An Approach to Find Embedded Clusters Using Density Based Techniques

S. Roy and D.K. Bhattacharyya

Dept of Computer Science & Information Technology,
Tezpur University, Napaam 784 028, Assam, India
dkb@tezu.ernet.in, swarup@india.com

Abstract. This paper presents an efficient clustering technique which can identify any embedded and nested cluster over any variable density space. The proposed algorithm is basically an enhanced version of DBSCAN [4] and OPTICS [7]. Experimental results are reported to establish that the proposed clustering technique outperforms both DBSCAN and OPTICS in terms of complex cluster detection.

Keywords: Variable density, embedded cluster, core-distance, cluster, core neighborhood, unsupervised.

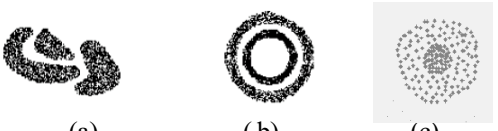
1 Introduction

Clustering is the process of grouping data into classes or *clusters* so that objects within a cluster have higher similarity, but very dissimilar to objects in other clusters [1]. From a machine learning perspective, clusters correspond to hidden patterns and the search for clusters is a *unsupervised learning*. From a practical perspective, clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, medical diagnostics, computational biology, and many others.

Cluster analysis has been considered as a difficult problem [2] because of many factors such as effective similarity measures, criterion functions, initial conditions, high dimensionality and different types of attributes, come into play in devising a well tuned clustering technique for a given clustering problem. A clustering algorithm has to be capable to identify any irregular and intrinsic cluster shapes over variable density space with outliers, as can be found in *Figure 1*.

Several good clustering algorithms have been proposed in the past decade ([1],[2]). *DBSCAN* is one of them, which can efficiently detect any clusters of arbitrary or hollow structure in presence of outliers or noise. However, a major deficiency of this algorithm is that it can not detect nested clusters over variable density space. Another major drawback of DBSCAN is that the results produced by DBSCAN are highly dependent on input parameters. Another successful successor of DBSCAN is OPTICS. It is also a density based clustering technique, which can work over variable density space successfully. However with the interactive version of OPTICS, a similar problem is encountered as we found in case of DBSCAN, it requires an additional

G. Chakraborty (Ed.): ICDCIT 2005, LNCS 3816, pp. 523 – 535, 2005.



(a) (b) (c)

Fig. 1. Irregular shaped clusters

parameters i.e. ε′. Since OPTICS provides an augmented ordering, it requires an additional cost to classify the objects. From our experiments, it has been observed that without a proper tuning of parameters it is very difficult to obtain qualitative clusters with OPTICS (Interactive). This paper presents an enhanced version of DBSCAN and OPTICS, which can detect any embedded cluster structures efficiently along with other constraints as mentioned above.

2 Related Works

Overtime, a number of clustering algorithms have been developed. Some of these are evolutionary, some are enhancements of some previously developed work and some others are revolutionary, introducing new concepts and methods. Major clustering techniques can be broadly classified into *partitional, hierarchical, density based, grid based* and *model based*. In this section, a selective review of some of the major techniques has been reported.

Partitioning methods like *k-means* [9] or *k-modes* [10] are most commonly used clustering algorithms. All the partitioning approaches have a similar clustering quality and vulnerable towards outliers. It cannot detect clusters of concave or non-globular shapes. Moreover, it requires number of clusters i.e. k as input parameter. The *Single Link* agglomerative clustering [11] is a suitable method for capturing clusters with non-globular shapes and nested structure, but this approach is very sensitive to noise and cannot handle clusters of varying density. On the other hand, it requires a post processing to achieve natural clusters. Other agglomerative clustering algorithms, e.g., *complete link* and *group average*, are capable of handling noise effectively, but sometimes they have a problem of finding globular clusters. CURE [8] is a bottom-up hierarchical clustering algorithm, which employs a method of choosing a well-formed group of points to identify the distances among clusters, instead of using a centroid-based approach or an all-points approach. In fact, CURE begins by choosing a constant number, c of *well scattered* points from a cluster. These points are used to identify the shape and size of the cluster. The next step of the algorithm shrinks the selected points towards the centroid of the cluster using some predetermined fraction. A *k-d* tree is used to store the representative points for the clusters. By definition, clusters are represented minimally, using DNF and minimal bounding rectangles. Here, emphasis is given on finding the clusters, not on the accuracy of the shapes of the clusters. CHAMELEON [5] combines a graph partitioning algorithm with a hierarchical clustering scheme that dynamically creates clusters. The first step of the algorithm partitions the data using a method based on a *k*-nearest neighbor approach

to graph partitioning. In the graph, the density of a region is stored as the weight of the connecting edge. The data is divided into a large number of small sub-clusters. The first step uses a multi-level graph partitioning algorithm. The partitioning algorithm used by CHAMELEON produces high quality partitions with a minimum number of edge cuts. The second step uses an agglomerative, or bottom-up hierarchical clustering algorithm to combine the sub-clusters and find the real clusters. CHAMELEON has been found to be very effective in clustering convex shapes, but can not handle outliers. WaveCluster[6] follows a grid-based approach. It maps the data onto a multi-dimensional grid and applies a wavelet transformation to the *feature space* instead of the objects themselves. Initially, it assigns the data to units based on their feature values. The number or size of these units affects the time required for clustering and the quality of the output. Then it identifies the dense areas in the transformed domain by searching for the connected components. If the feature space is examined from a signal processing perspective, then a group of objects in the feature space forms an *n*-dimensional signal. Rapid change in the distribution of objects, i.e., the borders of clusters, corresponds to the high frequency parts of be used to find areas of low and high frequency, and thus identifies the clusters. Wavelet transformation breaks a signal into its different frequency sub-bands, creating a representation that shows multi-resolutions, and therefore provides for efficient identification of clusters. Areas with low frequency and low amplitude are outside the clusters. With a large number of objects, signal processing techniques can be used to find areas of low and high frequency, and thus identify the clusters. WaveCluster has several significant positive contributions. It is not affected by outliers, and is not sensitive to the order of input. WaveCluster's main advantage, apart from its speedy handling of large datasets, is its ability to find clusters of arbitrary and complex shapes, including concave and nested clusters. However, one disadvantage of it is that the clustering results are highly sensitive to parameters settings. Next, we discuss two popular and efficient density based clustering algorithms, most relevant to our work, in detail.

3 Density Based Approach

The idea behind density-based approach for clustering is 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. In addition, some other definitions [5] are also associated with density based approach.

- The neighborhood within a radius ε of a given object is called the **ε–***neighborhood* of the object.
- If the **ε–***neighborhood* of an object contains at least a minimum number, *MinPts*, of objects, then the object is called a *core object*.
- Given a set of objects, *D*, we say that an object *p* is *directly density-reachable* from object *q* if *p* is within the **ε–***neighborhood* of *q* and *q* is a *core object*.

- An object p is *density-reachable* from object q with respect to **ε** and *MinPts* in a set D, if there is a chain of objects $p_1,....p_n,p_1=q$ and $p_n=p$ such that p_{i+1} is directly density reachable from p_i with respect to **ε** and *MinPts.*
- An object p is *density-connected* to object q w.r.t. **ε** and *MinPts* in a set of objects, D, if there is an object $o \in D$ such that both p and q are density-reachable from o w.r.t. **ε** and *MinPts.*
- *Density-based cluster* is a set of *density-connected* objects that is maximal with respect to *density-reachability*. Every object not contained in any cluster is considered to be a *noise*.

3.1 DBSCAN [4]

To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density-reachable from p wrt. **ε** and *MinPts*. If p is a *core point*, this procedure yields a cluster wrt. **ε** and *MinPts*. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database. DBSCAN is suitable for any large spatial domain with global density. However, in case of variable density space, DBSCAN suffers. Since it uses global parameters, i.e. **ε** and *MinPts*, DBSCAN may merge two clusters into one cluster, if the densities of those clusters are different and they are "closed" to each other. Let the *distance between two sets of points* S1 and S2 be defined as $dist\ (S1, S2) = min\ \{dist(p,q) \mid p \in S1, q \in S2\}$. Then, two sets of points having at least the density of the thinnest cluster will be separated from each other only if the distance between the two sets is larger than **ε**. Consequently, a recursive call of DBSCAN may be necessary for the detected clusters with a higher value for *MinPts*.

3.1.1 Analysis of DBSCAN

Usually, the complexity of a neighbourhood query processing is $O(n)$ and with the use of a spatial index such as a R^*-tree, it is $O(log_m n)$, where n is the size of the dataset and m is the number of entries in a page of R^*-tree. Similarly, the complexity of the DBSCAN algorithm becomes $O(nlog_m n)$ if a spatial index is used, otherwise it is $O(n^2)$. The algorithm can handle large amounts of data. DBSCAN is capable to handle noise efficiently and can identify all shapes of clusters; however, it can not identify complex cluster structures over variable density space.

3.2 OPTICS [7]

Another well known density based clustering algorithm is OPTICS (Ordering Points to Identify the Clustering Structure), which can address the issues of variable density cluster successfully. OPTICS creates an augmented ordering of the points in the database according to its densities. In addition to those common definitions used by other density based approaches, it includes the following concepts:

- The *core distance* of an object p is the smallest ε′ value that makes p a *core object*. If p is not a *core object*, the core distance of p is undefined.
- The *reachability distance* of an object q w.r.t. another object p is the greater value of the two distance measures, i.e. the *core distance* of p and the Euclidean distance between p and q. If p is not a *core object*; the *reachability distance* between p and q is undefined.

The algorithm creates an ordering of the objects in a database based on *reachability distance*, additionally storing the *core distance* and a suitable *reachability distance* for each object. Two algorithms were proposed in [7] to extract clusters interactively as well as automatically.

3.2.1 Analysis of OPTICS

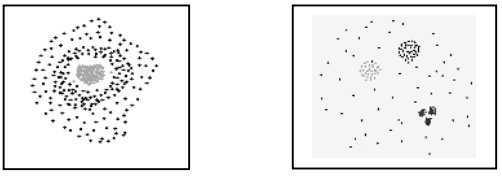
The OPTICS algorithm does not produce a clustering of a data set explicitly, but it is basically a preprocessing step for other clustering algorithms like DBSCAN. In contrast with the DBSCAN method, OPTICS provide a solution to the global density issue and varying density by giving every point object the augmented cluster-ordering containing information which is equivalent to the density-based clustering that corresponds to a broad range of parameter settings. The visualization technique proposed in [7] paper provides a good representation of the clustering structure, thus it can be used as a tool to get insight into the distribution of a data set. However, some limitations exist in this algorithm. The visualization technique of this algorithm requires proper values in the parameter settings in order to get good results. The experiments have been done to get a range of values that are considered as good values, but the usability of values may not be applicable to all types of data sets.

Our experiments reveal that interactive version of OPTICS can not detect embedded cluster structures even after several parameter settings. Apart from it, it requires $O(nlogn)$ complexity only for ordering the dense units, if a spatial index is used; further, it requires $O(n)$ time to cluster the ordered data sets. So, overall complexity will be at least $O(nlogn)+O(n)$ to extract the clusters.

4 Better Approach to Find Embedded Clusters

4.1 Motivation

Databases like gene expression databases, MR Image database and other real-data sets have the pattern of embedded or nested cluster structures. Moreover, they may have variable density. Since DBSCAN works with global density parameters, it can not detect underlying dense structure of varying density.If a low value for ε is set, it will detect several small clusters, which may not have significance in the real sense. Again, a larger value for ε may lead to ignorance of some useful clusters. So, with a single global parameter setting, DBSCAN is unable to detect the variable density clusters, as can be found in *Figure 2*.



(a) (b)

Fig. 2. Nested and Varying density clusters

On the other hand, in case of OPTICS, it is capable to detect those irregularly shaped variable density clusters, as shown in *Figure 1(a)*, *1(b)* and *2(b)*; however, it fails to detect those nested clusters, as can be found in *Figure 1(c) & 2(a)*. In case of *Figure 2(a),* with a low ε' setting, it can detect the interior two clusters with the outer region as noise and if a high value for ε' is set, it gives the similar results as DBSCAN. Moreover, it requires a prior ordering of objects in terms of *reachability distance,* which incurs additional cost. Thus, an algorithm which can detect embedded cluster structures as well as clusters of all shapes, as discussed in *Section 1* in presence of outliers is a current need.

4.2 Our Contribution

We present an integrated clustering approach, where both the density based ordering and clustering based on ordering, are integrated. Our approach can effectively address the previously mentioned clustering challenges. In addition, it can detect embedded or intrinsic clusters. It is basically an extension of those popular density based clustering algorithms, such as DBSCAN and OPTICS. It extends the concept of *core distance* of OPTICS and introduced the concept *core neighborhood* which enables to handle the problem of global density parameter setting, suffered by DBSCAN. It also handles the problems with varying density clusters as well as embedded clusters. Furthermore, like other well known density based approaches, it also gives the number of clusters naturally, in presence of noise.

4.3 Terminology Used

Here, we redefine some of the concepts used in DBSCAN in terms of our requirements. Concept of *core neighbor* is an extension of the concept of *core distance* used in OPTICS.

4.3.1 Definition: (Core Neighbor): A point p is a *core neighbor* of a point q if
1) *core-distance*(q) <> UNDEFINED, and
2) p resides within the *core distance* of q.
All the points within the *core distance* of q form the *core neighborhood* of q.

4.3.2 Definition: (Directly Core Density Reachable): A point p is *directly core density-reachable* from a point q w.r.t. *core-distance, MinPts* if
1) $p \in N_{\text{core-dist}}(q)$;
2) *core-distance*(q) <> UNDEFINED (core point condition) ; and
3) Diff(*core-distance*(p), *core-distance*(q)) $\leq \alpha$, where α is the pre-defined tolerance factor.

4.3.3 Definition: (Core Density-reachable): A point p is *Core density reachable* from a point q wrt. *core-distance* and *MinPts* if there is a chain of points $p_1, ..., p_n$, $p_1=q$, $p_n= p$ such that p_{i+1} is directly core density-reachable from p_i.

4.3.4 Definition: (Core Density Connected): A point p is *Core density connected* to a point q wrt. *core-distance* and *MinPts* if there is a point o such that both, p and q are core density-reachable from o.

4.3.5 Definition: (Cluster and Noise): Let *D* be a database of points. A cluster *C* wrt. *core-distance* and *MinPts* is a non-empty subset of *D* satisfying the following conditions:

1) $\forall p, q$: if $p \in C$ and q is density-reachable from p wrt. *core-distance* and *MinPts*, and diff(*core-distance*(p), *core-distance*(q))$\leq \alpha$ then $q \in C$, where α is the pre-defined tolarance factor.
2) $\forall p, q \in C$: p is *Core density-connected* to q wrt. *core-distance* and *MinPts*.

An object is *noise* if its *core distance* is greater than global parameter ε.

4.4 Finding Clusters

Intuitively, all the *core neighbors* of a point having *core distance* difference within α, form a uniform dense region. In the *Figure 3* the point *P* is a core object w.r.t. *MinPts* = 3 and *core distance* of *P* is ε′. The ε′ must be less than equal to ε (the user defined radius). The points within the *core distance* ε′ are the core neighbors of *P*. OPTICS use that *core distance* and *reachability distance* to order the points. On the other hand DBSCAN expands clusters by expanding the points within ε-neighbour -hood. From our observation we find that core distance is very much effective in detecting density variations. Variation in core distance implies a variation in density. Unlike OPTICS additional ordering is not essential to detect clusters. Cluster can easily be extracted same way as by DBSCAN, with a difference of expanding the ε′-neighbors instead of ε-neighbors. Our approach integrates these two approaches. It expands the *core neighbor* of a *core object* say *P* instead of expanding ε-neighbors. Iteratively the point *Q* is also expanded same way. If the core distance of *P* and *Q* are within a tolerance factor α then both of them are considered as belonging to the same cluster. If *core distance* of a point is greater than ε then it is a noise point. During expansion of *Q*, *P* becomes the *core neighbor* of *Q*. But if *P* is processed earlier than *Q* and it is already assigned a *cluster id*, then *P* is ignored.

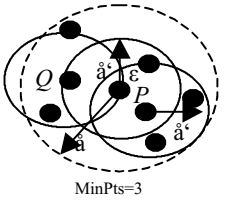


Fig. 3. Cluster expansion

5 The Algorithm

The algorithm proceeds as DBSCAN and OPTICS by expanding each *core-object* to get cluster structure. It continues to scan the datasets until all the objects are not processed. Each core object begins to expand all its neighbors of it, with respect to generating distance i.e. *core distance (ε′)*.

If the core distances of two objects do not differ by a pre-defined variance factor, say α, we consider them belonging to the same cluster or their density is same.

The main module of the algorithm is illustrated in *Figure 4*. It starts with an initial *core-distance* of an arbitrary object from the data sets. *GetCoreDist* computes the core distance of a unclassified object with respect to *MinPts* and ε. If the *core distance* isundefined i.e. if *core distance* is greater then ε, then the object is marked as noise. Otherwise, it will go for expanding the cluster with its neighbor objects within its *core neighbourhood*. Assign a new *cluster id* to the candidate object and mark all the neighbors of it with the same *id*, if it is not already assigned an *id*. Next, in an iterative

```
EnDBSCAN (SetOfPoints, ε, MinPts) // SetOfPoints is
UNCLASSIFIED

FOR i FROM 1 TO NoOfObjects DO
        Point: = SetOfPoints.get (i);
IF Point already not UNCLASSIFIED THEN

        CORE_DIST:=GetCoreDist (Point, MinPts, ε);

        IF CORE_DIST=UNDEFINED
                Mark Point as Noise;
        ELSE

        Expand Cluster (Point, CORE_DIST);
        END IF

END IF
END FOR
END; // EnDBSCAN
```

Fig. 4. Module EnDBSCAN

manner it expands for each of the objects in the neighborhood. *Figure 5* illustrates the sub module *Expand Cluster*. We consider the *core distance* of the starting object of a new cluster as the initial *core distance*; which is termed here as *previous core distance*. Two objects are in the same cluster if the difference between *previous core distance* and current candidate object's *core distance* is not more than a factor α. Otherwise, the candidate is considered to belonging to a different cluster and ignored that objects i.e. it will not expand that object. The underlying idea behind is that such a situation generally indicates a density variation, and the current candidate object is considered as belonging to a different cluster. Such a decision making may lead to some amount of repetition works on object processing. However, based on observation, it has been found that such a situation usually occurs only in the boundary of two different dense regions and the number of objects to be processed in repetition is also negligibly small. Thus, it can be easily handled by any trivial memory based technique (by storing *core-distance* and *core neighbor* of the rejected object).

5.2 Complexity Analysis

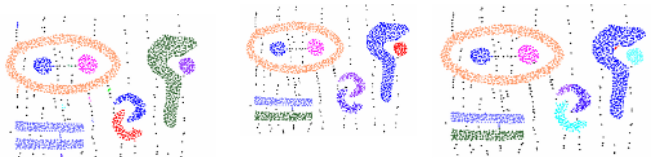
Because of the structural equivalence of the proposed EnDBSCAN to both DBSCAN and OPTICS, it has the same run-time complexity as that of DBSCAN and OPTICS that is, $O(nlogn)$, if a spatial index like R^* tree is used. However, EnDBSCAN requires to carry out some amount of repetition work in the boundary of two dense regions, but the number of points to be processed repeatedly is significantly very less when compared with the total number of points, so it can be neglected.

```
Expand Cluster (Point, Prev_Core)

IF Point already CLASSIFIED
      RETURN;
END IF

CORE_DIST:=GetCoreDist (Point, MinPts, ε);

IF CORE_DIST= =UNDEFINED THEN
   Mark Point as Noise;
    RETURN;
END IF

IF diff(CORE_DIST-Prev_Core) > α
   RETURN;
END IF

Mark the Point as CLASSIFIED;
Neighb:=GetNeighbour(Point,CORE_DIST);


IF Point not assigned ClusterId THEN
    Assign the Point with nextId ();
END IF

   Mark all the objects of Neighb (core neighborhood),which
are not already classified ,with Point.ClusterId.

FOR each NewPoint in Neighb DO
        Expand Cluster (NewPoint, Prev_Core);
END FOR

    END;// End Expand Cluster
```

Fig. 5. Cluster Expansion Module

6 Experiments

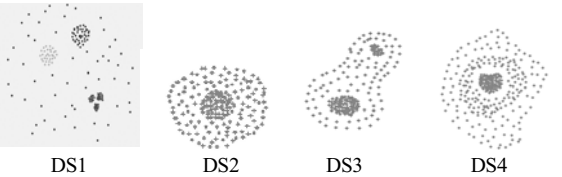
To carry out an experimental study on the proposed algorithm and to study its performances with its other counterparts, we developed a Java based user interface for easy synthetic data set generation as well as for visualizing the test results. We used a PIV Server with 128 MB RAM and the language used for coding is *Java 1.3* in *Windows Xp*. We used five sets of datasets, i.e. the *CHAMELEON* t7.10k.dat [5] dataset and four other synthetic data sets, as shown in *Figure 6* & *7* respectively. In case of *t7.10k.dat dataset*, it has been observed that all the three algorithms identify the desired clusters correctly. However, this dataset does not contain any nested cluster structure.



DBSCAN for *MinPts*=3 & ε=7 | *OPTICS MinPts* =7 & ε=12 ε′=8 | *EnDBSCAN* for *MinPts* =9 ε=13

Fig. 6. Results from *t7.10K.dat*

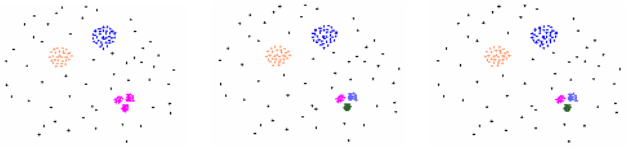
Next we tested the algorithm in light of synthetic datasets (*Figure 7*) and compared the results. It has been observed that our approach outperforms DBSCAN and OPTICS (interactive) in terms of nested cluster identifications. EnDBSCAN has been able to detect variable density clusters as well as nested or embedded cluster structures successfully, whereas the other two counterparts fail, even after multiple parameter settings.



DS1 DS2 DS3 DS4

Fig. 7. Synthetic Data

In case of test dataset *DS1*, both OPTICS (interactive) and EnDBSCAN are found successful (Figure 8) in detecting five natural clusters, where as DBSCAN fails to do so.



DBSCAN for (*MinPt*=5, ε=15) | *OPTICS* (Int.) for (*MinPt*=5, ε=10, ε'=5) | *EnDBSCAN* for (*MinPt*=5, ε=15)

Fig. 8. Results from DS1

In case of *DS2*, DBSCAN can only detect a single cluster. In case of OPTICS, for a smaller value of ε′, it can only detect the interior cluster pattern and rest as noise; otherwise it works same as DBSCAN. On the other hand, EnDBSCAN can successfully detect both the natural clusters.

In case of *DS3* (*Figure 10*), both DBSCAN and OPTICS fail to give the proper results. OPTICS gives two interior clusters and rest as noise. On the other hand, EnDBSCAN can detect all the three natural clusters. However, due to the order

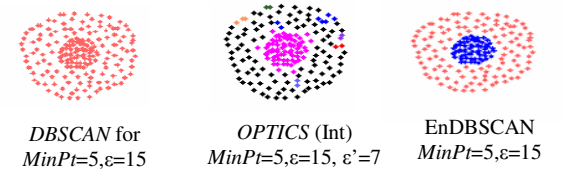


Fig. 9. Results from DS2

dependency nature of DBSCAN, it also results in overlapping of a boundary point between two different dense regions. In such case, generally the boundary point is assigned to that cluster which is scanned first. In case of *DS4* also, similar results found not included due to space limitation).

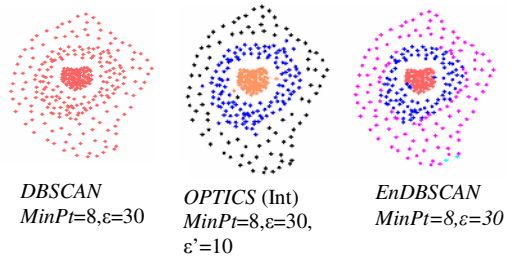


Fig. 10. Results from DS3

Based on our exhaustive experimental study it has been observed that for a tolerance factor i.e. α=2, the clustering results of the proposed algorithm can be found to be more effective. So, rather than considering it as an input parameter, we prefer to consider it as a constant. However, α may need to be tuned based on the distribution of data for different datasets. We reported execution time needed by EnDBSCAN in comparison with other counterparts, in the following figure. We implemented these algorithms without using any spatial indexing techniques. We generate data in such a way that density of data increases with the increase in size of the data.

Data Size	ε	ε'	MinPts	DBSCAN	OPTICS	EnDBSCAN
5000	8	6	3	7	10	15
8000	8	6	6	23	31	38
10000	8	6	7	35	48	56
15000	8	6	15	143	128	132
20000	8	6	20	271	202	226
25000	8	6	22	562	390	408
30000	8	6	25	946	609	654

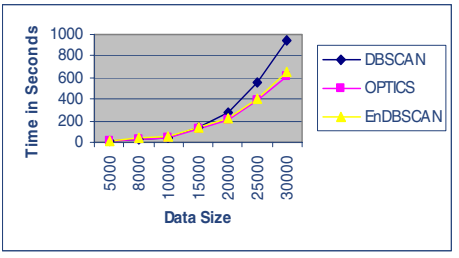


Fig. 11. Scalability Curve

From the graph it can be seen that when the data are sparse DBSCAN performs better than other two. But the scenario reversed when data become dense. In such case our's performs well over DBSCAN. However, from execution time point of view, performance of OPTICS is superior in comparison to EnDBSCAN, though OPTICS can not detect embedded cluster structure.

6.1 Clustering Effectiveness Comparison

A detailed comparative study among the three algorithms (i.e. EnDBSCAN, DBSCAN & OPTICS (Int.) was carried out in light of those real and synthetic datasets (as discussed in the previous *sub-section*). *Table 1* presents the same in terms of six crucial factors. As can be seen from the *column 1* of the table that like DBSCAN, the proposed algorithm also requires less number of input parameters than OPTICS. Similarly, *column 5* depicts that embedded clusters can be detected only by the proposed algorithm. Also, from the complexity point of view, *column 6* clearly shows that the performance of *EnDBSCAN* is similar with *DBSCAN* when a spatial index is used. However, OPTICS requires an additional complexity *O(n)* (at least) to classify those points after ordering, apart from *O(nlogn)*, when a spatial index is used. The rest other columns establish that in terms of the other quality parameters, the performance of the proposed algorithm is equally good with its other two counterparts.

Table 1. Comparison of EnDBSCAN with DBSCAN and OPTICS (Int)

Algorithms	**Input Parameters (1)**	**Outlier. Handling (2)**	**Scalability (3)**	**Varying Density (4)**	**Embed. Cluster (5)**	**Complexity (6)**
DBSCAN	*MinPts*, å	Yes	Yes	No	No	*O*(*n log n*)
OPTICS(Int)	*MinPts*, å , å'	Yes	Yes	Yes	No	*O*(*n log n*) + *O* (*n*)
EnDBSCAN	*MinPts*, å	Yes	Yes	Yes	Yes	*O*(*n log n*)

7 Conclusions

This paper presents an enhanced version of DBSCAN and OPTICS (Int.). The proposed enhanced version can detect any embedded cluster structure over spatial domain successfully. Another significant advantage of EnDBSCAN is that it requires less input parameters as well as less complexity than OPTICS.

References

[1] Han & Kamber, *Data Mining: Concepts & Technques*, Morgan Kaufmann, 2001.

[2] Kotsiantis & Pintelas, *Recent Advances in clustering: A Brief Survey,* www.math.upatras.gr/~esdlab/en/members/ kotsiantis

[3] Jiang, Tang & Zhang, *Cluster Analysis for Gene Expression Data: A Survey.* IEEE Trans. KDE, 2004.

[4] Ester, Kriegel, Sander and Xu. 1996, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise* in KDD96, Portland, pp 226-231.

[5] Karypis, Han & Kumar, *CHAMELEON: A hierarchical clustering algorithm using dynamic modeling.* IEEE Computer, 32(8), pp 68-75, 1999

[6] Sheikholeslami, Chatterjee and Zhang. *Wavecluster:A muti-resolution clustering approach for very large spatial database* in the *SIGMOD'98* Seattle, 1998.

[7] Ankerst, Breuing, Kriegel and Sander. *OPTICS: Ordering points to identify the clustering structure* in the *ACM-SIGMOD'99,* pp 49-60, 1999.

[8] Guha, Rastogi, and Shim, *'CURE: An Efficient Clustering Algorithm for Large Datasets* in the ACM SIGMOD Conf., 1998.

[9] McQueen, 'Some Methods for Classifications and Analysis of Multivariate Observations', in the Sympos. on Math, Statis. and Probabilty', pp 281-197, 1967

[10] Z Huang, 'A Fast Clustering Algorithm to cluster very large categorical datasets in Data Mining', SIGMOD'97.

[11] Kaufman and Rousseeuw.Finding Groups in Data: An Introduction to Cluster Analysis. New York: John Wiley & Sons, 1990.