International Journal of Computer Applications (0975 – 8887)
Volume 79 – No 14, October 2013

DBCLUM: Density-based Clustering and Merging Algorithm

Mohammad Fawzy
Computer Science Department
Faculty of Computers and information
Fayoum University, Egypt

Amr Badr
Computer Science Department
Faculty of Computers and information
Cairo University, Egypt

Mostafa Reda
Computer Science Department
Faculty of Computers and information
Cairo University, Egypt

Ibrahim Farag
Computer Science Department
Faculty of Computers and information
Cairo University, Egypt

ABSTRACT

Clustering is a primary method for DB mining. The clustering process becomes very challenge when the data is different densities, different sizes, different shapes, or has noise and outlier. Many existing algorithms are designed to find clusters. But, these algorithms lack to discover clusters of different shapes, densities and sizes. This paper presents a new algorithm called DBCLUM which is an extension of DBSCAN to discover clusters based on density. DBSCAN can discover clusters with arbitrary shapes. But, fail to discover different-density clusters or adjacent clusters. DBCLUM is developed to overcome these problems. DBCLUM discovers clusters individually then merges them if they are density similar and joined. By this concept, DBCLUM can discover different-densities clusters and adjacent clusters. Experiments revealed that DBCLUM is able to discover adjacent clusters and different-densities clusters and DBCLUM is faster than DBSCAN with speed up ranges from 11% to 52%.

General Terms

Data mining, Clustering, Density-based

Keywords

Data mining, DBSCAN, Density-Based Clustering

1. INTRODUCTION

Clustering in data mining is a discovery process that groups a set of data based on similarity in which intra cluster similarity is maximized and inter cluster is minimized. Clustering in spatial databases is very important and used in many applications like data segmentation, discretization of continuous attributes, data reduction, outlier detection, noise filtering, pattern recognition and image processing. These discovered clusters are considered foundation for other data mining techniques such categorizing web documents, grouping of genes and proteins, grouping of spatial location to predict the mobility and so on. There are existing clustering algorithms such as K-MEANS[8], PAM (Partitioning Around Medoids)[10], CLARANS(A Clustering Algorithm based on Randomized Search)[11], DBSCAN (Density-Based Spatial Clustering of Applications with Noise)[1] and ROCK (Robust Clustering using linKs)[16]. All above clustering data based on measuring similarity among data points, but they lack clustering data of different size and different densities.

This paper presents a new density-based clustering algorithm DBCLUM, which is an extension of DBSCAN [1].In DBSCAN, the density associated with a point is obtained by counting the number of points in a region of specified radius around the point. Points with a density above a specified threshold are constructed as clusters. The DBSCAN has the ability in discovering clusters with arbitrary shape such as spherical, drawn-out, linear, and elongated, etc. Furthermore, in contrast to some clustering algorithms, it does not require the predetermination of the number of clusters.

DBSCAN cannot handle different densities or different sizes datasets. If adjacent clusters are found, DBSCAN lacks to identify clusters and noise. In addition to, DBSCAN assumes that all points within genuine clusters are density reachable and points across different clusters are not. DBCLUM is developed to resolve these issues.

DBCLUM has two main steps; Clustering and merging. The clustering is to identify all the points within a given region w.r.t. eps and minPnts for every point in the dataset. So now a huge number of clusters are found. The following step is to merge only shared and density-similar clusters. So DBCLUM can resolve the problems of DBSCAN easily because, merging process can be controlled by decreasing or increasing threshold value. If threshold value is large, it will behave like DBSCAN. in contrast, if threshold value is reduced.

The rest of paper is organized as follows. Section 2 describes Related Works and section 3 describes basic concepts of density-based clustering. Section 4 describes the DBCLUM algorithm. Section 5 describes performance evaluation. Section 6 describes conclusion and future works.

.

2. RELATED WORKS

In this section, brief descriptions of existing clustering algorithms are introduced.

K-Means[8] which has widespread use. K-means tries to assign points to clusters such that the mean square distance of points to the centroid is minimized. The centroid is a center of gravity. Major drawback in k-means is a predetermined number of clusters K. K-Medoid [9], PAM [10] and CLARANS [11]. These techniques try to find representative points called medoids such that minimize the sum of distances of points from their closest medoid. These techniques have problems with datasets of different sizes, shapes and non-globular shapes like clusters in Figure 1.

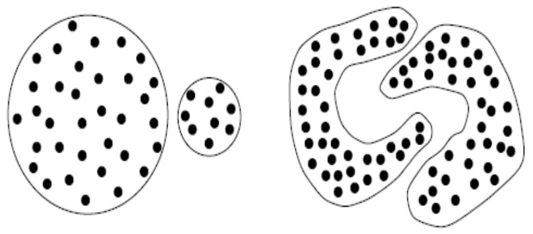

Figure 1: Datasets with different sizes and different shapes.

The DBSCAN (Density Based Spatial Clustering of Application with Noise) [1] is the basic clustering algorithm to mine the clusters based on objects density. In this algorithm, first the number of objects present within the neighbour region (Eps) is computed. If the neighbour objects count is below the given threshold value, the object will be marked as NOISE. The new cluster is formed when the number of points w.r.t eps is larger than minimum number of points MinPts. DBSCAN is a density based, i.e. it groups dense regions together.

OPTICS (Ordering Points To Identify the Clustering Structure) [4] adopts DBSCAN to handle varying densities problems. OPTICS achieves that by ordering points in a dataset provided that closest points become neighbor in the ordering. Although OPTICS actually does not produce a clustering of a data set explicitly, but instead creates an augmented ordering of the database representing its density-based clustering structure.

Incremental DBSCAN [12] algorithm is also is used for incremental updates of a clustering taking care insertion or deletion of a new object to the database.

SDBDC (Scalable Density-Based Distributed Clustering) [13] algorithm is applying DBSCAN on both local sites and global site to cluster distributed objects. In this method, DBSCAN algorithm is firstly carried out on each local site. Then, based on these local clustering results, cluster representatives are determined. Then, use local representatives on the global site to construct the distributed clustering.

DENCLUE [15] algorithm uses kernel density estimation. The result of density function gives the local density maxima value and this local density value is used to form the clusters. If the local density value is very small, the objects of clusters will be discarded as NOISE.

CURE (Clustering Using Representatives)[14], the concept of representative points is also employed in CURE to find non-globular clusters. The representative points to figure out non-globular clusters. Because, CURE finds out representative points with some criteria, it cannot handle some non-globular shapes, i.e., it finds points along the boundary, and then shrinks those points towards the center of the cluster.

MITOSIS [7] finds arbitrary shapes of arbitrary densities for high dimensional data.

2.1.1 SNN (A New Shared Nearest Neighbor Clustering Algorithm)[2] is also based on DBSCAN and it is applicable to high-dimensional data consisting of time series data.

A Fast DBSCAN (FDBSCAN) Algorithm [6] has been invented to improve the speed of the original DBSCAN algorithm. It considers only few selected representative objects belonging inside a core object's neighbor region as seed objects for the further expansion. But it suffers with the loss of result accuracy.

ST-DBSCAN [5], it is also based on DBSCAN. It handles problem of clustering spatial–temporal data according to its non-spatial, spatial and temporal attributes. It can detect some noise points when clusters of different densities exist by assigning to each cluster a density factor. Finally, it detects adjacent clusters by comparing the average value of a cluster with the new coming one.

The CHAMELEON [3] is a two phase algorithm. It generates a k-nearest graph in the first phase and then to find the clusters by combining the sub clusters. The Chameleon algorithm's key feature is that it gives importance for both interconnectivity and closeness in identifying the most similar pair of clusters. But, it suffers from low execution.

3. BASIC CONCEPTS

Definition 1: (Eps-neighborhood of a point) The Eps-neighborhood of a point p, denoted by NEps(p), is defined by NEps(p) = $\{q \in D \mid dist(p,q) \leq Eps\}$.

Definition 2: (Core point) the core point in NEps(p) is defined by Co(p) = $\{q \in NEps(p) \mid N(Eps) \geq MinPts\}$.
Therefore, for every point p in a cluster C there is a point p in C so that q is inside of the Eps-neighborhood of p and NEps(p) contains at least MinPts points.

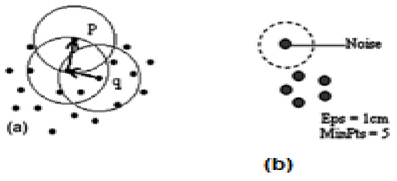

Figure 2: core points

Definition 3: (Noise point) the noise point in a cluster C is defined by N(p) = $\{q \in NEps(p) \mid N(Eps) \leq MinPts\}$.
If Eps-neighborhood is less than minimum number of points MinPts, the point is marked as noise as in Figure 2.

*Definition 4***:** (SM(C)) (x,y) is point P, SM(C)={x | x is the smallest}. It is to find the smallest x or y in all the points as in Figure 3.

Definition 5: (LR(C)) (x,y) is point P, LR(C)={x | x is the largest}. It is to find the largest x or y in all the points as in Figure 3.

Definition 6: (Area(C)) Area(C) = (LR(x) – SM(x)) * (LR(y)-SM(y)) as in Figure 3.

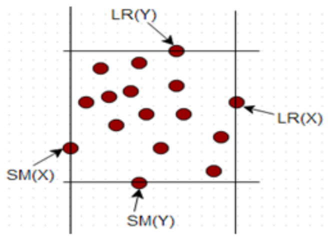

Figure 3: cluster area computation.

Definition 7: (Density Factor) C is cluster, $C\rho = \{\frac{no}{Area(C)}$ | no is the number of points in C}. density factor is used to measure the density and availability to merge with others.

Definition 8: MinMax (x,y)={ $\frac{|x-y|}{\max(x,y)}$ }.

Definition 9: The overlapped clusters in Dataset D id denoted by O(C1,C2)={C1 ∈ D, C2 ∈ D | o ∈ C1 and o∈ C2}. Use to test if any two clusters are joined and overlapped.

Definition 10: (Merge Clusters) A,B ∈ Dataset D and Aρ, Bρ . If o ∈ A and o ∈ B and MinMax(Aρ, Bρ) < threshold then A,B are merged in one cluster.

4. DBCLUM: DENSITY-BASED CLUSTERING AND MERGING

In this section, DBCLUM (Density Based Clustering and Merging) which is based on DBSCAN is presented. DBSCAN fails to identify varying densities datasets because it based on density-reachable concept. As Figure 4 shows, DBSCAN detects the two clusters as one cluster despite the varying densities and that is why, C2 is density-reachable from point O which included in C1. And then, points in C2 are density-reachable from O. Consequently, two clusters are detected as one cluster by DBSCAN. To enable DBCLUM to identify varying densities clusters, the concept of density-reachable is discarded. DBCLUM structure is different from DBSCAN structure and does not depend on density-reachable concept.

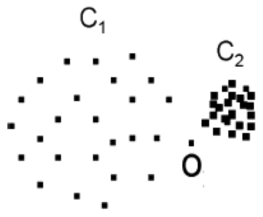

Figure 4: example of different densities clusters

Firstly, DBCLUM detects all the clusters individually, i.e. starts with an arbitrary point and get all neighbors within Eps region using Euclidean distance then travel to another point and form its cluster and so on to get all points in dataset finished.

Finally, merge overlapped clusters O(C1,C2) as in definition 9 into one cluster if MinMax(C1,C2) as in definition 8 < threshold.

By this way, the varying densities clusters problem is solved. DBSCAN lacks to identify and differentiate between adjacent clusters as Figure 5 shows.

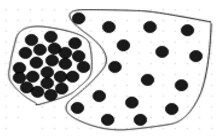

Figure 5: an example of adjacent clusters.

Because the adjacent clusters problem occur only when there is an ambiguous point or more between two clusters as in figure 5. In this case, density dictates joining it to C1 and distance dictates joining it to C2. Most of the algorithms use the distance option. DBCLUM get that point and get the all neighbors then it computes the density factor Cρ for that cluster and also C1ρ for the first cluster C2ρ for the second one. Then apply MinMax among the three factors and join it to the closest cluster.

4.1 DBCLUM Algorithm

While DBSCAN was requiring two input parameters, DBCLUM is requiring three parameters Eps,MinPts and Thresold to complete clustering process. Eps is the radius to get all neighbors within this radius, MinPts is the minimum number of points required to form a cluster In [1], a simple heuristic is used to determine the parameters Eps and MinPts. The heuristic suggests MinPts ≈ ln(n) where n is the size of the database and Eps must be picked depending on the value of MinPts. Finally, Threshold is the value on which it will determine whether merging two clusters or not.
The algorithm starts with a point p and gets all neighbors with respect to Eps, if the number of points is larger than MinPts, a cluster is formed and labeled. Otherwise, the point labeled as noise. After that, the algorithm visits another point and tries to form a cluster. This process is repeated until all points in the dataset are labeled as a cluster or as a noise. As shown in Figure 2, a random point is retrieved and retrieve the region of it w.r.t Eps. If size is less than MinPts, label this point as noise. Else label it as a cluster, set cluster id and add it to list of clusters.

```
Clusters DBCLUSTER (points,Eps,MinpPts)
  clusId := nextId(NOISE);
  FOR i from 1 to points.size DO
    Point : points.get(i);
    IF Point.rank > 1 THEN
        Continue;
    IF Point. clusId = UNCLASSIFIED THEN
        region = getRegion(points,Point,eps);
        IF region.size < MinPts Then
            Point.clusId = NOISE
        ELSE
          Clusters.add(region) ;
          clusId := nextId(clusId);
        END IF
    END IF
  END FOR
```

Now, the clusters are formed, but these clusters are overlapped and same points are found in more than one cluster. After that, clusters will be merged together based on joined clusters as in figure [6].

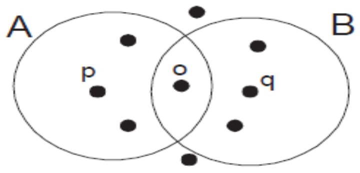

Figure 6: Clusters A,B are overlapped and joined in O point.

Merge process is applied on all clusters. After clusters are formed in the previous method, clusters are passed to DBMERGE method to run merge process. Firstly, get first cluster. If that cluster is joined and has overlap with another cluster, compute the density of the two clusters as defined in Definition 7 (Density Factor). By definition 8 (MinMax) , if densities are similar, join the two clusters. If the two clusters are not similar, clusters will be disjoined through getting overlapped points and distribute them to the cluster which is most density similar to them. To do that, overlapped point will be gotten, get its cluster and then compute the density factor to join this point to the most density-similar cluster. The remaining points in the cluster are discarded. Repeat these steps to reach that there are not joined clusters. The available clusters are the result of clustering.

```
// Merge Method
DBMERGE(clusters,thr)
FOR i FROM 1 TO clusters.size DO
    curCl := clusters.get(i);
    FOR j FROM i+1 TO clusters.size DO
      matC := clusters.get(j);
      IF Overlapped(curCl, matC) THEN
         curDen := computeDensity (curCl);
         matDen := computeDensity (matC);
         IF MinMax(curDen, matDen) < threshold THEN
           Merge(curCl, matC);
           flag := true;
         ELSE
           Overlapped := getOverlappedRegion(curCl, matC)
           FOR k FROM 1 TO Overlapped.size DO
              currPoint := Overlapped.get(k)
              curPReg := clusters.getRegion(clusters,currPoint ,eps);
                curPDen := computeDensity (curCl);
                IF MinMax(curPDen, curDen, matDen) < thr THEN
                   matC.remove(currPoint);
                ELSE
                   curCl.remove(currPoint);
                END IF
            END FOR
              IF curCl.size < MinPts THEN
                   curCl.empty();
                   clusters.remove(curCl);
              END IF
              IF matDen.size < MinPts THEN
                    matDen.empty();
                    matDen.remove(curCl);
              END IF
           END IF
         END IF
      END FOR
END FOR
```

5. PERFORMANCE EVALUATION

In this section, experimental results of DBCLUM are presented, and compare it with weka[17] available version of DBSCAN. DBCLUM was developed under weka 3-6-6 version to be available for public use and test. Although DBCLUM is applicable to any dataset, 2D dataset is chosen to make comparison and visualization much easier. Also, similar datasets are used to evaluate algorithms which will be compared with.
the experiment is based on three different datasets with different shapes and densities as shown in figure 7.First dataset, DS1, as shown in figure 7, has two different clusters in shape, density and size. The second dataset, DS2, not only has two different clusters in shape, size and density, but also, the two clusters are very close and adjacent. Because most of clustering algorithms lack to identify adjacent clusters, choosing two very adjacent clusters are cared. The third data set, DS3, has two nested clusters and noise beside the difference in shape, size and density. The size of these data sets ranges from 2,500 to 5,500 points, and their exact size is indicated in Figure 7. Note that, these data sets are generated synthetically.
By the way, all experiments are employed on PC which has windows 7, Core 2 Duo CPU @ 2.20GHz and 2 GB RAM.

5.1 Time Comparison

DBSCAN and DBCLUM are compared from time consuming view. As table 1 show, DBCLUM is faster than DBSCAN and speed up ranges from 11% to 52%.

Dataset	**DBCLUM**	**DBSCAN**	**Speed up**
DS1	1.1 seconds	2.28 seconds	52%
DS2	3.91 seconds	4.39 seconds	11%
DS3	7.16 seconds	9.09 seconds	22%

Table 1: Running time of algorithms in seconds.

5.2 Qualitative comparison

To cluster dataset using DBCLUM, three parameters (Eps,MinPts and threshold) must be entered. Eps and MinPts like DBSCAN to identify clusters and threshold to merge similar clusters in density. Figure 8, shows the clusters identified by DBCLUM algorithm and Figure 9 shows the clusters identified by DBSCAN algorithm.

From two figures, DBCLUM proved its ability to identify different shapes, sizes, densities clusters and identify noise points like DBSCAN. By comparing figure 8(DS2) and figure 9(DS2), DBSCAN fails to identify adjacent clusters while DBCLUM can identify them. If noise points are very similar to cluster points but has different density as in DS3, DBSCAN lacks to identify. DBCLUM can identify these noise points as figure 8 (DS3) shows.

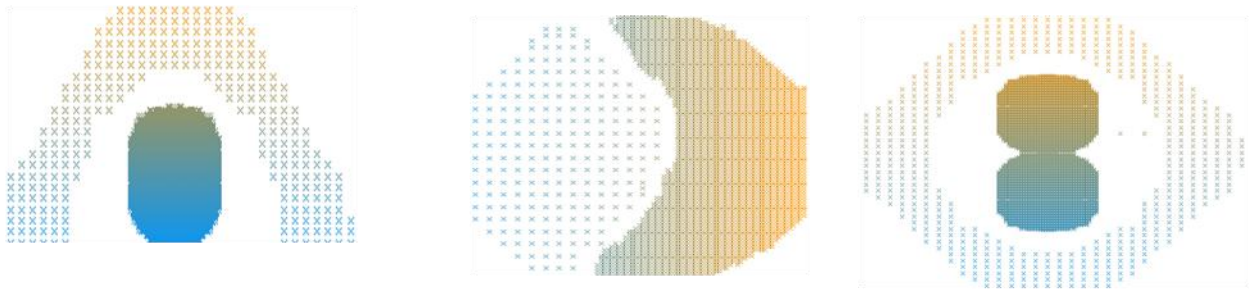
Figure 7: DS1, DS2 and DS3(5180, 2590 and 5180 points)

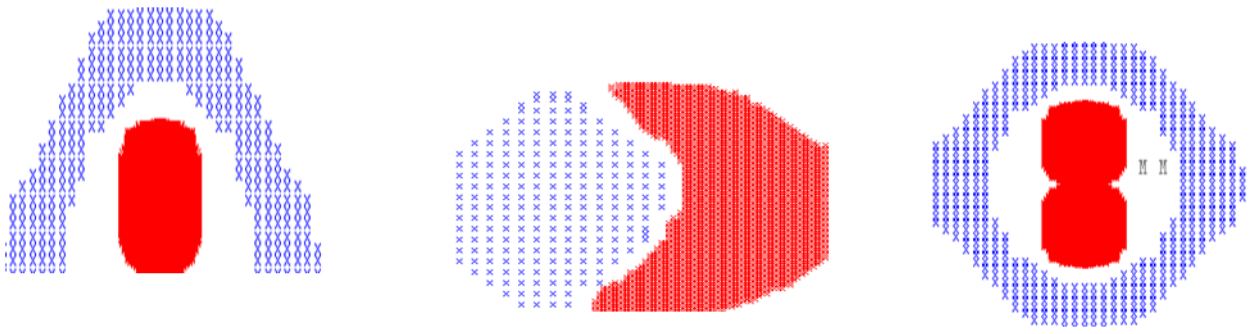
Figure 8: clusters and noises discovered by DBCLUM.

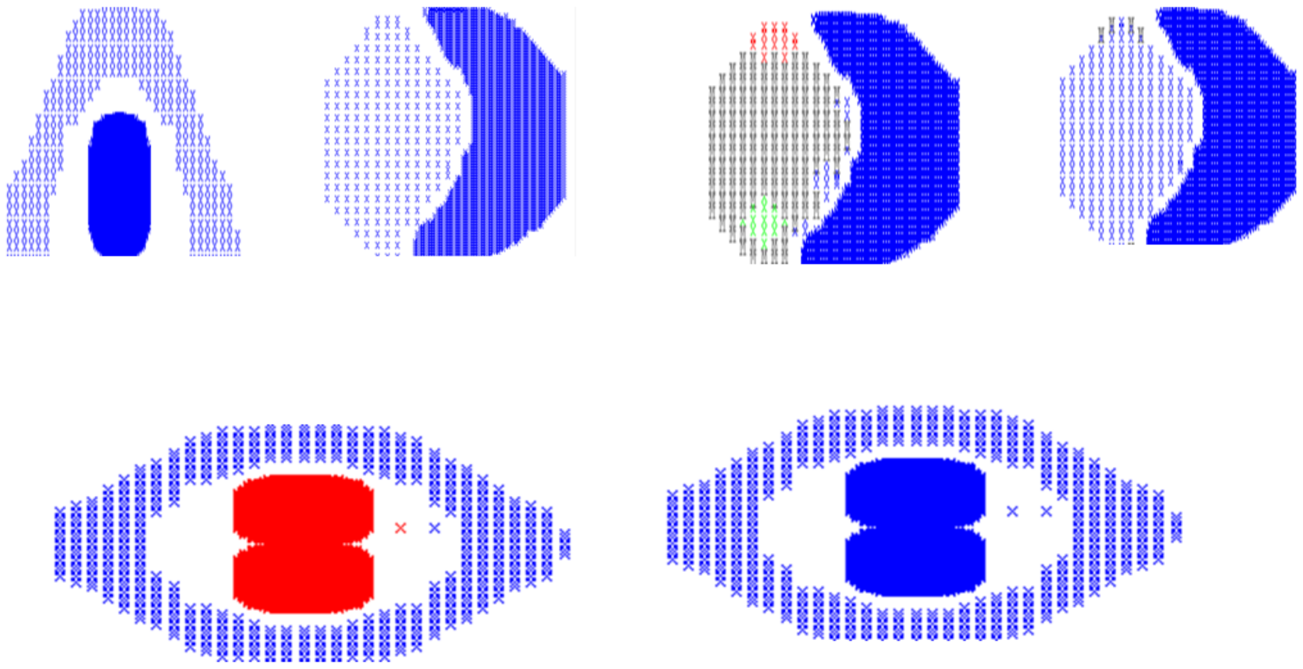
Figure 9: DBSCAN on the DS1, DS2, and DS3 data sets with different values of the Eps parameter.

6. CONCLUSION AND FUTURE WORKS

In this paper, a new clustering algorithm called DBCLUM are presented. DBCLUM can discover natural clusters of different shape, size and density. Experimental results with different datasets show the ability of DBCLUM to discover types of clusters while many existing algorithms fail to find.

Eventually the performance analysis and the output shows that the newly proposed DBCLUM algorithm gives better output, with less time and good performance.

The future research will be directed to reduce number of parameters of DBCLUM and get better performance.

7. ACKNOWLEDGEMENTS

We would like to thank M. Ester, H.P. Kriegel, J. Sander, and X. Xu for providing the DBSCAN program.

8. REFERENCES

[1] Ester M., Kriegel H.-P., Sander J., and Xu X. (1996) "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise" In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD"96), Portland: Oregon, pp. 226-231.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] P-N. Tan, M. Steinbach, V. Kumar (2005) "Introduction to Data Mining", Addison-Wesley.

[3] G. Karypis, E. H. Han, and V. Kumar (1999) "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling," Computer, vol. 32, no. 8, pp. 68–75.

[4] M. Ankerst, M. Breunig, H. P. Kriegel, and J. Sander (1999) "OPTICS: Ordering Objects to Identify the Clustering Structure, Proc. ACM SIGMOD," in International Conference on Management of Data, pp. 49–60.

[5] Derya Birant, Alp Kut (2007) "ST-DBSCAN: An Algorithm for Clustering Spatial-temporal data" Data and Knowledge Engineering pg 208-221.

[6] SHOU Shui-geng, ZHOU Ao-ying JIN Wen, FAN Ye and QIAN Wei-ning (2000) "A Fast DBSCAN Algorithm" Journal of Software: 735-744.

[7] N. A. Yousria, M.S. Kamel, and M.A. Ismail (2009) "A distance-relatedness dynamic model for clustering high dimensional data of arbitrary shapes and densities," Pattern Recognition, pp.1193-1209.

[8] J. MacQueen (1967) "Some methods for classification and analysis of multivariate observations", in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.

[9] H. Vinod (1969) "Integer programming and the theory of grouping", Journal of the American Statistical Association 64 (326) 506–519.

[10] L. Kaufman and P. Rousseeuw (1990) "Finding Groups in Data: An Introduction to Cluster Analysis": Wiley.

[11] R.T. Ng, J. Han (1994) "Efficient and effective clustering methods for spatial data mining, in: Proceedings of 20th International Conference on Very Large Data Bases", Santiago, Chile, pp. 144–155.

[12] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu (1998) "Incremental clustering for mining in a data warehousing environment", in: Proceedings of International Conference on Very Large Databases (VLDB'98), New York, USA, pp. 323–333.

[13] E. Januzaj, H.-P. Kriegel, M. Pfeifle (2004) "Scalable density-based distributed clustering", in: Proceedings of PKDD, Pisa, Italy, Lectures Notes in Computer Science, 3202, Springer, pp. 231–244.

[14] S. Guha, R. Rastogi, K. Shim (1998) "CURE: an efficient clustering algorithms for large databases", in: Proceeding ACM SIGMOD International Conference on Management of Data, Seattle, WA, pp. 73–84.

[15] A. Hinneburg, D.A. Keim (1998) "An efficient approach to clustering in large multimedia databases with noise", in: Proceedings of 4th International Conference on Knowledge Discovery and Data Mining, New York City, NY, pp. 58–65.

[16] S. Guha, R. Rastogi , K. Shim (2000) "ROCK: A robust clustering algorithm for categorical attributes," Inf. Syst., vol. 25, no. 5, pp. 345–366.

[17] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham (1999) "Weka: Practical machine learning tools and techniques with java implementations".