ISSN: 2456-8848, Vol. I, Issue. II, Sept-2017

International Journal
of
Linguistics & Computing Research

Spatial Data Clustering: DBSCAN & GDBSCAN

Girish Kumar Singh
Department of Computer Science and Application
Dr. Harisingh Gour Central University
Sagar, Madhya Pradesh, India
gkrsingh@gmail.com

Abstract: Spatial data mining is to mine high-level spatial information and knowledge from large spatial databases. Spatial data consists of spatial objects made up of points, lines, regions, rectangles, surfaces, volumes, and even data of higher dimensions which also include time. Spatial data mining algorithms heavily depend on the efficient processing of neighborhood relations since the neighbors of many objects have to be investigated in a single run of a typical algorithm. Spatial clustering, which groups similar spatial objects into classes, is an important component of spatial data mining. In this paper two most popular spatial data clustering algorithms viz. DBSCAN and GDBSCAN has been studied and also suggest an improvement to these clustering algorithms.

I. INTRODUCTION

Spatial data mining is the analysis of geometric or statistical characteristics and relationships of spatial data which may exist implicitly. In particular, these methods can be used for understanding spatial data, discovery of relationships between spatial and non spatial data, construction of spatial knowledge-bases, query optimization, data reorganization in spatial databases, and capturing the general characteristics in a simple and concise manner. This has wide applications in GIS, remote sensing, image databases exploration, medical imaging, robot navigation, and other areas where spatial data are used. In the last few years a growing interest in practical applications of spatial data mining could be observed in the areas such as marine ecology [1], space exploration [2], remote sensing [3], traffic analysis [4], and climate research [5]. With further development of spatial data mining methods, the number of applications should increase further. Following examples presents some possible applications of Spatial Data Mining.

i. Analysis of Crime Sites: A large number of robberies are committed in a large metropolitan area. A criminologist who analyzes of these robberies may visualize crime sites using GIS system and use maps presenting the locations of other objects, to find the patterns. The analysis is performed using a large number of queries and data transformations to produce buffer zones around the crime sites. A statistical package may be used to analyze relationships.

ii. Real estate investment analysis: A financial analyst researches the price changes of the houses in a local market. A regression model is constructed based on the characteristics, which were extracted using a number of GIS operations. Such a model is used to predict the changes of the price of new investments. Also, general statistics are produced to present the influence of different variables on the price.

iii. Analysis of the regional sales: A market researcher analyzes sales of a national company based on the lines of products, time, the area of sales and profits. An OLAP system is employed and cross tables are used for visualization.

iv. Creation of Thematic Maps in Geographic Information System: Map showing information on only one topic. E.g., temperature, rainfall, population, etc. is called *Topological map*. In Geographic Information System a single map represents more than one topic. For example a map of India, shows the various information like boundary of the states, railway lines, rivers, forest areas, industrial areas etc. Using spatial data clustering we can make thematic map from the map used in Geographic Information System.

Steps in Spatial Data Mining: Discovery of knowledge in spatial databases is a complex process, which covers many interrelated steps. These steps are connected through the feedback loop based on the results obtained in the discovery process. Some of the steps can be outlined as the follows:

i. Learning the application domain: which may include studying prior knowledge, analyzing the goals of the mining process, and building concept hierarchies.
ii. Data cleaning and data preprocessing: which includes removal of noise and outliers if appropriate, unifying different formats of the data and different data source, handling missing data fields, and sampling.
iii. Construction of data warehouses for On-Line Analytical Processing (OLAP), which includes aggregating the data and materializing views.
iv. Choosing the data mining algorithm that analyzed data. It includes the decision on the purpose of the model and rules derived by the data mining process. The algorithms for data mining may include clustering, classification, association, prediction, etc.
v. Selection of the data items, dimensions, attributes, and measures used in the mining process.
vi. Data reduction and projection, which includes finding good data representation, reduction of data dimension, focusing on relevant items of data and relevant attributes.
vii. Pattern extraction, which includes finding rules, patterns, and models through data mining algorithms.
viii. Interestingness analysis, which filter out the uninteresting patterns.
ix. Visualization of the discovered knowledge through tables, rules, graphs, charts, maps, diagrams, etc.
x. Application of the discovered knowledge through documenting it, reporting it to be the users, taking actions based on the discovered rules, and resolving conflicts with previously believed information.

Clustering is the process of grouping a set of objects into classes or clusters so that objects within a cluster have high similarity in comparison to one another, but are dissimilar to objects in the other clusters. As branch of statistics, cluster analysis has been studied extensively for many years, focusing mainly on distance-based cluster analysis. Clustering has also been studied in the field of machine learning as a type of unsupervised learning because it does not rely on predefined class and class-labeled training examples. However, efforts to perform effective and efficient clustering on large database only started in recent years with the emergence of data mining. Based on survey, it is strongly felt that there are ample scopes for the researchers to explore this important area, especially in the context of spatial data domain.

Rest of the paper is organized as follows: The organization of the report is as follows:

1. To carry out an experimental study on some of the popular spatial data clustering algorithms.
2. To analyze the pros & cons of two popular spatial data clustering algorithms viz. DBSCAN & GDBSCAN.
3. To propose an enhancement /improvement in the existing two popular spatial data clustering algorithms i.e. DBSCAN & GDBSCAN.
4. To explore for a suitable application of the proposed improved version of the clustering algorithm in the content base image retrieval.

II. SPATIAL DATA CLUSTERING

Spatial clustering, which make groups of similar spatial objects, is an important component of spatial data mining. Spatial clustering can be used in the identification of areas of similar land usage in an earth observation database or in merging regions with similar weather patterns, etc. As a data mining function, spatial clustering can be used as a stand-alone tool to gain insight into the distribution of data, to observe the characteristics of each cluster, and to focus on a particular set of clusters for further analysis. It may also serve as a preprocessing step for other algorithms, such as classification and characterization, which will operate on the detected clusters.

Due to its immense applications in various areas, spatial clustering has been a highly active topic in data mining research, with fruitful, scalable clustering methods developed. These algorithms mostly work on numerical attributes and can be separated into four general categories: partitioning method, hierarchical method, density-based method and grid-based method.

The partitioning algorithm typically starts with an initial partition of the database and then uses an iterative control strategy to optimize an objective function. Each cluster is represented by the gravity center of the cluster or by one of the representative objects located near center of the cluster. The shape of the clusters found by a partitioning algorithm may be convex which is very restrictive. There are many partitioning methods like *k*-mean algorithm [6], EM (Expectation Maximization) algorithm [7], PAM (Partition around Medoid, *k*-medoid) algorithm [8], CLARA [8], CLARANS [13] etc.

Hierarchical algorithms create a hierarchical decomposition of the database *D*. The hierarchical decomposition is represented by a dendrogram, a tree that iteratively splits *D* into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of *D*. The dendrogram can either be created from the leaves up to the root (agglomerative approach) or from the root down to the leaves (divisive approach) by merging or dividing clusters at each step.

The grid-based clustering approach uses a multi resolution grid structure. It quantizes the space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space. Some typical examples of the grid-based approach include STING [10], which explore statistical information stored in the grid cells; Wave Cluster, which clusters objects using a wavelet transform method; and CLIQUE [11], which represents a grid and density-based approach for clustering in high-dimensional data space.

The general idea is to continue growing the given cluster as long as the density in the neighborhood exceeds some threshold; that is for each data point within a given cluster, the neighborhood of given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outlier) and discover clusters of arbitrary shape.

Ester et al. present the density-based clustering algorithm DBSCAN [12]. Unlike CLARANS, DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a locality-based algorithm, relying on a density-based notion of clustering. The density-based notion of clustering states that within each cluster, the density of the points is significantly higher than the density of points outside the cluster. The algorithm uses two parameters, Eps and *MinPts* to control the density of the cluster. The Eps-neighborhood of a point is defined by $N_{Eps}(p)=\{\, q \in D \mid dist\,(\,q\;p) \leq NEps \,\}$. The distance function dist(*p,q*) determines the shape of the neighborhood. *MinPts* is the minimum number of points that must be contained in the neighborhood of that point in the cluster. DBSCAN starts with an arbitrary point and retrieves all points with the same density reachable from the point using *Eps* and *MinPts* as controlling parameters. If the point is a core point, then the procedure yields a cluster. If the point is on the border, then DBSCAN goes on to the next point in the database. The algorithm may need to be called recursively with a higher value for *MinPts* if "close" clusters need to be merged because they are within the same Eps threshold.

DBSCAN is designed to handle the issue of noise and is successful in ignoring outliers, but although it can handle shapes that are hollow, there is no indication that it can handle shapes that are fully nested. The major drawback of DBSCAN is the significant input required from the user. Even if *MinPts* is set globally at a specific number, it is still necessary to manually determine *Eps* for each run of DBSCAN. The algorithm can handle large amounts of data, and the order of processing does not affect the shape of the clusters. However, the time to calculate Eps is significant, and not factored into the runtime, which is still O(*n**log *n*) for the algorithm itself. In addition, the algorithm is not designed to handle higher dimensional data.

DENCLUE (DENsity based CLUstEring) [13] is a generalization of partitioning, localitybased and hierarchical or grid-based clustering approaches. The algorithm models the overall point density analytically using the sum of the influence functions of the points. Determining the density-attractors causes the clusters to be identified. DENCLUE can handle clusters of arbitrary shape using an equation based on the overall density function. The authors claim three major advantages for this method of higher-dimensional clustering.

GDBSCAN [14] algorithm is the generalized version of DBSCAN algorithm. The algorithm GDBSCAN generalizing DBSCAN in two important ways: First, one can use any notion of a neighborhood of an object if the definition of the neighborhood is based on a binary predicate which is symmetric and reflexive. For example, when clustering polygons, the neighborhood may be defined by the intersect predicate. Second, instead of simply counting the objects in the neighborhood of an object, we can use other measures, e.g. considering the non-spatial attributes such as the average income of a city, to define the "cardinality" of that neighborhood. Thus, the generalized GDBSCAN algorithm can cluster point objects as well as spatially extended objects according to both, their spatial and their nonspatial attributes. Furthermore, we present four applications using 2D points (astronomy), 3D points (biology), 5D points (earth science) and 2D polygons (geography) demonstrating the applicability of GDBSCAN to real world problems.

OPTICS [15] computes an augmented cluster ordering for automatic and interactive cluster analysis. This ordering represents a density-based clustering structure of the data. It contains information that is equivalent to density-based clustering obtained from a wide range of parameter settings. The OPTICS algorithm creates an ordering of the objects in a database, additionally storing the core-distance and suitable reachability-distance for each object. An algorithm was proposed to extract cluster based on the ordering information produced by OPTICS. Such information is sufficient for the extraction of all density-based clustering with respect to any distance Є' that is smaller than the distance Є used in generating the order. Because of the structural equivalence of the OPTICS algorithm to DBSCAN, OPTICS algorithm has the same run-time complexity as that of DBSCAN, that is O(n log n) if spatial index is used.

OPTICS algorithm is suitable to be used as a tool for cluster analysis as it can extract traditional clustering information, such as representative points and arbitrary shaped clusters. In addition, Optics can also reveal the intrinsic and hierarchical clustering structure. In contrast with the DBSCAN method, Optics provide a solution to the global density issue by giving every point object the augmented cluster-ordering containing information which is equivalent to the density-based clustering that corresponds to a broad range of parameter settings.

III. DBCSAN: DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE

The algorithm presented in tries to recognize the clusters by taking advantage of the act that within each cluster the typical density of points is considerably higher than outside of the cluster. Furthermore, the density within areas of noise is lower than the density in any of the clusters. This idea leads to a density-based approach for clustering. In this algorithm, we assume that the objects to be clustered are represented as ***points*** in the measurement space. We will formalize the notion of "density-based clusters" and "noise" in a database D of points in k-dimensional space S. Note of that, both the notion of clusters and the algorithm DBSCAN, work well in 2D or 3D Euclidean space as in some high dimensional feature space.

III.I A DENSITY BASED NOTION OF CLUSTERS

***Eps*-neighborhood of a point**: The *Eps*-neighborhood of a point p, *Eps N* (p) is a set of points having distance from p not more than Eps.
A point in a cluster is called core point if its *Eps*-neighborhood contained more number of points than the specified threshold *MinPts*. If Eps-neighborhood contained less number of points than the *MinPts* then point is a border point.
Directly density-reachable: A point p is *directly density-reachable* from a point q w.r.t. Eps and *MinPts*, if p is inside N_{Eps} (q) and the cardinality of N_{Eps} (q) is larger than *MinPts* (core point condition).
Density-reachable: A point p is density-reachable from a point q with respect to *Eps* and *MinPts*, if there is a chain of *points* $p_1, \ldots\ldots p_n$, $p_1 = p$, $p_n = q$ such that p_{i+1} is directly density- reachable from p_i .
Density-connected: A point p is density-connected to a point q w.r.t. Eps and *MinPts*, if there is a point o such that both, p and q are density-reachable from o w.r.t. *Eps* and *MinPts*.
Density-connectivity is symmetric.
Cluster: Let D be a dataset. A non-empty subset C of D is called cluster if it satisfies the following conditions:

1) Every p, q in C: if p is in C and q is density-reachable from p w.r.t. Eps and MinPts, then q is also in C. (Maximum Condition)
2) Every p, q in C: p is density-connected to q w.r.t. Eps and MinPts. (Connecting Condition)

Noise: Let C_1, C_2 . . . C_k be the clusters of the dataset D w.r.t. *Eps* and *MinPts,* $i = 1, 2 \ldots k$, then the set of points not belonging any cluster C_i is known as *noise*.

According to the notion of density-based clusters, we can get some observations.
(1) A cluster C w.r.t. *Eps* and *MinPts* contain at least *MinPts* points because it is defined as a set of density-connected points which have the maximal density-reachability.
(2) A cluster C w.r.t. *Eps* and *MinPts* is uniquely determined by any of its core points, since any point which can be density-reachable from a core point, can also be density-reachable from other core points.

III.II THE ALGORITHM

DBSCAN (Density Based Spatial Clustering of Applications with Noise) which can discover the clusters and noise in a spatial database. DBSCAN uses global values of *Eps* and *MinPts* for all the clusters. The density parameters of the thinnest cluster seem to be good approximation for these global parameter values.

The main idea of DBSCAN is a two-step approach to find one cluster. First, choose an arbitrary point from the database satisfying the core point condition. Second, retrieve all points that are density-reachable from the core point to find one cluster. Those points that dissatisfy the core condition and those points that are not density-reachable by any core points will be marked to be noise. This procedure is based on the fact that a cluster is uniquely determined by any of its core points.

```
DBSCAN(SetOfPoints, Eps, MinPts)
{
  ClusteredId := 1;
  Initially mark all points as unclassified;
   For ( i=1 ;i<  SetOfPoints.Size;i++)
   {
    Point=SetOfPoints[i];
    If(Point.ClusterId==UNCLASSIFIED)
     {
       if
(ExpandCLuster(SetOfPoints,Point,ClusterId,Eps,MinPt
s)
         {
           ClusterId ++;
          }
      }
   }
}

Boolean ExpandCluster(SetOfPoints, Point, ClusterId,
Eps, MinPts)
 {
   Nbd=Neighbourhood(SetOfPoints,Points, Eps);
   If(Nbd.Size<MinPts)
    {
      Point.ClusterId=NOISE;
      Return FALSE;
   }
  else
    {
      Point.ClusterId=ClusterId;
       For (i=1 ; i<= Nbd.Size)
        { Nbd[i].ClusterId = ClusterId;}
      Nbd.delete(Point);
      While(Nbd !=Empty)
       {
           currentP=Nbd.first();
           result=Neighbourhood((SetOfPoints,    currentP,
Eps)
           if (result.sizs >= MinPts)
            {
              for(i=1;i<=result.size;i++)
               {
                  resultP :=result[i];
                   if(resultP.cluster.Id= UNCLASSIFIED ||
resultP.cluster.Id = NOISE)
                    {
                      if(resultP.cluster.Id=
UNCLASSIFIED)
                       {
                          Nbd.append(resultP);
                        }
                      resultP.ClusterId=ClusterID;
                    }
                }
            }
       } // end while
   }
return TRUE;
}
```

For each of the n points of the database, we have at most one scan to find its Epsneighborhood. The retrieval can be successfully performed by successive region query, while region query is supported efficiently by spatial index such as R-tree [16], R*-tree [17]. Since the Epsneighborhood is in practice small compared to the whole size of data space, the average run time complexity of a single region query is O(*log n*). Thus, the average run time complexity of DBSCAN is O(*n * n*) or O (*n * log n*) depending on whether the data space is indexed or not.

DBSCAN relies on a density-based notion of clusters. It requires only one input parameter (*Eps*) and supports the user in determining an appropriate value for it. It has the advantage that it can identify clusters of arbitrary sizes. However the clusters identified may vary greatly according to the value of the input parameter. A global density value is not always sufficiently descriptive.

There are also some other open problems discussed in literature. One of them is that only point objects have been considered (for example, what if we are looking for clusters of polygons in space). Another is that this technique is not directly applicable in higher dimensions, as then an area can be dense in relation to a small number of dimensions but very sparse in relation to the whole space.

IV. GDBSCAN

This is the generalized version of DBSCAN, which can cluster point objects as well as spatially extended objects according to both, their spatial and their non-spatial attributes.

The idea of "density-based clusters" can be generalized in two important ways. First, use any notion of a neighborhood instead of an *Eps*-neighborhood if the definition of the neighborhood is based on a binary predicate which is symmetric and reflexive. Second, instead of simply counting the objects in a neighborhood of an object use other measures to define the "cardinality" of that neighborhood.

IV.I A GENERALIZED DEFINITION OF DENSITY BASED NOTION OF CLUSTERS

***Neighborhood* of an object:** Let *NPred* be a binary predicate on *D* which is reflexive and symmetric i.e., for all p, $q \in D$: $NPred(p, p)$ and, if $NPred(p, q)$ then $NPred(q, p)$. Then the *NPred*-neighborhood of an object $o \in D$ is defined as $NNPred(o) = \{o' \in D \mid NPred(o, o')\}$.

***MinWeight* of a set of objects:** Let *wCard* be a function from the power set of the Database *D* into the nonnegative Real Numbers, *wCard*: $2^D \rightarrow \mathbb{R} \geq 0$ and *MinCard* be a positive real number. Then, the predicate *MinWeight* for a set *S* of objects is defined to be true iff wCard*(S)* ≥ *MinCard.*

Directly density-reachable: An object *p* is *directly density-reachable* from an object *q* with respect to *NPred* and *MinWeight* if

1) *p* ∈*NNPred(q)* and
2) *MinWeight*(*NNPred(q))* = *true* (core object condition).

Density-reachable: An object *p* is *density-reachable* from an object *q* with respect to *NPred* and *MinWeight* if there is a chain of objects $p_1, ..., p_n$, $p_1 = q$, $p_n = p$ such that for all i=1, ..., n: p_{i+1} is directly density-reachable from p_i with respect to *NPred* and *MinWeight*.

Density-connected: An object *p* is *density-connected* to an object *q* with respect to *NPred*, *MinWeight* if there is an object *o* such that both, *p* and *q* are density-reachable from *o* with respect to *NPred*, *MinWeight*.

Density-connected set: A *density-connected set C* with respect to *NPred, MinWeight* in *D* is a non-empty subset of *D* satisfying the following conditions:

1) Maximality: For all *p, q* ∈*D*: if *p* ∈*C* and *q* is density-reachable from *p* with respect to *NPred*, *MinWeight*, then *q* ∈*C*.
2) Connectivity: For all *p, q* ∈*C*: *p* is density-connected to *q* with respect to *NPred*, *MinWeight*.

Clustering: A clustering *CL* of *D* with respect to *NPred*, *MinWeight* is a set of density-connected sets with respect to *NPred*, *MinWeight* in D, $CL = \{C_1, . . ., C_k\}$, such that for all *C* the following holds: if *C* is a density-connected set with respect to *NPred*, *MinWeight* in *D*, then $C \in CL$.

Noise: Let $CL=\{C_1 ,. . .,C_k\}$ be a clustering of the database *D* with respect to *NPred, MinWeight*. Then we define the *noise* in *D* as the set of objects in the database *D* not belonging to any density-connected set C_i , i.e. $noise_{CL} = D \setminus (C_1 \cup . . . \cup C_k)$.

IV.II THE ALGORITHM

```
GDBSCAN (SetOfObjects, NPred, MinCard, wCard)
// SetOfObjects is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfObjects.size DO
    Object := SetOfObjects.get(i);
    IF Object.ClId = UNCLASSIFIED THEN
      IF
ExpandCluster(SetOfObjects,Object,ClusterId,
NPred,MinCard,wCard) THEN
        ClusterId:=nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // GDBSCAN

ExpandCluster(SetOfObjects, Object, ClId, NPred,
MinCard, wCard): Boolean;
    IF wCard({Object}) □□0 THEN // point not in
selection

SetOfPoints.changeClId(Object,UNCLASSIFIED);
      RETURN False;
    END IF
    seeds:=SetOfObjects.neighborhood(Object,NPred);
    IF wCard(seeds) < MinCard THEN // no core point
      SetOfObjects.changeClId(Object,NOISE);
      RETURN False;
    END IF
    // Object is a core object
    SetOfObjects.changeClIds(seeds,ClId);
    seeds.delete(Object);
    WHILE seeds □□Empty DO
      currentObject := seeds.first();
      result                                    :=
SetOfObjects.neighborhood(currentObject, NPred);
      IF wCard(result) >=□MinCard THEN
        FOR i FROM 1 TO result.size DO
          P := result.get(i);
          IF wCard({P}) > 0 AND P.ClId IN
{UNCLASSIFIED, NOISE} THEN
            IF P.ClId = UNCLASSIFIED THEN
              seeds.append(P);
            END IF;
            SetOfObjects.changeClId(P,ClId);
          END IF; // wCard > 0 and UNCLASSIFIED or
NOISE
        END FOR;
      END IF; // wCard >=□MinCard
      seeds.delete(currentObject);
    END WHILE; // seeds ≠□Empty
  RETURN True;
END; // ExpandCluster
```

DBSCAN: DBSCAN is a specialization of the algorithm GDBSCAN using the following parameters: *NPred*: "distance ≤ *Eps*", *wCard*: cardinality, *MinWeight*(*N*): $|N| \geq MinPts$.

The runtime of GDBSCAN obviously is O(*n* * runtime of a neighborhood query): *n* objects are visited and exactly one neighborhood query is performed for each of them. The number of neighbourhood queries cannot be reduced since a cluster a cluster ID for each object is required. Thus, the overall runtime depends upon on the performance of the neighborhood query. Fortunately, the most interesting neighbourhood predicts are based on spatial proximity – like distance predicts or intersection which can be efficiently supported by spatial index structures.

Runtime complexity of GDBSCAN

Runtime complexity of	a single neighbourhood query	the GDBSCAN algorithm
without index	O(*n*)	$O(n^2)$
with spatial index	O(log *n*)	O(*n* * log *n*)
with direct access	O(1)	O(*n*)

GDBSCAN generalized DBSCAN in two important ways: GDBSCAN can cluster point object as well as spatially extended objects according to both, their spatial and their non-spatial attributes. GDBSCAN clustering algorithm can applied in various field viz. Earth Science (5D points), Molecular Biology (3D points), Astronomy (2D points), and Geography (2D polygons).

V. AN IMPROVED VERSION OF DBSCAN

Clustering algorithms are attractive for the task of class identification in spatial databases. In DBSCAN, if C_1 and C_2 are two clusters and very close to each other, it might happen that some point *p* belongs to both C_1 and C_2. Then *p* must be a border point in both clusters because otherwise since C_1 would be equal to and C_2 since we use global parameter. In this case, point p will be assigned to the cluster discovered first. This means the DBSCAN is sensitive to order of input. This is a drawback of the DBSCAN.

The drawback of DBSCAN can be improved if the function ExpandCluster() is modify as follows. To ensure that DBSCAN is not sensitive to order of input, we check occurrence of the above situation. If above situation occur then we find the minimum dissimilarity of the point and the two clusters and assign the point to the cluster with minimum dissimilarity. If the two dissimilarities are same than we merge both clusters; assuming that this point work here as connection point.

```
Boolean ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts)
{
  Nbd=Neighbourhood(SetOfPoints,Points, Eps);
  If(Nbd.Size<MinPts)
    {
      Point.ClusterId=NOISE;
      Return FALSE;
    }
  else
    {
      Point.ClusterId=ClusterId;
      For (i=1 ; i<= Nbd.Size)
        { Nbd[i].ClusterId = ClusterId;}
      Nbd.delete(Point);
      While(Nbd !=Empty)
       {
          currentP=Nbd.first();
          result=Neighbourhood((SetOfPoints, currentP, Eps)
          if (result.sizs >= MinPts)
           {
             for(i=1;i<=result.size;i++)
              {
                resultP :=result[i];
                if(resultP.cluster.Id= UNCLASSIFIED || resultP.cluster.Id = NOISE)
                  {
                    if(resultP.cluster.Id= UNCLASSIFIED)
                      {
                         Nbd.append(resultP);
                      }
                    resultP.ClusterId=ClusterID;
                  }
                else
                  {
                    if(reultP.ClusterId !=ClusterID)
                      {
                        d1=MinDistance (Cluster[resultP.ClusterId], resultP);
                        d2=MinDistance (Cluster[ClusterId], resultP);
                        if(d2<d1)
                          {
                             resultP.ClusterId=ClusterID;
                          }
                        if (d2=d1)
                          {
                            merge two cluster.
                          }
                      }
                  }
              }
           }
       } // end while
    }
return TRUE;
}
```

The above improvement does not affect the computational time of the DBSCAN because the above situation rarely occur. If such situation occurs then to find the dissimilarities for two clusters depends on the number of points in the clusters which is very less in comparison to the whole data set.

CONCLUSION

Spatial data mining is to mine high-level spatial information and knowledge from large spatial databases. DBSCAN and GDBSCAN are two most popular algorithms based on density notion. GDBSCAN is generalized version of DBSCAN. The proposed improved algorithm removes the shortcoming in these two algorithms.

REFERENCES

[1]. Z. Kemp and H.T.K. Lee. A Marine Environmental System for Spatiotemporal Analysis. In proc. of 8th Symp. On Spatial DATA Hnadling SDH'98, Vancour, Canada, 1998.

[2]. U.M. Fayyad and P. Smyth. Image Database Exploration: Progress and Challenges. In Proc. 1993 Knowledge Discovery in Database Workshop, Washington, DC, 1993.

[3]. R.F Cromp and W.J. Cambell. Data Mining of Multidimensional Remotely Sensed Image. In Proc. of 2nd International Conf. on Information and Knowledge Management Arlington, VA, 1993.

[4]. Zeitouni K., Chelghoum N., Spatial Decision Tree - Application to Traffic Risk Analysis, ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'01), Beirut, Lebanon, June 26-29, 2001, pp. 0203-0212.

[5]. E.C. Shek, R.R. Muntz, E. Mesrobian, and K. Ng. Scalable Exploratory Data Mining of Distributed Geoscientific Data. In Proc of the Second Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA 1996.

[6]. MacQueen J. B., Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Symposium on Math, Statistics, and Probability (pp. 281–297). Berkeley, CA: University of California Press, 1967.

[7]. Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B, 39(1):1–38.

[8]. L. Kaufman and P.J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, 1990.

[9]. R.T. Ng and J. Han. Efficient and Effective Clustering Method for Spatial Data Mining. In Proc. of 1994 Int. Conf. Very Large Data Bases, Morgan Kaufmann, San Francisco, CA, 1994.

[10]. Wang, Wei, Jiong Yang, and Richard Muntz. (1997). *STING: A Statistical Information Grid Approach to Spatial Data Mining.* Proceedings of the 23rd Very Large Databases Conference (VLDB 1997). Athens, Greece.

[11]. Agrawal, Rakesh, Johannes Gehrke, Dimitrios Gunopulos and Prahhakar Raghavan.(1998). *Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications.* Proceedings of the 1998 ACM-SIGMOD International Conference on Management of Data, Seattle, Washington, June 1998.

[12]. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density Based Algorithm for Discovering Clusters in Large Spatial Databases. In Proc. of the Second Int. Conf. on Data Mining KDD-96, Portland, Oregon, 1996.

[13]. A. Hinneburg and D. Keim, "An efficient approach to clustering Large multimedia databases with noise," in Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD‟98), 1998, pp. 58–65.

[14]. Jörg Sander, Martin Ester, Hans-Peter Kriegel, Xiaowei Xu: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications.

[15]. Mihael Ankerst, Markus M. Beruning, Hans-Peter Kriegle, Jorg Sander. OPTICS: Ordering To Identify the Clustering Structure. Proceedings of ACM SIGMOD'99 International Conference on Management of Data, Philadelphia PA, 1999.

[16]. A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA, 1984, pp. 47-57

[17]. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Point and Rectangles. In Proc. of 1990 to ACM- SIGMOD Intl. Conf. on Management of Data, Atlantic City, USA, 1990, pp. 322--331.