-MST is as follows:
(1) Compute all the pair-wise Euclidean distances of representatives and sort them in ascending order. This forms a pool of potential R-MST edges;
(2) Pick edges from the pool of potential edges obtained in (1), starting with the shortest edge. Only those edges that do not create a cycle with the edges that have already been chosen are picked. The process terminates when the number of such edges is equal to (*Nr* - 1), where *Nr* is the number of representative points.

2. 3 Using R-MST for Spatial Clustering

After the R-MST has been constructed, multi-resolution clustering can be easily achieved. Suppose a user wants to find *k* clusters. A graph search through the R-MST is initiated, starting from the largest cost edge, to the lowest cost edge. As an edge is traversed, it is marked as deleted from the R-MST. The number of partitions resulting from the deletion is computed. The process stops when the number of partitions reaches *k*. Any change in the value of *k* simply implies re-initiating the search-and-marked procedure on the R-MST. Once the R-MST has been divided into

k partitions, we can now propagate this information to the original dataset so that each point in the dataset is assigned to one and only one partition/cluster. A naive approach is to go through the dataset once, and compute the distance between each point to all the representative points. The data point is then assigned to the cluster whose representative point it is closest to. However, we observed that if a data point falls into a dense cell, say *dc*, then the nearest representative point is the representative point of *dc*. Thus, we can immediately assign the data point the same cluster label as the representative point of cell *dc*. For those data points that fall into non-dense cells, we use an indexing structure for high dimensional data, the X-tree [1], to speedup the search for the closest representative point. Once the closest representative point is found, the corresponding data point will be assigned the same cluster label as this its closest representative point.

Figure 5 illustrates the multi-resolution clustering in *GridMST*. *GridMST* operates in two modes for multi-resolution clustering: a manual mode and an automatic mode. In the manual mode, the user will specify the value of *k*. In the automatic mode, the system automatically searches for the optimal number of clusters based on some pre-defined inter-cluster distance threshold.



(a) k=3 *(b) k=4* *(c)k=5*

Figure 5. Multi-resolution clustering

3. D-GridMST (Distributed GridMST)

After discussing the technique of GridMST that is mainly applicable in clustering spatial database in the centralized paradigm, we, in this section, will present D-GridMST, the distributed version of GridMST that works with distributed spatial databases. The adaptation from GridMST to D-GridMST mainly involves (i) Generation of global data model by combining local data models in centralized site, and (ii) Local data clustering analysis in each of the distributed sites.

In order to produce the clustering result of these distributed databases that is comparable to result of a centralized database in D-GridMST, globalization of local data model is entailed to obtain global data model that captures the cluster features for the whole dataset. The local data model we use in D-GridMST is simple and small in size, which contributes to the small network transfer through the network. Specifically, the globalization of local data model in D-GridMST involves:

(1) Globalize range of every dimension of data in each distributed site

Global range of every dimension of the data is required to construct a structure of global multi-dimensional cube. Here, we assume the all the spatial data reside in the distributed databases are homogenous in nature, which ensures that such combination can be performed meaningfully. The object of obtaining global range of every dimension is to ensure that the grid constructed is able to encapsulate all the data points stored in the distributed sites. To this end, all distributed sites are required to provide the central site with the information regarding the maximum and minimum values, i.e. the range, of every dimension of local data points. Upon the receipt of such information from all distributed sites, the central site will commence to produce the global range information. Specifically, in a d dimensional dataset, let $L_{max}(i)$ and $L_{min}(i)$ be the local maximum and minimum values of dimension i, the global maximum and minimum values of dimension i, denoted as $G_{max}(i)$ and $G_{min}(i)$, are produced as follows:

$$G_{min}(i)=min(L_{min}(1), L_{min}(2), \ldots L_{min}(d))$$
$$G_{max}(i)=max(L_{max}(1), L_{max}(2), \ldots L_{max}(d))$$

The range of i^{th} dimension, denoted as $R(i)$ can be computed as

$$R(i)= G_{max}(i)-G_{min}(i) \quad (1 \leq i \leq d)$$

(2) Globalize local occupied cells in each distributed site
Here**,** the occupied cells refer to the cells that occupied by the data points in the database. In other words, the occupied cells are those whose density is at least 1. Local occupied cells are those cells occupied by the local data points in the distributed site, and global occupied cells are those cells occupied by all the data points. The global occupied cells serve as the potential pool for the selection of dense cells: the dense cells are only the occupied cells whose neighborhood density exceeds some threshold.

The global occupied cells are the union of local occupied cells. Suppose there are S distributed sites and LOC(i) denotes the local occupied cells of the i^{th} distributed site. The global occupied cells, denoted by GOC, can be generated as follows:

$$GOC=LOC(1) \cup LOC(2) \ldots \cup LOC(S)$$

3. 1 D-GridMST Algorithm

In this section, we will give D-GridMST the algorithm that performs clustering of distributed spatial databases. The algorithm of D-GridMST is presented in Table 1.

3. 2 The Complexity of D-GridMST

The complexity analysis in the subsection includes the analysis of its computational complexity, space complexity and transfer overhead. The notation that will be used in the analysis are first presented below:

Step	Transfer/ Location	Operation
1	DS⟶CS	Transfer local range of every dimension of data
2	CS	Globalize local range to global range
3	CS⟶DS	Transfer global range and create global multi-dimensional cube *C*
4	DS	Assign local points into *C* and compute the density
5	DS ⟶CS	Transfer local occupied cells and their densities
6	CS	Globalize local occupied cells of the cube
7	CS	Generate representative points and construct MST
8	CS	Perform multi-resolution clustering using MST
9	CS⟶DS	Transfer clustering result of representative points
10	DS	Label local data points

Table 1. Algorithm of D-GridMST

Table 2 gives the annotation of the notations used in the ***Transfer/location*** field in Table 1.

Value	Meaning
CS	Clustering operations in centralized site
DS	Clustering operations in all the distributed sites
CS to DS	Data are transferred from centralized site to all the distributed sites
DS to CS	Data are transferred from all distributed sites to the centralized site

*Table 2. Annotations of the **Transfer/location** field in Table 1.*

Let N be the total size of spatial databases in all distributed sites
d be the number of dimension of each spatial database
S be the total number of the distributed sites
N_c be the number of cells in the multi-dimensional cube
N_r be the number of the global representative points

(a) Computational complexity:

The computational for D-GridMST involves Step 2, 4, 6, 7, 8 and 10 of the algorithm. Globalizing the local range to the global one in Step 2 requires a complexity of O(d). Assigning local points into the multi-dimensional cube C and compute the density in Step 4 requires a complexity of O(N), where $N=N_1+N_2+\ldots+N_d$. In Step 6, Globalizing local occupied cells of the cube involves the union of occupied cells in the cube of all distributed sites, whose complexity is at most O($d*N_c$). Generation of representative points and construction of MST requires O($N_c+N_r^2$). Using MST to cluster the representatives only requires O(N_r). Finally, labeling all points has a complexity of O(N). In sum, the computational complexity is O($d+N+d*N_c+N_c+N_r^2+ N_r+N$). Given $d<<N$, $N_c<<N$ and $N_r<<N$, thus the computational complexity of D-GridMST is O(N).

(b) Space Complexity

The storage requirements for the centralized and distributed sites are different in D-GridMST: the centralized site has to store the range of each dimension of all distributed sites (O(d*S)), the occupied cells of all distributed sites (O(N_c*S)) and the MST generated (O(N_r)). Because d<<N_c and N_r<<N_c for most spatial datasets, thus the space complexity of D-GridMST is O(N_c*S). As for each distributed site, it will have to store the global range of data across all the sites and clustering result of representative points in addition to the original data in this site, therefore the storage requirement for distributed site S_i is O(N_i+$Max(d$*$S, N_r)$) ($1 \leq i \leq S$).

(c) Transferring Overhead

The data transferring between the centralized and distributed sites occurs in Step 1,3, 5,and 9, respectively. The overhead of transferring the local range of each dimension of distributed data and the global data range between the centralized and distributed sites is O(2*d*S). Transferring local occupied cells and their densities from distributed sites to centralized site has a overhead of O(2*S*N_c) at most. Finally, the overhead of transferring the clustering result of representative points from centralized site to distributed sites is O(N_r). Therefore, the total transferring overhead is O(2*d*S +2*S*N_c+ N_r).

Remarks:
From the above analysis, we can see that D-GridMST is promising in the sense that:
(1) It is efficient because that the computational time is linear with respect to the total size of the distributed data;
(2) It is space economic because D-GridMST only imposes small space requirements on both centralized and distributed sites;
(3) It has small transferring overhead since the overhead of O(2*d*S +2*S*N_c+ N_r) is definitely much smaller than that of the strategy that all the data is first centralized before clustering is performed.

4. Experimental Results

We divide the experimental results in this section into two major parts. We will first evaluate the effectiveness and efficiency of GridMST, followed by study of effectiveness and efficiency of D-GridMST.

In the first part, we compare the performance of GridMST against three related clustering techniques, i.e. DBSCAN[4], C2P[10] and DENCLUE[8]. DBSCAN is a widely used density-based clustering algorithm. C2P is the latest agglomerative hierarchical clustering algorithm using the notion of representative points of dataset. [10] has shown experimentally that C2P outperforms BIRCH and CURE in terms of efficiency in multi-resolution clustering with comparable clustering results as CURE.

DENCLUE, which models density of the data points using a kernel method, provides multi-resolution view of data clusters.

In the second part, we will study the effect of the size of spatial databases in each distribute site and the number of distributed sites on the efficiency of D-GridMST. Also, we will propose a metric for measuring the similarity of between the clustering results of using the same clustering algorithm in clustering the centralized and distributed datasets.

4.1 Effectiveness of Clustering Centralized Databases Using GridMST

Figures 6 to 8 show the results of clustering Datasets 1-3 (different colors are used to denote the different clusters). We see that GridMST works well for all the datasets. DBSCAN and DENCLUE are unable to cluster Dataset 1 and Dataset 3 correctly. C2P generates the correct clustering results for all datasets except Dataset 3. On closer examination, we note that C2P does not work well on datasets that are concave. According to merge method of C2P, when two clusters are merged, a subset of points that are closest to the cluster center is selected and used as the representative points of the new cluster. The choice of these "closest" points reduces the effectiveness of these points being able to reflect the true shape and size of the cluster. From this experiment, we can clearly see than GridMST outperforms other methods in clustering concave clusters and dealing with effects resulting from outliers.

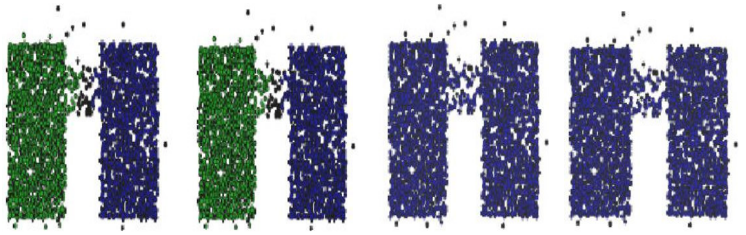


Figure 6. Clustering result for Dataset 1

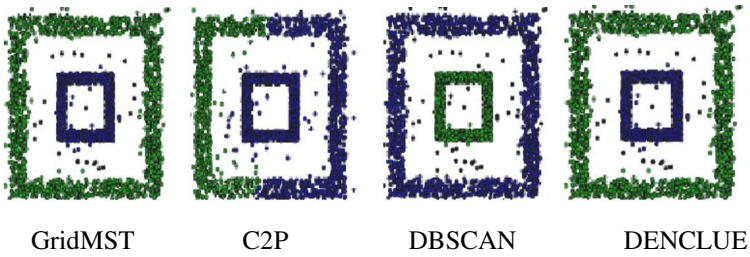


Figure 7. Clustering result for Dataset 2

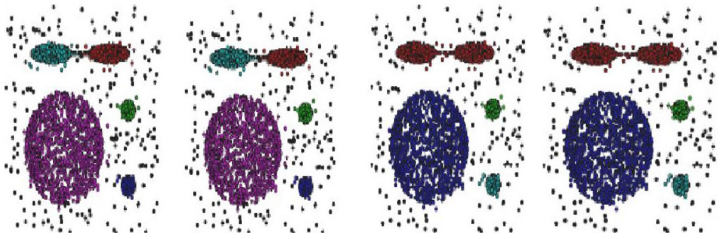


Figure 8. Clustering result for Dataset 3

4.2 Efficiency of Clustering Centralized Databases Using GridMST

In this experiment, we evaluate the execution time of GridMST, DBSCAN, C2P and DENCLUE on different sizes of datasets. The sizes of these datasets range from 100,000 to 1,000,000 points. Figure 9 shows that GridMST is scalable to large datasets and outperforms other three methods. DBSCAN carries out clustering on the entire dataset. Therefore, the execution time goes up rapidly when the dataset size increases. In C2P, a number of sub-clusters are obtained first and clustering is performed on these sub-clusters followed by a labeling process. This strategy helps somewhat but its time complexity of $O(N\log N+M^2\log M)$ where M is the number of sub-clusters, is still much higher than GridMST whose time complexity is approximately linear with respect to the dataset size. Since the complexity analysis of DENCLUE reveals that it has a linear complexity with respect to the size of dataset, the time of DENCLUE is the closest to that of GridMST among all the methods yet still slightly high than that of GridMST. GridMST achieves speedup by extracting the representative points of the dataset and clustering these representative points before labeling the whole original dataset. The clustering of representatives turns out to be very efficient since the number of these representative points is normally far less than the number of points in the whole dataset.

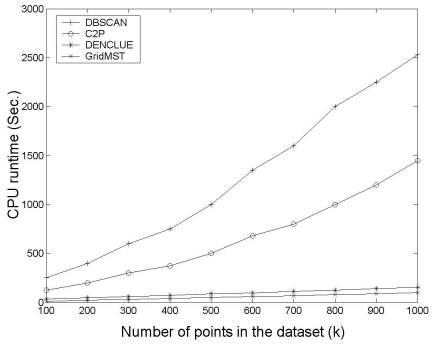


Figure 9. Execution time of clustering centralized datasets on varying dataset size

4.3 Efficiency of Clustering Distributed Databases Using D-GridMST

Apart evaluating the efficiency of GridMST on Centralized databases, we will also study the efficiency of D-GridMST on distributed databases. We will study the effect of the size of spatial databases in each distribute site and the number of distributed sites on the efficiency of D-GridMST. For the sake of simplicity, we set of the spatial databases in each distributed sites in this experiment as of each equal size. The size of the database in each distributed site ranges from 100,000 to 500,000 and the number of distributed sites varies from 10 to 30 in the experiments. The results are shown in Figures 10 and 11. The two basic findings of the experiments are: (i) The execution time of clustering is approximately linear with respect to the size of databases residing in the distributed sites. This is because the assignment of each point in the database into the multi-dimensional cube and the labeling of all the points in the database will dominate the execution time of D-GridMST, thus the complexity of D-GridMST is linear with respect to the total size of databases it works on; (ii) The execution time is nearly not affected by the factor of *S*. This is because local clustering in each distributed site can inherently be paralleled.

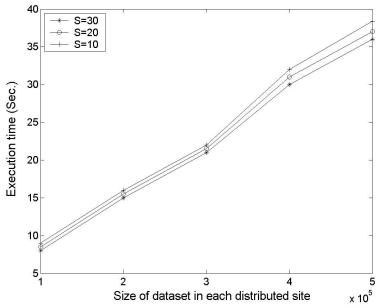


Figure 10. Execution time of D-GridMST on varying dataset size in each distributed site

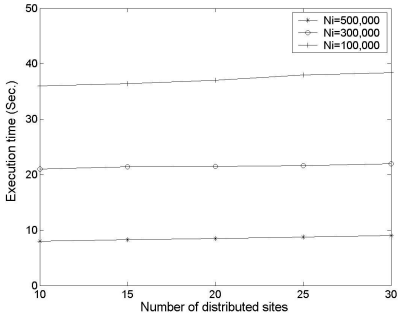


Figure 11. Execution time of D-GridMST on varying number of the distributed sites

4.4 Effectiveness of Clustering Distributed Databases Using D-GridMST

As far as effectiveness of a distributed clustering algorithm is concerned, we are interested whether the clustering result is consistent with that of the centralized version of the algorithm. To measure this, we devise a metric that, to some extent, reflects the closeness of the two clustering results of centralized and distributed versions of the algorithm. This metric, termed as Clustering Similarity (*CluSim* for short), is defined as follows:

$$CluSim = \frac{1}{N}\sum_{i=1}^{N}(label_{cen}(p_i) = label_{dis}(p_i)) * 100\%$$

where $label_{cen}(p_i)$ and $label_{cen}(p_i)$ denote the cluster label of the point p_i using the centralized and distributed versions of clustering algorithm, respectively. $label_{cen}(p_i) = label_{cen}(p_i)$ returns 1 when point p_i is assigned the same cluster label under the two algorithms and returns 0 otherwise. Given that the cluster labels of two algorithms might not be consistent (the same cluster may have different cluster labels in the two clustering results). Therefore, we have to first make the cluster labeling of the two algorithms consistent before *CluSim* can be computed. This can be achieved by finding the best-matched pairs of centriods of clusters in the two results by computing their distance. The points in the clusters whose centriods are a best-matched pair have the same clustering labels.

Using above formula, we compute *CluSim* of D-GridMST and two exiting distributed clustering algorithms *K*-means and EM. The result are D-GridMST(100%), *K*-means (93%) and EM(90%). Clearly, D-GridMST outperforms the other two algorithms and is able to produce exactly the same clustering result as that produced by working on the centralized data from all distributed sites.

5. Conclusion

In this paper, we will propose a distributable clustering algorithm, called Distributed-GridMST (D-GridMST), which deals with large distributed spatial databases. D-GridMST employs the notions of multi-dimensional cube to partition the data space involved and uses density criteria to extract representative points from spatial databases, based on which a global MST of representatives is constructed. Such a MST is partitioned according to users' clustering specification and used to label data points in the respective distributed spatial database thereafter. Since only the compact information of the distributed spatial databases is transferred via network, D-GridMST is characterized by small network transferring overhead. Experimental results show that D-GridMST is effective since it is able to produce exactly the same clustering result as that produced in centralized paradigm.

References

[1] S. Berchtold, D. A. Keim and H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. *Proc. 22nd International Conference on Very Large Data Base (VLDB'96),* Mumbai, India, 1996.

[2] D. K. Bhattacharyya and A. Das. A New Distributed Algorithm for Large Data Clustering. *IDEAL'2000*, pp.29-34, 2000.

[3] A. Bouchachia. Distributed Data Clustering. *CAiSE*, 2003.

[4] M. Charikar, C. Chekuri, T.Feder, and R. Motwani. Incremental Clustering and Dynamic Information Retrieval. *ACM Symposium on Theory of Computing*, 1997.

[5] I. S. Dhillon and D. S. Modha. A Data-clustering Algorithm on Distributed Memory Multiprocessors. *Large-Scale Parallel Data Mining*, pp 245-260, 2002.

[6] G. Forman and B. Zhang. Distributed Data Clustering Can be Efficient and Exact. *SIGKDD Explorations*, Vol.2 Issue 2, pp 34-38, 2000.

[7] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering Large Datasets in Arbitrary Metric Spaces. *Proc. 15th International Conference on Data Engineering (ICDE'99),* Sydney, Australia, 1999.

[8] J. He, A. H. Tan, C. L. Tan, and S.Y. Sung. On Quantitative Evaluation of Clustering Systems. *Information Retrieval and Clustering,* Kluwer Academic Publishers, 2002.

[9] E. L. Johnson and H. Kargupta. Collective Clustering from Distributed Heterogeneous Data. *Large-Scale Parallel Data Mining*, pp 221-244, 2000.

[10] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proc. 5th Berkeley Symposium on Math, Statistics and Probability,* 1, pages 281-297, 1967.

[11] M. Zait and H. Messatfa. A Comparative Study of Clustering Methods. *Future Generation Computer Systems*, Vol.13, pp. 149-159, 1997.