International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:9, 2008

A Distributed Algorithm for Intrinsic Cluster Detection over Large Spatial Data

Sauravjyoti Sarmah, Rosy Das, and Dhruba Kr. Bhattacharyya

***Abstract*—**Clustering algorithms help to understand the hidden information present in datasets. A dataset may contain intrinsic and nested clusters, the detection of which is of utmost importance. This paper presents a Distributed Grid-based Density Clustering algorithm capable of identifying arbitrary shaped embedded clusters as well as multi-density clusters over large spatial datasets. For handling massive datasets, we implemented our method using a 'shared-nothing' architecture where multiple computers are interconnected over a network. Experimental results are reported to establish the superiority of the technique in terms of scale-up, speedup as well as cluster quality.

***Keywords*—**Clustering, Density-based, Grid-based, Adaptive Grid.

I. Introduction

IDENTIFICATION and extraction of hidden information and patterns from huge datasets is a challenge in data mining. Clustering is the process of division of a dataset into subsets or clusters, so that the intra-cluster similarity is as high as possible, while data in different clusters are dissimilar [1]. From the aspect of geometry, clustering is the process of identifying dense regions from sparsely populated regions. Clustering is very effective in discovering hidden patterns of datasets and is an important research topic. Major clustering techniques have been classified into partitional, hierarchical, density-based, grid-based and model-based. Among these techniques, the density-based approach is famous for its capability of discovering arbitrary shaped clusters of good quality even in noisy datasets [2].

In this paper, we present an efficient distributed intrinsic cluster detection algorithm, which can handle massive spatial datasets with better clustering quality. The proposed algorithm, that uses a 'shared-nothing' architecture, can be initiated in any of the available nodes (computers). The initiator node of the Formatting Toolbar at the top of your *Word* window starts a partitioning strategy thereby dividing the whole data set into partitions and then distributing the partitions to each of the available computers on the network (one partition is also retained by itself). Every node clusters only its local data. The initiator node manages the task of dynamic load balancing and merges the cluster results produced by each of the nodes. The clustering method exploits a grid based technique to group the data points into blocks and the density of each grid cell is calculated. The blocks are then clustered by a topological search algorithm. For finer clustering result, a triangle-subdivision method is used. The algorithm finds quality clustering even over variable density space. The rest of the paper is organized as follows. Section 2 provides a selected review on density based, grid based and distributed clustering and also it reports the background of the proposed work. Section 3 illustrates the proposed algorithm. In section 4, we present the experimental results and the performance analysis of the work. Lastly, we conclude with a summary in section 5.

Sauravjyoti Sarmah is with the Dept. of CS & Engg., Jorhat Engineering College, Jorhat-785007, Assam, India and is pursuing his Ph.D. from the Dept. of CS & Engg., Tezpur University, Tezpur-784028, Assam, India (corresponding author phone-9707506445; e-mail: sauravjs@gmail.com).

Rosy Das is a research scholar with the Dept. of CS & Engg., Tezpur University, Tezpur-784028, Assam, India (e-mail: rosy8@tezu.ernet.in).

Dhruba Kr. Bhattacharyya is Professor in the Dept. of CS & Engg., Tezpur University, Tezpur-784028, Assam, India (e-mail: dkb@tezu.ernet.in).

II. Related Works

This section reports a selected review on some of the relevant density based as well as grid based clustering techniques.

A. Density Based Approach

The idea behind density based clustering approach is that the density of points within a cluster is higher as compared to those outside of it. DBSCAN [2] is a density-based clustering algorithm capable of discovering clusters of various shapes even in presence of noise. The key idea of DBSCAN is that for each point of a cluster, the neighborhood of a given radius (ε) has to contain at least a minimum number of points and the density in the neighborhood has to exceed some threshold. It is efficient for large spatial databases but, for massive datasets, it becomes very time consuming, even if the use of R* tree is made. Another drawback of DBSCAN is that due to the use of the global density parameters, it fails to detect embedded or nested clusters.

B. Grid Based Approach

Grid based methods divide the data space into a finite number of cells that form a grid structure on which the clustering operations are performed. There is high probability that all data points that fall into the same grid cell belong to the same cluster. Therefore all data points belonging to the same cell can be aggregated and treated as one object [3]. It is due to this nature that grid-based clustering algorithms are computationally efficient which depends on the number of cells in each dimension in the quantized space. It has many advantages such as the total number of the grid cells is

independent of the number of data points and is insensitive of the order of input data points. Some of the popular grid-based clustering techniques are STING [4], WaveCluster [5], CLIQUE [6], pMAFIA [7] etc. STING [4] uses a multi-resolution approach to perform cluster analysis. The advantage of STING is that it is query-independent and easy to parallelize. However the shapes of clusters have horizontal or vertical boundaries but no diagonal boundary is detected. WaveCluster [5] also uses a multidimensional grid structure. It helps in detecting clusters of data at varying levels of accuracy. It automatically removes outliers and is very fast. However, it is not suitable for high dimensional data sets. CLIQUE [6] is a hybrid clustering method that combines the idea of both density-based and grid-based approaches. It automatically finds subspaces of the highest dimensionality and is insensitive to the order of input. Moreover, it has good scalability as the number of dimensions in the data increases. However, the accuracy of the clustering result may be degraded at the expense of simplicity of the method. pMAFIA [7] is an optimized and improved version of CLIQUE. It uses the concept of adaptive grids for detecting the clusters. It scales exponentially to the dimension of the cluster of the highest dimension in the data set.

C. Clustering over Multi Density Data Space

One of the main applications of clustering spatial databases is to find clusters of spatial objects which are close to each other. Most traditional clustering algorithms try to discover clusters of arbitrary densities, shapes and sizes. Very few clustering algorithms show preferable efficiency when clustering multi-density datasets. This is also because small clusters with small number of points in a local area are possible to be missed by a global density threshold. Some clustering algorithms that can cluster on multi-density datasets are Chameleon [8], SNN [9] (shared nearest neighbor), and the multi-stage density-isoline algorithm [10] and so on. Chameleon [8] can handle multi-density datasets, but for large datasets the time complexity is too high. SNN [9] algorithm can find clusters of varying shapes, sizes and densities and can also handle multi-density dataset. The disadvantage of SNN is that the degree of precision is low on the multi-density clustering and finding outliers. The multi-stage density-isoline algorithm [10] clusters datasets by the multi-stage way and the idea of density-isoline. The disadvantage of the algorithm is that each cluster cannot be separated efficiently. DGCL [11] is based on density-grid based clustering approach. But, since it uses a uniform density threshold it causes the low density clusters to be lost.

D. Clustering Over Variable Density Space

Most of the real life datasets have a skewed distribution and may also contain nested cluster structures the discovery of which is very difficult. Therefore, we discuss two density based approaches, OPTICS [12] and EnDBSCAN [13], which attempt to handle the datasets with variable density successfully. OPTICS can identify embedded clusters over varying density space. However, its execution time performance degrades in case of large datasets with variable density space and it can not detect nested cluster structures successfully over massive datasets. In EnDBSCAN [13], an attempt is made to detect embedded or nested clusters using an integrated approach. Based on our experimental analysis in light of very large synthetic datasets, it has been observed that EnDBSCAN can detect embedded clusters; however, with the increase in the volume of data, the performance of it also degrades. EnDBSCAN is highly sensitive to the parameters *MinPts* and ε. In addition to the above mentioned parameters, OPTICS requires an additional parameter i.e. $\varepsilon^{/}$.

E. Massive Data Clustering Using Distributed and Parallel Approach

Parallel and distributed computing is expected to relieve current clustering methods from the sequential bottleneck, providing the ability to scale massive datasets and improving the response time. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged.

In [14], a parallel implementation of the DBSCAN algorithm based on low cost distributed memory multi-computers is presented. Here, a centrally located dataset is spatially divided into nearly equal partitions with minimum overlap. Each such partition is sent to one of the processors for parallel clustering. The clustering results of the partitions are then collected by the central processor in an orderly manner and they are merged together to obtain the final clustering. The algorithm is scalable both in terms of speedup and scale-up and significantly reduces the computation time. In [15], a parallel version of the k-means algorithm was proposed based on shared nothing architecture. This algorithm was designed based on the Single Program Multiple Data (SPMD) model having several processors, each having its own local memory, connected together with a communication network. Another parallel version of DBSCAN, called PDBSCAN [16], also uses a shared-nothing architecture with multiple computers interconnected through a network. Here, as a data structure, the dR*-tree was introduced which is a distributed spatial index structure in which the data is spread among multiple computers and the indexes of the data are replicated on every computer. The master distributes the entire dataset to every slave. Each slave locally clusters the replicated data and the interference between computers is minimized due to local access of data. The slave-to-slave and master-to-slaves communication is done via message passing. The master manages the task of dynamic load balancing and merges the result produced by the slaves. PDBSCAN offers nearly linear speedup and has excellent scale-up and size-up behavior. In [17], a Density Based Distributed Clustering (DBDC) algorithm was presented where the data are first clustered locally at different sites independent of each other. The aggregated information about locally created clusters are extracted and transmitted to

a central site. On the central site, a global clustering is performed based on the local representatives and the result is sent back to the local sites. The local sites update their clustering based on the global model, that is, merge two local clusters to one or assign local noise to global clusters. For both the local and global clustering, density-based algorithms are used. This approach is scalable to large datasets and gives clusters of good quality. In [18], a parallel version of the AutoClass system, P-AutoClass is described. In [19], a Collective Hierarchical Clustering (CHC) algorithm for analyzing distributed and heterogeneous data was presented.

Based on our selected survey and experimental analysis, it has been observed that:

1) Density based approach is most suitable for quality cluster detection over massive datasets.
2) Grid based approach is suitable for fast processing of large datasets.
3) Almost all clustering algorithms require input parameters, determination of which are very difficult, especially for real world data sets containing high dimensional objects. Moreover, the algorithms are highly sensitive to those parameters.
4) Distribution of most of the real-life datasets are skewed in nature, so, handling of such datasets for all types for qualitative cluster detection based on a global input parameter seems to be impractical.
5) None of the techniques discussed above, is capable in handling multi-density datasets as well as multiple intrinsic or nested clusters over massive datasets qualitatively.

F. Motivation

An algorithm which is capable of handling voluminous data and at the same time effectively detects multiple nested or embedded clusters even in presence of noise is of utmost importance. This paper presents a clustering algorithm which can effectively address the previously mentioned clustering challenges. The density-grid clustering algorithm (GDCT) [20] finds clusters according to the structure of the embedding space. For handling massive datasets, a distributed clustering technique is presented which can effectively address the scalability problem. Better speedup and scale-up are the major attractions of the proposed technique.

III. Theoretical Background of the Work

The distribution of data in a data set is not uniform in general. Some portions of the data space are highly dense while some portions are sparse. Therefore, the data space is divided into grid cells and the grid cells whose densities are similar are merged. These similar dense grid cells are together called the *adaptive grid cell*. Once merging of grid cells according to density terminates, a rough cluster is obtained. Thus, adaptive grid cell represents the maximal space that can be covered by the similar dense grid cells. Here, we introduce some definitions which are used in the proposed algorithm:

A. Density Based Approach

Definition 1 *Cell Density*: The number of spatial point objects within a particular grid cell.

Definition 2 *Useful Cell*: Only those cells which are populated i.e., which contain data points will be treated as useful cell.

Definition 3 *Neighbor Cell*: Those cells which are edge neighbors or vertex neighbors of a current cell are the neighbors of the current cell. Fig. 1 shows the neighbor cells (shaded) of the current cell *P*.

Definition 4 *Density Confidence of a cell*: If the ratio of the densities of the current cell and one of its neighbors is less than some β (user input) then β is the density confidence between them. The density confidence plays an important role in cluster formation. For two cells P_1 and Q_1 to be merged into the same cluster the condition, $\beta \leq d_n(P_1) / d_n(Q_1)$ where d_n represents the density of that particular cell, should be satisfied.

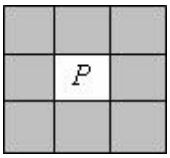

Fig. 1 The white cell is the current cell and all its neighbors are in the gray cells

Definition 5 *Reachability of a cell*: A cell p is reachable from a cell q if p is a neighbor cell of q and cell p satisfies the density confidence condition w.r.t. cell q.

Triangle is a special form of a quadrilateral *i.e*. triangles are degenerated quadrilaterals with two of the vertices merged together. *Triangle-subdivision* is adopted for interpolation of data with better accuracy as compared to that in rectangle. This is because of the fact that partitioning of the data set can be performed more efficiently in triangular shape than in rectangular shape due to its smaller space dimension. The definitions that we have introduced for triangle-subdivision are as follows:

Definition 6 *Triangle Density:* The number of spatial point objects within a particular triangle of a particular grid cell.

Definition 7 *Useful Triangle:* Only those triangles which are populated i.e., which contain data points will be treated as useful triangle.

Definition 8 *Neighbor Triangle*: Those triangles which have a common edge to the current triangle are the neighbors of the current triangle. Figure 2 shows the neighbor triangles (shaded) of the current triangle *P*.

Definition 9 *Density Confidence of a triangle*: If the ratio of the densities of the current triangle and one of its neighbors is less than $\beta/4$ then the two triangles can be merged into the same cluster. Therefore the following condition should be satisfied: $\beta/4 \leq d_n(T_{PI}) / d_n(T_{QI})$ where d_n represents the density of the particular triangle.

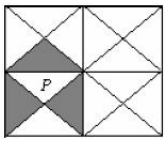

Fig. 2 Neighbor triangles (shaded) of the triangle P

Definition 10 *Reachability of a triangle*: A triangle p is reachable from a triangle q if p is a neighbor triangle of q and triangle p satisfies the density confidence condition w.r.t. triangle q.

Definition 11 *Cluster*: A cluster is defined to be the set of points belonging to the set of reachable cells and triangles. A cluster C w.r.t. β is a non-empty subset satisfying the following condition,

$\forall p, q$: if $p \in C$ and q is reachable from p w.r.t. β, then $q \in C$, where p and q are either cells or triangles respectively.

Both cell-reachability and triangle-reachable relation follows symmetric and transitive property within a cluster C.

Definition 12 *Noise*: Noise is simply the set of points belonging to the cells (or triangles) not belonging to any of its clusters. Let C_1, C_2, ...C_k be the clusters w.r.t. β, then

$$noise = \{no\_p \mid p \in n \times n, \forall i : no\_p \notin C_i\} \quad (1)$$

where no_p is the set of points in cell p and C_i (i=1,...,k).

B. Density Confidence

The density confidence for a given set of cells reflects the general trend of that set. If the density of one cell is abnormal from the others it will not be included in the set. Similarly, each useful cell has a density confidence with each of its neighbor cells. If the density confidence of a current cell with one of its neighbor cell does not satisfy the density confidence condition than that neighbor cell is not included into the local dense area. On the contrary, if it satisfies the condition than we treat the neighbor cell as a part of the local dense area and merge the cell with the dense area. In comparison to other methods of setting a global threshold, this method has the ability to recognize the local dense areas in the data space where multi-density clusters exist.

In light of the above definitions, following *lemmas* are stated.

Lemma 1 Let C be a cluster w.r.t. β and let p be any cell in C. Also, let T_p be a triangle in p. Then C can be defined as the set, $S = \{s \bigcup s_t \mid s$ is cell-reachable from p w.r.t. β and s_t is triangle-reachable from T_p w.r.t. $\beta\}$

Proof: Suppose r is a cell or a triangle, where $r \in s \bigcup s_t$ and r is neither cell-reachable nor triangle-reachable from p w.r.t. β. But, a cluster according to Def. 11 will be the set of points which are cell-reachable or triangle-reachable from p. Therefore, we come to a contradiction and hence the proof. □

Lemma 2 A cell (or triangle) corresponding to noise points is not cell-reachable (or triangle-reachable) from any of the clusters. For a cell p we have, $\forall p$: p is not reachable from any cell (or triangle) in C i.e. $p \notin C$.

Proof: Suppose, C be a cluster w.r.t β and let p be a cell (or triangle) corresponding to noise points. Let p be cell-reachable (or triangle-reachable) from C, then $p \in C$. But, this violates the Def. 12 that noise points are belonging to cells that are neither cell-reachable nor triangle-reachable from any of the clusters. Therefore, we come to the conclusion that p is not reachable from any cell (or triangle) in C. □

Lemma 3 *A cell* (*or a triangle*) *r can be cell-reachable* (*or a triangle-reachable*) *from only a single unique cluster.*

Proof: Let C_1 and C_2 are two clusters w.r.t. β and let p be any cell (or a triangle) in C_1 and q is any cell (or a triangle) in C_2. Suppose a cell r is cell-reachable (or a triangle-reachable) from both p and q, then $r \in C_1$ and $r \in C_2$. This will mean that the clusters C_1 and C_2 should be merged. This violates the basic notion that clusters are unique sets. Thus, we can conclude that if r is cell-reachable (or a triangle-reachable) from p w.r.t. β, r is not cell-reachable (or a triangle-reachable) from q w.r.t. β, *i.e.* $r \in C_1$ and $r \notin C_2$. Therefore the lemma has been proved. □

IV. The Proposed Technique

In this section, we discuss the proposed distributed algorithm. We adopt the shared nothing architecture and consider a system having k-nodes where the entire dataset D is located in any of the nodes (say node 1). Node 1 executes a fast partitioning technique to generate the k initial partitions. The partitions are then sent to k nodes (including itself) for cluster detection using a grid-density based clustering technique (GDCT) which can operate over variable density space. Finally, the local cluster results are received from the nodes at the initiator node (node 1) and a merger module is used to obtain the final cluster results. Basically the technique works in three phases and the output of each phase becomes the input of the subsequent phase.

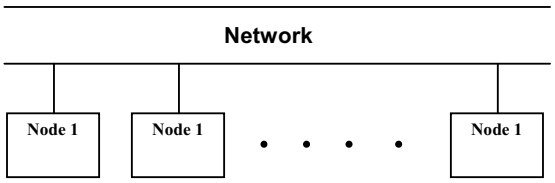

Fig. 3 The Shared-nothing architecture

An overview of the hardware architecture is shown in Fig. 3. It consists of a number of nodes (e.g. PCs) connected via a network (e.g. Ethernet). Next, we describe the architecture as shown in Fig. 4, phase-wise:

A. Phase I: Partitioning the dataset

Phase I of the architecture is executed in one of the nodes (node 1). The dataset is spatially divided into equal size square grid cells and density of each grid cell is computed. The square mesh is then partitioned with some overlap between adjacent partitions and distributed over k available computers (nodes). No subsequent movement of data between partitions will take place.

Initially, the data space is divided into $n \times n$ non-overlapping square grid cells, where n is a user input, and maps the data points to each cell. It then calculates the density of each cell.

Assuming, the grid mesh D contains the set of $n \times n$ objects say, $D = O_0, O_1, O_2, ..., O_{(n \times n)-1}$. Suppose, $O_j = (a_{0j}, a_{1j}, a_{2j}, ..., a_{(d-1)j}; d_n)$ represents a grid cell with d real-valued attributes a_i, i=0,….,d-1 and density d_n. The i^{th} attribute value of object O_j is drawn from domain a_j. If there are k clients, the grid mesh D is partitioned into k subsets $D_0, D_1, ..., D_{k-1}$ ordered in sequence. We refer the clients by the corresponding partition D_j that it receives for processing.

$$D = D_0 \cup D_1 \cup D_2 \cup ..... \cup D_{k-1}$$
$$D_i \cap D_j \neq \phi,$$
$$i, j = 0,...., k-1 \quad (2)$$

The partially overlapped partitions are shown in Fig. 5 for 2D case. An overlap of one grid cell occurs between two adjacent partitions. The overlapped regions are much smaller than the partitions. The grid cells in the overlapped regions are locally clustered in both the adjacent partitions. Thus they provide the information for merging together the local clustering results of two adjacent partitions. The overlapped width should be at least one cell width because adjacent cells are neighbors according to *Definition* 3.

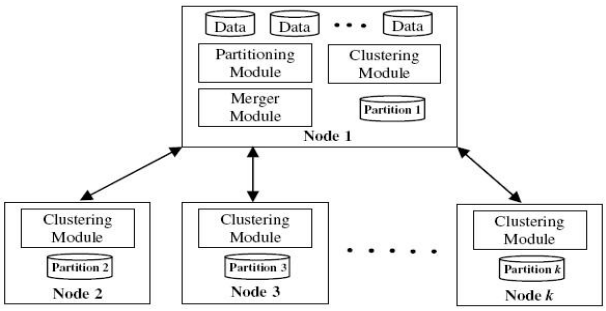

Fig. 4 The architecture of the Proposed Technique

The grid mesh D is partitioned in this manner based on the values of a selected attribute of the data objects say a_s. The values of a_s have a range of $[min\_a_s, max\_a_s]$. We need to select $(k+1)$ constants in the given range. Let c_i, $i = 1,\ldots, k+1$ represent the constants such that $c_i = min\_a_s$, $c_{k+1} = max\_a_s$ and $c_i < c_{i+1}$. Therefore the overlapped region can be represented as:

$$D_i = \{\exists j(O_j \in D) \mid c_i - cell\_width \leq a_{sj} \leq c_{i+1}\},$$
$$i = 2,..., k-1 \quad (3)$$
$$D_i = \{\exists j(O_j \in D) \mid c_i \leq a_{sj} \leq c_{i+1} + cell\_width\},$$
$$i = 1 \quad (4)$$
$$D_i = \{\exists j(O_j \in D) \mid c_i - cell\_width \leq a_{sj} \leq c_{i+1}\},$$
$$i = 1 \quad (5)$$

The constant c_j should be selected in such a manner that cardinality of set D_j becomes nearly equal to $\lceil N/k \rceil$, where N is total number of data points in the dataset.

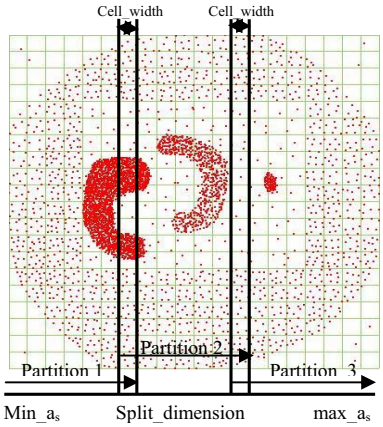

Fig. 5 Overlapped spatial partitioning of a 2D data set

Moreover, those grid cells which fall within the overlapped regions are marked. Care has been taken for load balancing. The k partitions thus obtained are then sent to k nodes for global as well as intrinsic cluster detection (Fig. 6).

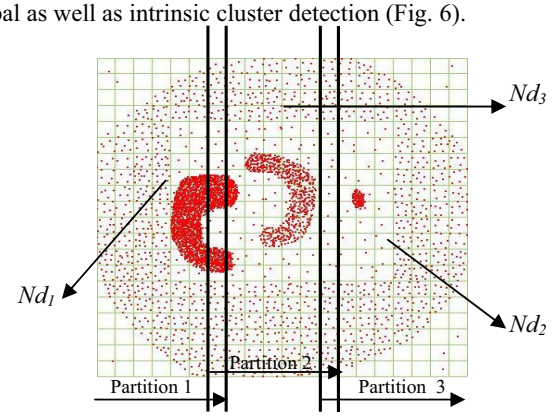

Fig. 6 Here the dataset is divided into three partitions and transmitted to three computers (Nd_k) for local clustering, $k = 3$

A previous version of the clustering algorithm to detect intrinsic clusters is given in [20]. However, it was not scalable to huge datasets and there was no precise method to calculate the number of grid interval.

Computing the Number of Grid Intervals (n)

The following formula is used to calculate the number of intervals n.

$$n' = \sqrt{\frac{N}{M}} \quad (6)$$
$$n = [n' - 5, n' + 5] \quad (7)$$

where N is the number of data points and M is a coefficient

to adjust the value of *n'*. It is a positive integer. In fact, it stands for the average number of data samples in a cell.

A detailed experimentation on the number of data points (*N*) and the coefficient *M* has been carried out and the graph is shown in Fig. 7. The value of *n'* is calculated as in Eq. (6). Based on our wide range of experiments, it is observed that *n* varies within the range as given in Eq. (7).

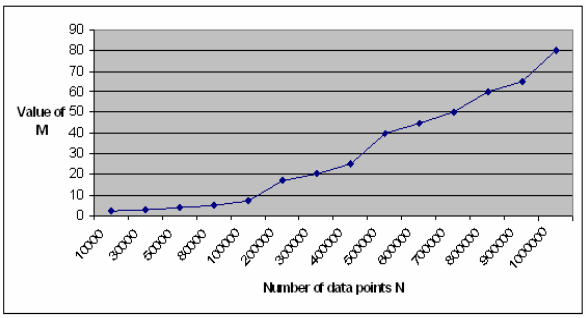

Fig. 7 *M* depends on the number of data points

Load Balancing

Partition D_i is sent to processor P_i, =1,2,..,*k* for concurrent clustering. Since no data movement takes place after the partitions are created, care should be taken so that each processor receives nearly equal number of data objects for processing. This will ensure that all the processors finish the clustering job at the same time provided the processors have same processing speed. If the processing speeds are different, then the input data should be distributed to the processors proportionate to their processing speed. We assume that the processing speeds of the processors are equal, so they receive nearly equal amount of data. For doing this the range of a_s is divided into intervals of width of one *cell-width* and the frequencies of data in each interval is counted.

$$\text{Let } b = \lceil (\max\_a_s - \min\_a_s) / cell\_width \rceil$$
$$N' = \lceil N/k \rceil$$
$$d_1 = \min\_a_s$$
$$d_i = d_{i-1} + cell\_width, \quad i = 2,3,...,b$$
$$F_i = \{\exists j (O_j \in D) \mid d_i \le a_{sj} \le d_{i+1}\}, \quad i = 2,3,..., b \tag{8}$$

f_i = Cardinality of set F_i

Now, the constants c_i defined earlier are computed as $c_i = d_s$ such that

$$\sum_{j=1}^{s} f_j \le i.N' \le \sum_{j=1}^{s+1} f_j, \quad i = 1,2,..., k \tag{9}$$

which will ensure that each partition gets number of objects nearly equal to *N/k*.

Minimized communication cost

The proposed method saves transmission cost by avoiding inter-node communication during the process of local clustering. To achieve this goal, each concurrent process of GDCT in each of the nodes, *Nd* = 1,2,…,*k*, should avoid accessing those data located on any of the other computers, because the access of the remote data requires some form of communication. Therefore, nearby objects should be organized on the same computer. This is why an overlap of one *cell_width* has been taken into consideration.

B. Phase II: Local Clustering

Phase II of the architecture is executed in each of the *k* nodes. This phase plays the actual role of clustering. In this phase, each node executes the proposed algorithm, GDCT over the partition of data received from the initiator node to detect the global and nested clusters. The aim of our clustering algorithm unlike our previous version [20] is to discover intrinsic as well as global clusters over large spatial datasets of variable density.

Grid-Density Clustering using Triangle Sub-division (GDCT)

In any node, the cells of the partition received are sorted according to their density values. The result is an ordered sequence $<C_{P(i)}>$, where $P(i)$ denotes a permutation of the index *i* defining the sorted order of the cells *C*. The algorithm uses the cell information (density) of the grid structure and clusters the data points according to their surrounding cells.

The cell with the highest density becomes the cluster initiators. The remaining cells are then clustered iteratively in order of their densities, thereby building new clusters or merging with existing clusters. The useful cells adjacent to a cluster can only be merged. A neighbor search is conducted, starting at the highest density cell and inspecting adjacent cells. If a neighbor cell is found which satisfies the density confidence condition of a cell, then the neighbor cell is merged with the current cell to form the adaptive grid, and the search proceeds recursively with this neighbor cell. This search is similar to a graph traversal where the nodes represent the cells and an edge between two nodes exists if the respective cells are adjacent and satisfies the density confidence condition of a cell.

The adaptive grid formed is an approximation of the innermost cluster or the cluster with the maximum density, minus the boundary region. The cells falling inside a particular adaptive grid are classified with the same cluster id. The adaptive grid will reflect the rough cluster formed.

The cluster shape in the boundary region of the cluster varies more since there is a transition from denser region to sparser region when we are considering intrinsic or variable density clusters. Therefore, this region needs special analysis. So, after the adaptive grid is formed, there might still be some points of the approximate clusters that lie outside the adaptive grid as shown by the red ellipse (black color ellipse for gray scale images) in Fig. 8. Since the points inside these regions

do not enter the adaptive grid though they are a part of the cluster, we therefore expand the cell in the boundary region with the help of triangles. The points in the boundary region of the cluster have been left out because the cells in which they reside have not satisfied the density confidence of a cell with its neighbor belonging to the adaptive grid so formed. This is because only a small portion of that part of the cluster has fallen in a different cell. Therefore the density of that cell is much less than its adaptive grid neighbor.

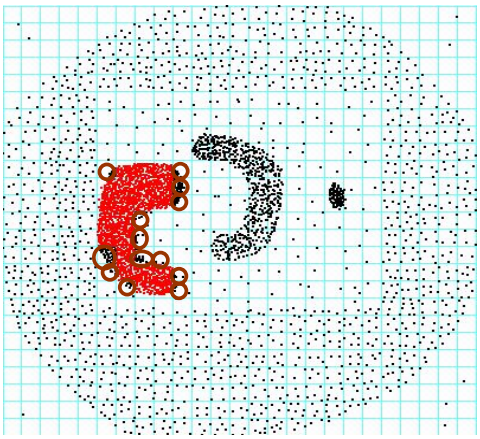

Fig. 8 Example grid approximation for a dataset (n = 25)

Therefore, for finding the finer clustering, a cell is triangulated i.e. the cell is divided into four triangles. Only those cells in the adaptive grid are triangulated which have at least one of its useful neighbor cells as unclassified. The cells which are unclassified and have at least one of its neighbor cells belonging to the most recent adaptive grid formed are also triangulated. The data points of the cells that have been triangulated are mapped to the respective triangles in which they fall. The Barycentric coordinates [21] have been used for finding which point falls in which triangle. This method has been chosen since it is independent of the cyclic order of the vertices.

Procedure of GDCT

The execution of the algorithm includes the following 9 steps:

1) Create the grid structure.
2) Compute the density of each cell.
3) Sort the cells according to their densities.
4) Identify the maximum dense cell from the set of unclassified cells.
5) Traverse the neighbor cells starting from the dense cell and form the adaptive grid (rough cluster).
6) Triangle-subdivision of the border cells of the adaptive grid which has at least one of its neighbors as a useful cell.
7) Triangle-subdivision of the unclassified neighbor cells of those border cells.
8) Merge the triangles and assign cluster_id.
9) Repeat steps 4 through 9 till all cells are classified.

The process of forming the adaptive grid starts by considering the cell P_1 with the maximum density from the sorted list. From P_1, the first adaptive cell expands to the neighboring cells P_{1i} (where cell P_{1i} is the i^{th} neighbor of P_1) depending upon two conditions which are

1) If P_{1i} is not a member of any adaptive cell, and
2) The densities of P_1 and P_{1i} differ by some threshold β which is an input parameter.

Let, $d_n(P_1)$ and $d_n(P_{1i})$ denote the densities of P_1 and P_{1i} respectively, then P_{1i} will merge with P_1, if $\beta \leq d_n(P_1)/d_n(P_{1i})$. The cells that satisfy the conditions given above are merged to form the adaptive cells. The process of adaptive cell formation continues from P_{1i} in the same way until no neighboring cells P_{1j} of P_{1i} satisfy the condition. The process then backtracks to P_{1i} and the process restarts with the next neighbor cell of P_{1i} which has not already been processed. The adaptive grid formation continues recursively until no more cells satisfy the density confidence condition of a cell.

This adaptive grid is an approximation of the cluster with the maximum density. The cells falling inside that particular adaptive grid are classified with the same *cluster_id* which reflects the rough cluster. The process then checks the neighbors of the last formed adaptive grid cells. If any one of the neighbors is an unclassified useful cell then both the adaptive grid cell as well as the unclassified neighbor cell is triangulated. Suppose P_m is a cell of the adaptive grid last formed and cell P_i is one of it's unclassified useful neighbor cell where $P_i \in \{P_{i1}, P_{i2},\ldots, P_{i8}\}$. Then P_i as well as P_m is then triangulated in a manner as shown in Fig. 9. During Triangle-subdivision, a particular grid cell is divided into four triangles.

Each of the triangles T_{ki} inside the cell P_i is verified for the following cases:

Case 1: If T_{ki} has a neighbor triangle T_{mi} which is a part of adaptive grid cell P_m, then their densities $d_n(T_{mi})$ and $d_n(T_{ki})$ are compared for the density confidence condition of a triangle given as, $\beta / 4 \leq d_n(T_{mi}) / d_n(T_{ki})$. If this condition is satisfied, then triangle T_{ki} is merged with the triangle T_{mi} of the adaptive grid and obtains the *cluster_id* of P_m.

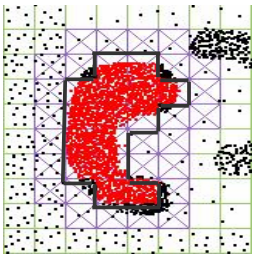

Fig. 9 Triangle-subdivision of grid cells (black polygon shows the adaptive grid or a rough cluster)

Case 2: T_{ki} has a neighbor triangle T_{ji} which has already been classified and the densities of T_{ki} and T_{ji} satisfy the condition given in case 1, then T_{ki} will be merged with T_{ji} and T_{ki} will be classified with the same *cluster_id* as T_{ji}.

The process of triangle merging stops when no more triangles satisfy the density confidence condition of a triangle.

The process then starts the next adaptive cell formation from the next cell P_2 which is the cell of maximum density from the set of unclassified cells. The process continues recursively merging neighboring cells that satisfy the density confidence condition of a cell. Therefore, the adaptive grid formation and triangle-subdivision method are repeated alternately till all the useful cells have been classified. The classified cells and triangles will now give the distinct clusters and finally the data points receive the *cluster_id* of the respective cells and triangles.

The cluster expansion based on the set of cells detects embedded and nested cluster structures since after expansion of a cluster the algorithm searches for the next candidate seed cell which reflects a variation in density in the dataset. The process starts expanding the new density region till there is again a density variation. This process iterates till all the cells have been classified. The triangle expansion gives a finer clustering result since the cluster expansion based on cells misses some border points as can be seen in Fig. 8. The expansion based on triangle-subdivision detects the border points which have been left out by cell based expansion. Therefore, the quality of the clusters becomes highly accurate in addition to detecting intrinsic and multi-density clusters.

During clustering, it considers only the grid cells to identify the possible global and embedded clusters and assigns *cluster_id* accordingly. For the partition D_i in node i, the grid cells in it will be assigned *cluster_id* according to the clusters formed in that partition. The *cluster_id* will be used during the server based merging process.

The cluster expansion based on grid cells reduces the computation time as all the data points are not considered for cluster expansion only the density information of each cell is used. Moreover, the cluster_id information is used during Phase III merging process. It saves the cost of merging to a great extent. Finally, Phase II transmits the cluster objects to the server along with the cluster_id information.

C. *Phase* III: *Merging*

In Phase III, the cluster results received from the k nodes undergo a simplified, yet faster merging procedure to obtain the final clusters. Since the Phase II process in a node may yield more than one cluster along with the embedded clusters, so there are always possibilities for merging during Phase III operation. The Merger module works as follows:

1) Join the partitions received from the k nodes according to their overlapping marks.
2) Consider the marked grid cells (overlapping cells) of the candidate clusters.
3) If any of the marked grid cells is identified by different *cluster_id*s by different partitions (say l, m), then assign any one of the *id*s (say l) to that cell.
4) Assign all those cells having the same *cluster_id* as the replaced id (m) with l.

D. *Complexity Analysis*

Phase I: The partitioning of the dataset into $n \times n$ non-overlapping cells results in a complexity of $O(N)$ where N is the total number of data points. The grid mesh D is spatially partitioned into k partitions with overlap of one cell width which results in a complexity of $O(n \times n)$, where $n << N$. Each of these k partitions will have nearly equal (approximately N/k) data points. The data points along with the grid information for each of k partitions will be sent to the k nodes. Therefore $(N/k) + t$ points will be sent, where t is the average number of points present in an overlapped region. Next, to transmit these $(N/k) + t$ points to each node requires a communication time of $O((N/k) + t)$.

Phase II: This phase is executed in each of the k nodes. Computing density of the cells in each node requires $O((n \times r) \times ((N/k) + t))$, where r is the average number of cells along the selected attribute based on which partitioning in Phase I has been performed. The sorting of cells according to their density results in a complexity of $O((n \times r) \log (n \times r))$.

The expansion of the adaptive grid results in $O(m)$ time complexity, where m is the number of cells in an adaptive grid formed and $m << (n \times r)/k$ in the average case. Cell subdivision into triangles takes place only in case of the border cells of the adaptive grid and its neighboring cells, Say, there are p border and q neighbor cells where $q >> p$. This step results in a complexity of $O(p+q)$. If the number of clusters obtained is n_c then the overall time complexity for the clustering will be $O(n_c \times m \times (p+q))$.

Therefore, total time complexity will be $O((n \times r) \times ((N/k) + t)) + O((n \times r) \log (n \times r)) + O(n_c \times m \times (p + q))$. Thus the complexity due to density calculation almost dominates the other components, since $(N/k) + t) >> (n \times r)$. The clusters detected in this phase are transmitted back to the initiator node with a transmission cost of $O((N/k) + t))$.

Phase III: Merging of the clusters obtained from the k nodes will take $O(N+k.t)$ time.

Thus, the overall time complexity of distributed GDCT will be $O(N) + O(n \times n) + O((N/k) + t)) + O((n \times r) \times ((N/k) + t)) + O((N/k) + t))$. Therefore, the time complexity becomes $O(N)$ since $N >> (n \times n)$.

Advantages of proposed distributed algorithm

The advantages of the proposed algorithm are:

1) Embedded cluster Detection,
2) $O(N)$ complexity,
3) Handling of huge datasets,
4) Handling of single linkage problem.

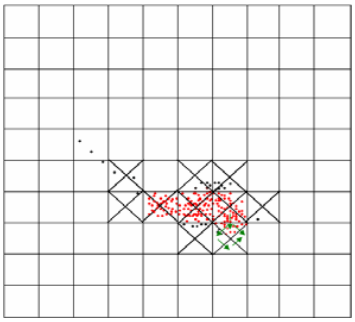

Fig. 10 The arrows show triangle reachability

The first three advantages can be understood from Sections IV(*A*), IV(*B*), IV(*D*). For the fourth point we consider Fig. 10. Once the adaptive grid has been formed, the triangle sub-division process starts. For neighbour traversal in triangles there has to be at least a common edge between triangles. Two triangles will be merged according to Def. 9. As can be seen in figure the chain of single points will not be merged as they do not satisfy Def. 10. The final cluster obtained is shown in Fig. 11. Thus, the single linkage problem which affects DBSCAN does not affect the proposed algorithm.

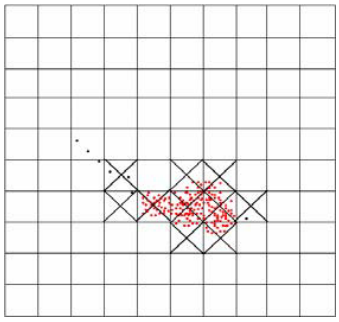

Fig. 11 Single linkage problem handled

V. Performance Evaluation

To evaluate the technique in terms of quality of clustering, we used the synthetic data set generated as shown in Fig. 12.

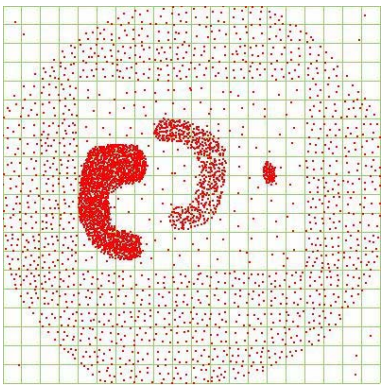

Fig. 12 Synthetic Dataset

The results of the synthetic dataset in Fig. 12 are shown in Fig. 13.(a) and 13.(b).

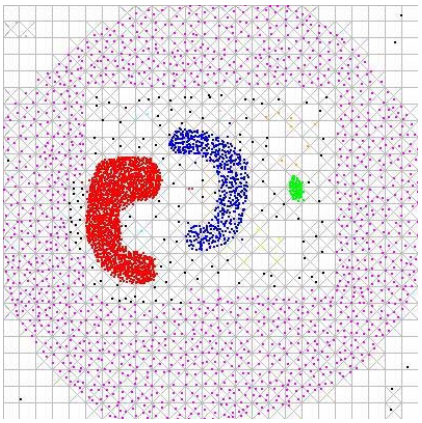

Fig. 13.(a) After full expansion

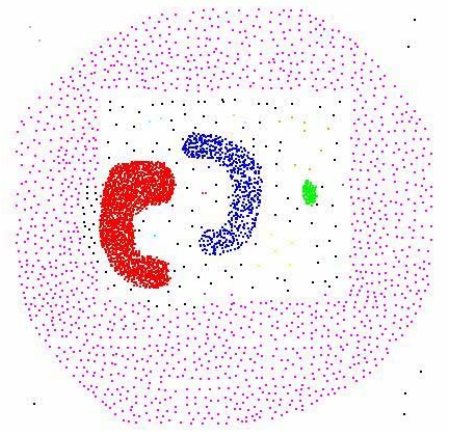

Fig. 13.(b) Final five clusters

The algorithm was experimented with several synthetic datasets generated and the result of one of them is shown in Fig. 14.

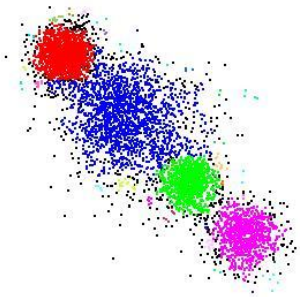

Fig. 14 Final four clusters

The algorithm was also applied on the Chameleon t4.8k.dat and t7.10k.dat datasets [9]. The results obtained are shown in Fig. 15(*a*) and 15(*b*) respectively. The result obtained when

the algorithm was applied on t5.8k.dat dataset is shown in Fig.16. From our experiments it has been found that the clustering result is dependent on the threshold β which varies in the interval [0.5, 0.7].

Fig 15.(a) t4.8k.dat dataset

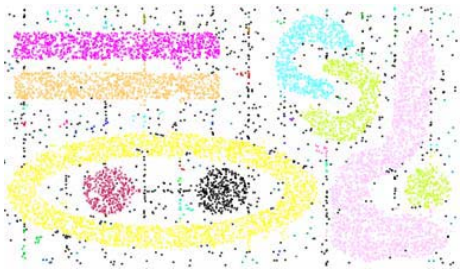

Fig. 15(b) t7.10k.dat dataset

Fig. 16 Clusters obtained from t5.8k.dat dataset

From the experimental results given above, we can conclude that GDCT is highly capable of detecting intrinsic as well as multi-density clusters qualitatively.

A. Performance and Scalability Analysis

A sequential algorithm is evaluated in terms of its execution time which is expressed as a function of its input size. On the other hand, the execution time of a distributed algorithm depends not only on the input size but also on the distributed architecture and the number of processors employed. By adding more processors we would like to decrease the execution time or increase the volume of data handled by using more processors. This section reports an empirical study on the characteristics of the proposed distributed algorithm by measuring execution time, speedup, efficiency and scale-up factors.

Since there is no inter-processor communication except for a single processor communicating with each of the remaining processors. Each processor has the same specification i.e. PIV with 1 GHz speed and 128 MB RAM and the processors are connected through Ethernet LAN of speed 10/100 Mbps. To smooth out any variation, each experiment was carried out for five times and the average result were taken and each reported data point is to be interpreted as an average over five measurements. Our implementation is in C in Linux environment. Next, we generated several synthetic datasets containing arbitrary number of arbitrary shaped clusters having 2,00,000, 4,00,000, 6,00,000, 8,00,000 and 10,000,000 objects respectively and experimentation was carried out.

Parallel Execution Time: The parallel execution time, denoted by $T(k)$, of a program is the time required to run the program on a k-processor parallel computer. When $k = 1$, $T(1)$ denotes the sequential run time of a program on a single processor. From our experiments we conclude that the execution time decreases significantly as the number of processors increase.

Speedup: Speedup is a measure of relative performance between a multiprocessor system and a single processor system, defined as, $S(k) = T(1)/T(k)$. On experimenting it has been found that the speedup factor increases with the increase in the number of processors. The relation between speedup and the number of processors used is shown in Fig. 17.

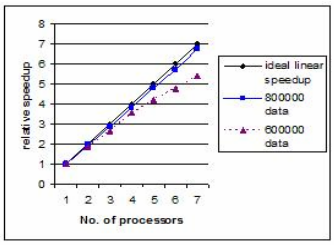

Fig. 17 Relative Speedup curves for two data sets with points $N = 8\times10^5$ and 6×10^5. The number of dimensions and the number of clusters are fixed for both the data sets. The solid line represents "ideal" linear relative speedup. For each data set, a dotted line connects observed relative speedup.

Scale-up: Scale-up measures how well the parallel algorithm handles large datasets as the number of processors increases. The scale-up characteristic of the proposed algorithm has been found to be satisfactory with the increase in the number of processors as can be seen from Fig. 18.

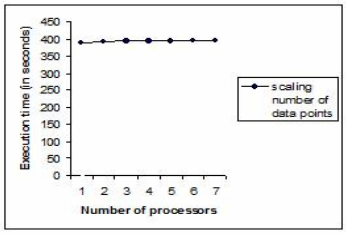

Fig. 18 Scale-up curve: The number of data points is scaled by the number of processors while dimensions and number of clusters are held constant.

VI. Comparison of GDCT With its Counterparts

DBSCAN requires two input parameters *MinPts* and ε.

TABLE I
COMPARISON OF THE PROPOSED ALGORITHM WITH ITS COUNTERPARTS

Algorithms	No. of Parameters	Multi-Density Clusters	Embedded Clusters	Complexity
DBSCAN	2 (*MinPts,* ε)	No	No	O(*N log N*) using R* tree
OPTICS	3 (*MinPts,* ε, ε')	Yes	Yes	O(*N log N*) using R* tree
Proposed Algorithm	2 (n, β)	Yes	Yes	O(*N*)

Moreover, it cannot detect embedded clusters. OPTICS on the other hand, requires three input parameters *MinPts*, ε and $\acute{\varepsilon}$. But, it can detect embedded clusters. However, its performance degrades while detecting multiple nested clusters over massive datasets. Again, GDLC and Density-isoline algorithms can detect multi-density clusters but fail to detect intrinsic cluster structures. GDCT requires the number of grid cells, i.e. n and threshold β as input parameters. Moreover, from our experiments we conclude that the threshold β does not vary significantly with different datasets. GDCT can effectively detect embedded clusters over variable density space as well as multiple nested clusters. A detailed comparison is given in Table I.

VII. CONCLUSION

This paper presents a distributed clustering technique for massive numeric datasets. The clustering algorithm is based on a grid-density approach and can detect global as well as embedded clusters qualitatively by sharing the computational efforts among k processors. Experimental results are reported to establish the superiority of the algorithm in light of several synthetic data sets. Results in terms of scale-up and speedup are reported to establish the superiority of the technique in light of several synthetic datasets. In this paper we have only considered two-dimensional objects. But, spatial databases also contain extended objects such as polygons. Therefore, there is scope for scaling GDCT to detect clusters in such datasets with minor modifications, research of which is in progress.

REFERENCES

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* India: Morgan Kaufmann Publishers, 2004.
[2] M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in *International Conference on Knowledge Discovery in Databases and Data Mining* (KDD-96), Portland, Oregon, 1996, pp. 226-231.
[3] C. Hsu and M. Chen, "Subspace Clustering of High Dimensional Spatial Data with Noises", *PAKDD*, 2004, pp. 31-40.
[4] W. Wang, J. Yang, and R. R. Muntz, "STING: A Statistical Information Grid Approach to Spatial data Mining", in *Proc.* 23rd *International Conference on Very Large Databases*, (VLDB), Athens, Greece, Morgan Kaufmann Publishers, 1997, pp. 186 - 195.
[5] G. Sheikholeslami, S. Chatterjee and A. Zhang, "Wavecluster: A Multi-resolution Clustering approach for very large spatial database", in *SIGMOD*'98, Seattle, 1998.
[6] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications", in *SIGMOD Record ACM Special Interest Group on Management of Data*, 1998, pp. 94–105.
[7] H. S. Nagesh, S. Goil and A. N. Choudhary, "A scalable parallel subspace clustering algorithm for massive data sets", in *Proc. International Conference on Parallel Processing*, 2000, pp. 477.
[8] L. Ertoz, M. Steinbach and V. Kumar, "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data", in *SIAM International Conference on Data Mining* (SDM '03), 2003.
[9] G. Karypis, Han and V. Kumar, "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling", *IEEE Computer*, 32(8), pp 68-75, 1999.
[10] Y. Zhao, S. Mei, X. Fan, S. Jun-de. 2003. Clustering Datasets Containing Clusters of Various Densities. *Journal of Beijing University of Posts and Telecommunications*, 26(2):42-47.
[11] H. S. Kim, S. Gao, Y. Xia, G. B. Kim and H. Y. Bae, "DGCL: An Efficient Density and Grid Based Clustering Algorithm for Large Spatial Database", *Advances in Web-Age Information Management* (WAIM'06), pp. 362-371, 2006.
[12] M. Ankerst, M. M. Breuing, H. P. Kriegel and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure", in *ACM-SIGMOD*, pp. 49-60, 1999.
[13] S. Roy and D. K. Bhattacharyya, "An Approach to Find Embedded Clusters Using Density Based Techniques", in *Proc. ICDCIT*, LNCS 3816, pp. 523-535, 2005.
[14] B. Borah, D. K. Bhattacharyya and R. K. Das, "A Parallel Density-Based Data Clustering Technique on Distributed Memory Multicomputers", in *Proc. ADCOM*, Ahmedabad, 2004.
[15] I. S. Dhilon and D. S. Modha, "A Data-Clustering Algorithm on Distributed Memory Multiprocessors", in *International Conference on Knowledge Discovery and Data Mining* (SIGKDD 99), 1999.
[16] X. Xu, J. Jager and H. P. Kriegel, "A Fast Parallel Clustering Algorithm for Large Spatial Databases", *Data Mining and Knowledge Discovery*, 3, Kluwer Academic Publisher, pp. 263-290, 1999.
[17] E. Januzaj, H. P. Kriegel and M. Pfeifle, "Towards Effective and Efficient Distributed Clustering.Workshop on Clustering Large Data Sets", *ICDM*'03.Melbourne, Florida, 2003.
[18] D. Foti, D. Lipari, C. Pizzuti and D. Talia, "Scalable Parallel Clustering for Data Mining on Multicomputers",15 IPDPS workshops, pp. 390-398, 2000.
[19] E.K. Johnson and H. Kargupta, "Collective Hierarchical Clustering from Distributed, Heterogeneous Data", *Large Scale Parallel data Mining*, LNCS 1759, Springer, 2000.
[20] S. Sarmah, R. Das and D. K. Bhattacharyya, "Intrinsic Cluster Detection Using Adaptive Grids", in *Proc. ADCOM'07*, Guwahati, 2007.
[21] Available: http//steve.hollasch.net /cgindex/math /barycentric.html