An Extended DBSCAN Clustering Algorithm

Ahmed Fahim
Department of Computer Science, Prince Sattam Bin Abdulaziz University, Aflaj, Saudi Arabia
Department of Computer Science, Faculty of Computers and Information, Suez University, Suez, Egypt

***Abstract*—Finding clusters of different densities is a challenging task. DBSCAN "Density-Based Spatial Clustering of Applications with Noise" method has trouble discovering clusters of various densities since it uses a fixed radius. This article proposes an extended DBSCAN for finding clusters of different densities. The proposed method uses a dynamic radius and assigns a regional density value for each object, then counts the objects of similar density within the radius. If the neighborhood size ≥ MinPts, then the object is a core, and a cluster can grow from it, otherwise, the object is assigned noise temporarily. Two objects are similar in local density if their similarity ≥ threshold. The proposed method can discover clusters of any density from the data effectively. The method requires three parameters; MinPts, Eps (distance to the k^{th} neighbor), and similarity threshold. The practical results show the superior ability of the suggested method to detect clusters of different densities even with no discernible separations between them.**

Keywords*—Cluster analysis; density-based clustering; varied density clusters; data mining; extended density-based spatial clustering of applications with noise (E-DBSCAN)

I. Introduction

Cluster analysis is used for knowledge discovery rather than prediction. It aims to discover the groups of similar data in a given dataset. In other words, the objective of clustering is to divide a set of objects into some subsets such that similar objects are collected together in a subset and dissimilar objects are assigned to different subsets. Every subset is known as a cluster. Clustering is an unsupervised machine learning task because it requires neither a training set nor known labels for the discovered clusters. The resulting clusters depend on what the cluster is. A fine clustering algorithm should have the ability to discover clusters of different forms, sizes, densities, and handle noise in data. It should not be sensitive to the ordering of input data, and the parameters given by the user. In addition, the number of parameters should be as very small as possible. Most of the mentioned factors may be satisfied in a density-based method called DBSCAN "Density-Based Spatial Clustering of Applications with Noise" [1]. It is considered the leader in discovering clusters based on regions' density in data space. It defines the density of an object as the count of its neighbors in a given radius. But this idea does not apply to a dataset having different densities. So many modified versions of it and new methods have been proposed to overcome this drawback. OPTICS [2] algorithm is an extension for DBSCAN and doesn't deliver clusters explicitly. It estimates an arrangement of points in a dataset based on the core distance and reachability distance. DENCLUE [3] is a density-based approach that uses influence functions (maybe parabolic functions, square wave function, or the Gaussian function). It applies the influence function on each object in the dataset and specifies the density attractors that are the regional maxima of the overall density function. It fails to find clusters of various densities because it employs two input parameters σ and ζ which are equivalent to Eps and Minpts in DBSCAN.

This paper introduces an extended DBSCAN algorithm, wherein in this version the neighborhood radius (Eps) is not fixed for all objects as in the basic version. Eps will be equal to the distance to the k^{th} neighbor. Thus, the Eps will vary from one object to another. MinPts parameter controls the regional density of objects; where the regional density of an object is equal to the sum of distances to its MinPts-nearest neighbors. For each object, the algorithm counts the objects in its k-nearest neighbors that have a similar regional density to it. Certainly, this number will be less than or equal to the value of k-nearest neighbors. If the object has more than or equal to MinPts similar objects then it is a core object and the cluster can grow from it otherwise the object is noise temporarily. So, the algorithm requires another parameter to judge the similarity of local densities among objects. The practical results show the superior ability of the suggested method in handling various density clusters.

The main contribution of this research is that it presents a way to overcome the main problem of the DBSCAN. It allows the Eps to vary from one object to another. It redefines the density of an item as the number of similar items within its neighborhood, in addition to the sum of the distances to the MinPts-nearest neighbors. This technique made the DBSCAN able to detect clusters of different densities.

The article is arranged as follows; subsection *A* presents the DBDCAN algorithm and its main problem. Section II demonstrates several related techniques. The suggested method is shown in Section III. Section IV illustrates the efficacy of the suggested technique and explains the outcomes. Finally, the concluding section brings the paper to a close.

A. DBSCAN Algorithm

The DBSCAN technique is the head technique in the density-based class. It counts the objects in a fixed neighborhood radius (Eps) of the current object. If the count of objects in this radius is larger than or equal to a threshold (MinPts), then this object is a dense (core) object, and the cluster can be grown from it otherwise, the object is considered temporarily as a noise object. So, the objects in the dataset are classified into cores, borders, or noises, as shown in Fig. 1. A border object is not a core; it belongs to a core object's neighborhood. A noise object is neither a core nor a border object.

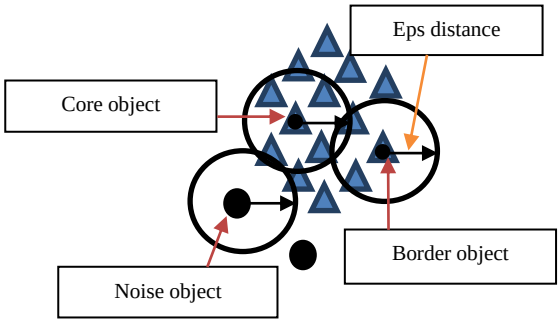


Fig. 1. Concept of the Core, Border and Noise Object where Minpts = 3.

The following definitions are used by DBSCAN [1]:

Definition 1: Eps-neighborhood of an object *x* is denoted by $N_{Eps}(x) = \{y \mid dis(x,y) \leq Eps\}$, where dis(*x*,*y*) is the Euclidean distance.

Definition 2: Directly density-reachable, an object *x* is directly density-reachable from an object *y* wrt. Eps, and MinPts if
1) x ∈
is part of the neighborhood of y.
2) y is a core object (|neighborhood of y| ≥ MinPts).

Definition 3: Density-reachable, an object *x* is density-reachable from an object *y* wrt. Eps and MinPts if there is a chain of objects where each object is directly density-reachable from the previous one in the chain.

Definition 4: Density-connected, two objects are density-connected wrt. Eps and MinPts, if there is an object *z* such that both objects are density-reachable from z.

Definition 5: A cluster C is a non-empty subset of objects satisfying:
1) ∀ *x*, *y*: if *x* ∈ C and *y* is density-reachable from *x*, then *y*∈ C.
2) ∀ *x*, *y* ∈ C: *x* is density-connected to *y*.

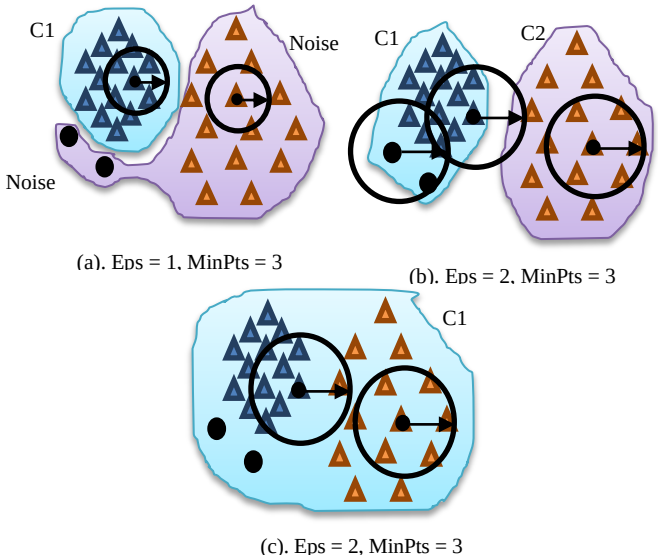
Definition 6: Noise is the collection of objects that are not assigned to any cluster.

DBSCAN works as follows:

```
DBSCAN (dataset D, Eps, MinPts)
// all points in the dataset are Unclassified
ClusId = 0   // -1 Unclassified, 0 Noise
For i = 1 to D.size
   Point = D.get(i)
   IF Point.ClusId <> -1
      Neighbors = D.regionQuery(Point, Eps)
      If Neighbors.size ≥ MinPts
         ClusId = ClusId + 1
         Point.ClusId = ClusId
         ExpandCluster(D, Neighbors, ClusId, Eps, MinPts)
      Else
          Point.Clus_Id = 0 // point is noise temporarily
      End If
    End If
End For
End // DBSCAN
```

```
ExpandCluster(D, Neighbors, ClusId, Eps, MinPts)
For Each point in Neighbors
   Point.ClusId = ClusId
End For
While Neighbors.size < > 0
    CurrPoint = Neighbors.first();
    Res =D.regionQuery(CurrPoint, Eps)
    If Res.size ≥ MinPts
      For i = 1 to Res.size
          ResPoint = Res.get(i);
          If ResPoint.ClusId = -1
               Resoint.ClusId = ClusId
               Neighbors.append(ResPoint)
          Else If ResPoint.ClusId = 0 // point is noise
              ResPoint.ClusId = ClusId // point is a border point
          End If
      End For
    End If
      Neighbors.delete(CurrPoint)
End While
End // ExpandCluster
```

The Expand_cluster function collects the density-connected core objects and the border objects that are connected directly to any core in the cluster. This approach discovers clusters of varied shapes and sizes efficiently, but it has trouble detecting bunches of different densities because it uses global values for its parameters; the Eps, and the MinPts threshold that represents the minimum density for any core object.



(a). Eps = 1, MinPts = 3

(b). Eps = 2, MinPts = 3

(c). Eps = 2, MinPts = 3

Fig. 2. Eps Value is not Suitable for all Clusters in Data.

Fig. 2 explains the problem of using a single value for the Eps parameter in DBSCAN. In Fig. 2(a), the assigned value for the Eps is small. So, the DBSCAN discovers the dense cluster C1 that contains all the blue triangles, and classifies the other objects as noises -orange triangles and the two black circles-. In Fig. 2(b), the value of Eps is increased, and the clusters are well separated, so the DBSCAN discovers the clusters C1, C2 but it merges the noise objects (the two black circles) with the dense cluster C1. That means its ability to handle noise

decreased. In Fig. 2(c), the value of Eps is the same as in Fig. 2(b), but the clusters are closer to each other. So, the DBSCAN places all the objects in one cluster. From Fig. 2, we see that a global value for the Eps is unsuitable at all for a dataset that contains clusters of different densities.

II. Related Methods

Because clustering techniques play a significant role in data mining and knowledge discovery, they are used in various applications like image processing, search engine, bioinformatics, pattern recognition, market research, social network analysis, and so many others. The clustering strategies may be categorized into four classes: partitioning, hierarchical, density-based, and grid-based methods.

A. Partitioning Methods

Algorithms that are belonging to partitioning class such as k-means [4], PAM "Partitioning Around Medoids" [5], CLARA "Clustering LARge Applications" [5], and CLARANS "Clustering Large Applications based on RANdomized Search" [6], [7] describe a cluster as a collection of objects with the least variance from the mean or the medoid of the cluster; where the mean is the center of the objects in the cluster and A medoid is a cluster representative object with the least amount of dissimilarity to the other objects in the cluster. Convex-shaped clusters are favorable by this definition. If the data contain clusters of different shapes, then; this definition is not suitable, and the clustering technique does not find the actual clusters; it may divide some clusters or merge some of them. These methods are unable to discover overlapped clusters or clusters of different shapes. Also, these strategies necessitate knowing the number of clusters in advance. The DBSCAN method was used to solve this problem, as well as a suitable selection for the initial centers [8]. The k-means method works as follows: -

```
k-means(dataset D, k)
    For i =1 to k
        x = 1+ rand()% (D.size -1)
        means[i] = D[x]    //select the initial means randomly
    End For
    Assign_object_to_nearest_cluster_and_update_means()
                                                //given below
    do
      oldmse = mse
      Assign_object_to_nearest_cluster_and_update_means()
    While mse < oldmse
Return means
//-----------------------------------------------------------------
Assign_object_to_nearest_cluster_and_update_means()
   For i =1 to k
     npc[i] = 0 // npc refers to number of points in each cluster
     oldmeans[i] = means[i]
     means[i] = 0
   End For
   mse = 0
   For i = 1 to D.size
       dis[j] = 0      min=1
       For j = 1 to k
          dis[j] = dis(oldmeans[j], D[i])
          If dis[j] < dis[min]
             min = j
          End If
       End For
     clusid[i] = min      // assign point to the closest cluster
     mse = mse + dis[min]    // update mean square error
     means[min] = means[min]+D[i]
     npc[min] = npc[min]+1  //update the number of points
                                                in the cluster
   End For
   For j= 1 to k
      means[j]= means[j]/ npc[j]  // update the means
   End For
End // Assign_object_to_nearest_cluster_and_update_means
```

B. Hierarchical Methods

In the second category, hierarchical techniques create a hierarchical dendrogram like a tree structure. These techniques are classified into agglomerative and divisive methods. Agglomerative methods build the dendrogram from the bottom-up, while divisive methods build it from the top-down. The agglomerative methods are more familiar than the divisive methods. A hierarchical method starts with initial clusters that may be singleton clusters. In each step, it picks to combine two clusters based on a metric measure as in the single link method [9], complete link [10], and the average link method. There are several techniques in the hierarchical class such as CURE "Clustering Using Representatives" [11] that selects representatives for each cluster and uses these representatives in cluster calculations. BIRCH "Balanced iterative reducing and clustering using hierarchies" [12] is another model of hierarchical methods, which introduced the idea of the cluster feature tree. Where each cluster feature holds the count of objects in the cluster, their linear sum, and their square sum. These cluster features are arranged in a height-balanced cluster feature tree. But it uses the centroid of a cluster as a representative and redistributes the objects to the closest seed. This means that it prefers convex-shaped clusters.

C. Density-based Methods

This category introduces another description of what a cluster is. This description depends on a density concept, where a cluster is a collection of density-connected objects. A dense object is known as a core object which has not less than a specified number of objects within its neighborhood of a specified radius. This term was introduced in the DBSCAN method [1]. It is the main founder for the density-based methods to find clusters of different forms and sizes. It does not demand to know the number of clusters in advance. Also, it can treat noise objects. Unfortunately, this method fails to detect clusters of different densities since it uses a fixed value for its parameters; Eps and MinPts. This limitation motivated many researchers to propose ideas to overcome this problem. So, it has attracted the interest of researchers from all over the world.

In [13], The author presented a concept that allows Eps to change from one cluster to the next while maintaining control over the density of each core object within the cluster. The method needs two input parameters; MinPts and MaxPts. The

minimum density for core objects in the cluster is controlled by MinPts, and the maximum density is controlled by MaxPts. The method arranges the objects in the dataset based on the distance to the MaxPts neighbor, and it starts to create a cluster from the core that has the minimum distance to the MaxPts neighbor. When there is a very tiny difference in density within the cluster, this approach yields good quality clusters. Because this method focuses on homogenous clusters, if the variance in density inside the cluster grows, the method will divide the cluster.

In K-DBSCAN [14], the authors developed a two-step algorithm. To begin, the method computes each object's density as the average distances to its k-nearest neighbors and sorts the objects based on their densities; from the density curve, they can see how many levels (k) of density are present in the dataset; and finally, the method divides the dataset into k different levels of density using the k-means. Second, each density level is subjected to a modified version of DBSCAN. The number of density levels derived from the curve determines the final output. The average distance to the k-nearest neighbors is similar to the k-dist plot, but the algorithm may see incorrect levels of density, causing some clusters to split. Using the k-means to partition the data into different levels of density leads to a problem because they consider each level of density as a new dataset and apply a modified version of DBSCAN on it. Surely, the objects in the same level do not have the same density there will be variance in the density of objects and their modified version of DBSCAN does not consider this. If the dataset has a single density level this means k=1 in k-means and all objects are assigned the same level of density, the modified version of DBSCAN that is used in this method may assign all objects the same cluster unless clusters are well separated. Since the method assigns an object and all its k-nearest neighbors -that are in the same level of density- the same cluster.

In [15], the author presented a new paradigm to deal with different density clusters. The approach gives each object a local density value equal to the sum of its k1-nearest neighbors' distances. Then, it divides objects into attractors and attracted objects by counting the objects in the object's k-nearest neighbors that are denser. The clusters evolve from denser to sparser objects. This approach requires four input parameters, which is a significant number. Furthermore, the fine-tuning setting is a difficult operation.

In [16], to enlarge the cluster, the authors recommended using a mutual k-nearest neighborhood to establish k-deviation density of points and a deviation factor to locate direct density reachable neighbors for core points. This procedure yields inaccurate results. The method tends to split the clusters since it allows a very small density deviation within a cluster. The problem in this method comes from the computational way of the *k*-deviation density.

GMDBSCAN [17] is a grid-based technique. It is based on a spatial index (sp-tree). It allows the Minpts to vary from one cell to the next based on the density of each grid cell, uses the same Eps value in each grid cell, and runs DBSCAN on the data in each grid cell. This approach employs several parameters that have an impact on the clusters that produce. The problem with GMDBSCAN is that it takes a long time to run on large datasets.

GMDBSCAN-UR [18] is a GMDBSCAN algorithm adaptation. It selects as representative objects some well-scattered points that form the shape and extent of the dataset in each grid cell. GMDBSCAN's time-consuming difficulty is solved using this solution. It permits one DBSCAN parameter (Minpts or Eps) to vary from one cell to the next, and it performs better than GMDBSCAN. With varying density clusters, however, it does not yield reliable results. VDBSCAN [19] separates the k-dist plot based on the curve's sharp change. It chooses an appropriate Eps value for each partition and runs DBSCAN on it. When the dataset comprises clusters of varied uniform densities, it works well. When there is a density gradient, however, erroneous clusters result. Unless the clusters are well separated, it may split dense clusters or merge sparser ones.

DBSCAN-DLP [20] splits the input data into distinct density levels based on some statistical characteristic of density variation, then estimates the Eps value for each density level before applying DBSCAN clustering to each density level with corresponding Eps. This approach works best with clusters of uniform density; the variance in density of objects inside a cluster should be very minimal and below a certain threshold. The authors of [21] proposed a mathematical method for selecting various Eps values from the k-dist plot and running the DBSCAN algorithm on the data, advancing from least to biggest Eps while disregarding previously clustered elements. To discover inflection points on the curve when the curve alters its concavity, the method employs spline cubic interpolation. Some clusters are divided as a result of this procedure. DSets-DBSCAN [22] applies DSets clustering first, with DBSCAN's input parameters derived from the original cluster extracted by DSets. Most earlier methods calculated Eps based on some local density criteria and used DBSCAN to find clusters in a dataset with several density levels.

CMDD (Clustering Multi-Density Dataset) [23] combines DBSCAN with k-nearest neighbors to generate k local density values for each object in the dataset, this method uses two parameters MinPts and k for k-nearest neighbors, where MinPts represents the lowest density for a core object, and k represents the highest density for a core object in the cluster. It begins by clustering the densest unclassified objects, it is based on the same definitions of DBSCAN and computes k values for each object and needs to arrange the objects in the dataset based on their density to the kth neighbor. So, it consumes more time than the proposed method. It produces good clusters of varied densities, but the denser cluster may take some objects from the adjacent cluster as shown in experiments. The proposed method only computes one value for each object and does not need to arrange the objects in the dataset. So, it starts creating clusters from any object. CMDD method uses cluster initiator or reference. The proposed method does less computation than CMDD.

In [24], the author introduced a clustering method that requires only two intake parameters, uses all k-nearest neighbors to compute the thickness of objects which leads to an increase in the effects of noise on density, and this leads the

method to merge clusters with a smooth gradient in density. The density of object x is compared with the density of its k-nearest neighbors y1, y2, …, yk, then the object x with its similar neighbors are assigned the same cluster-id, how many objects are similar to x is not matter, to expand this cluster; each object y is compared with its k-nearest neighbors p1, p2, …pk. i.e there is no cluster initiator. The proposed method compares between the object x and all objects that are a candidate to be in its cluster using the k-nearest neighbors, there must be MinPts similar object to object x, otherwise, the object x is noise temporally.

EXDBSCAN "an Extension of DBSCAN to detect Clusters in Multi-Density Datasets" [25] requires a single parameter (MinPts), and it assumes that ε (Eps) has a small value, how to select the initial value for ε is not specified, also when creating the next cluster if the point p is an outlier then ε is increased by Δε, how to compute Δε is not specified, the process of increasing ε by Δε may be executed k times before p becomes a core point. The cluster is generated as soon as p becomes a core, and the algorithm then tests whether p is an outlier. If p is an outlier, all points in the cluster are assigned unclassified and p assigned outlier. If the initial value of ε is very small and the first checked point is a core, the method will split this cluster. Since the method starts the creation of the next cluster with the first initial value of ε, this may lead to dividing some clusters.

The authors of [26] presented a clustering technique based on density peaks (DPC). DPC is a new density and distance-based clustering technique. The concept behind this strategy is that cluster centers have high local densities and are spread out. It calculates an object's local density by counting its neighbors over a defined distance termed dc (as Eps in DBSCAN). After locating the centers, clusters are established by allocating each object to the cluster that contains the object's nearest neighbor with a higher density. However, the dc distance has an impact on clustering outcomes, and this approach requires the user to input the number of clusters. This approach can find clusters of various sizes and forms but does not perform well in the presence of varied density clusters, and it does not handle noise well.

In [27], the authors presented "a shared-nearest-neighbor-based clustering by fast search and find of density peaks algorithm" (SNN-DPC). SNN-DPC has some faults. To begin, manually setting the number of shared-nearest neighbors k is required. Second, SNN-DPC still selects cluster centers using a decision graph or requires the number of required clusters as an input. Nonetheless, these center-based algorithms have trouble clustering datasets with a variety of clusters [28].

D. Grid-based Methods

Grid-based clustering is mainly oriented towards spatial datasets. The quantization of the data space into multiple cells is the central principle of these methods. Grid-based approaches do not operate directly with objects; instead, they work with objects in the same grid cell as a single unit, and they merge the cells to build clusters using statistical information. Many algorithms are belonging to this family of clustering algorithms like STING (Statistical Information Grid) [29], wave cluster [30], and CLIQUE (Clustering in Quest) [31].

III. Proposed Method (An Extended DBSCAN Clustering Algorithm)

This section describes the details of the suggested approach that is based on DBSCAN but controls the density permitted within each cluster. The proposed technique finds the k-nearest neighbors for each object, and computes the local density (LD) of each object as the sum of lengths to the MinPts-nearest neighbors; where MinPts is the minimum density for a core object. The maximum density for a core object is less than or equal to *k* since the proposed method relies on the k-nearest neighbors. The Eps value is varied from one object to another since Eps equals the distance to the k^{th} neighbor. For any object x, the method finds its k-nearest neighbors and counts the number of similar objects to object x. If the count of similar objects to object x is larger than or equal to MinPts, object x is a core and the cluster starts to grow, otherwise, object x is assigned noise temporarily. The proposed method counts similar objects within a dynamic radius instead of counting all objects within a fixed radius as in the basic DBSCAN. The method relies on the next mathematical formulas:

Equation (1) defines the set of *k*-nearest neighbors *knn*(*x*) for an object *x*.

$$knn(x) = \{y_i | dis(x, y_i) \leq dis(x, y_k), i = 1,2, \ldots, k\} \quad (1)$$

where $dis(x, y_i)$ is the Euclidean distance.

Equation (2) defines the Local Density (*LD*) of an object *x* as the total of lengths to its MinPts-nearest neighbors.

$$LD(x) = \sum_{i=1}^{MinPts} dis(x, y_i) \quad (2)$$

Equation (3) computes the Density Similarity (DS) between two neighbors.

$$DS(x, y) = \begin{cases} \frac{LD(x)}{LD(y)} * 100 & LD(y) \geq LD(x) \\ \frac{LD(y)}{LD(x)} * 100 & LD(x) > LD(y) \end{cases} \quad (3)$$

Equation (4) checks whether two neighbors are Similar Neighbor (*SN*) or not.

$$SN(x, y) = \begin{cases} 1 & DS(x, y) \geq Similarity\_Level \\ 0 & DS(x, y) < Similarity\_Level \end{cases} \quad (4)$$

Where, $Similarity\_Level$ is an input parameter.

Equation (5) counts the number of similar neighbors to an object *x*.

$$|SN(x, y_1, y_2, \ldots, y_k)| = \begin{cases} \geq MinPts & x\ is\ Core \\ < MinPts, x \notin knn(core) & x\ is\ Noise \\ < MinPts, x \in knn(core) & x\ is\ Porder \end{cases} \quad (5)$$

The proposed method is based on the following definitions:

Definition 1: Directly density-reachable, an object y is directly density-reachable from an object x wrt. $knn(x)$ and MinPts if
1) $y \in knn(x)$ and
2) $|SN(x, y_1, y_2, ..., y_k)| \geq$ MinPts (core object condition).

Definition 2: Density-reachable, an object x is density-reachable from an object y wrt. $knn(y)$ and MinPts if there is a chain of objects $x_1, ..., x_n$ where $x_1 = y$, and $x_n = x$ such that each object is directly density-reachable from the previous object in the chain.

Definition 3: Density-connected, an object x is density-connected to an object y wrt. $knn(z)$ and MinPts if there is an object z such that both objects are density-reachable from z wrt. $knn(z)$ and MinPts.

Definition 4: A cluster C is a non-empty subset of objects meeting the next needs:
1) $\forall x, y$: if $x \in$ C and y is density-reachable from x wrt. $knn(x)$ and MinPts, then y$\in$ C.
2) $\forall x, y \in$ C: x is density-connected to y wrt. $knn(z)$ and MinPts.

Definition 5: Noise is the collection of objects that are not assigned to any cluster.

The method works as follow:

Input: dataset D, k, MinPts, SL(Similarity Level)

Output: set of clusters

E-DBSCAN(dataset D, k, MinPts, SL)

```
All objects are unclassified
For each object x in dataset D
        Find the knn(x) as in (1)
        Compute LD(x) as in (2)
        Compute DS(x,y) as in (3)
        Count SN(x) as in (4)
End for
clusterId=0
for i= 1 to D.size()
   If SN(x_i) ≥ MinPts and unclassified(x_i) then
        clusterId = clusterId +1
        Grow_Cluster(x_i, clusterId)
   End if
End for
End // E-DBSCAN
//**********************************
Grow_Cluster(x , clusterId)
Assign x clusterId
Add all unclassified SN(x,y_j) to seedList and assign them
clusterId
While seedList is not empty
  X=getTop()
  If SN(X) ≥ MinPts
     append all unclassified SN(X, y_j) to seedList and
     assign them clusterId
  End if
  seedList.Remove(X)
End while
End //Grow_Cluster
```

The variable $SN(x)$ stores the number of similar neighbors for the object x and the function $SN(X, y_i)$ returns the objects y_i which are similar to the object X. The function Remove deletes the top element from the seedList after classifying its similar neighbors so the algorithm reaches a terminate point. Cluster creation is started from any core object by calling the function Grow_Cluster(x,id), where x is the first core object in the cluster that has the label id. The next section presents some results that show the ability of the suggested approach in finding clusters of various densities even without separation between clusters.

IV. Results and Discussion

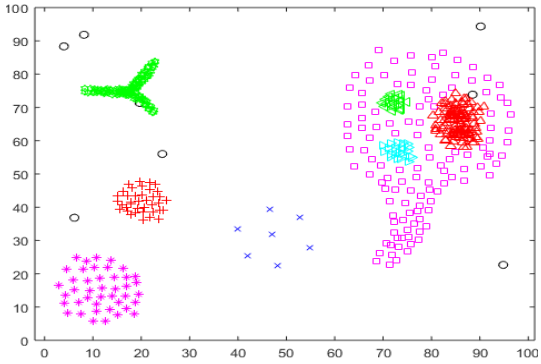
This section explains the outcomes of running the suggested method E-DBSACN to many datasets, which reflects its superior ability to detect clusters of diverse densities. We also compared the suggested method's findings to those of DBSCAN [1], CMDD [23], and the method in [24]. Two dimensions datasets were chosen for easy visualization. Each black circle represents a noise object in all next figures. The first dataset is pictured in Fig. 3. It has 476 objects that form clusters of varied shapes, sizes, and densities. There are six noise objects in total.

The right section of this dataset is the most complex, as it comprises a low-density cluster with three high-density clusters, one of which has two denser clusters. The algorithm discarded the noise objects accurately. The proposed method (E-DBSACN) discovers 8 clusters and 8 noise objects as displayed in Fig. 3(a). When the similarity level is increased from 71 to 73, the red triangles cluster is divided into two clusters, also the magenta squares cluster is divided into two clusters and two more objects are assigned noise as shown in Fig. 3(b). When the similarity level is increased from 73 to 75, the blue stars cluster is divided into two clusters and three more objects are assigned noise as shown in Fig. 3(c).

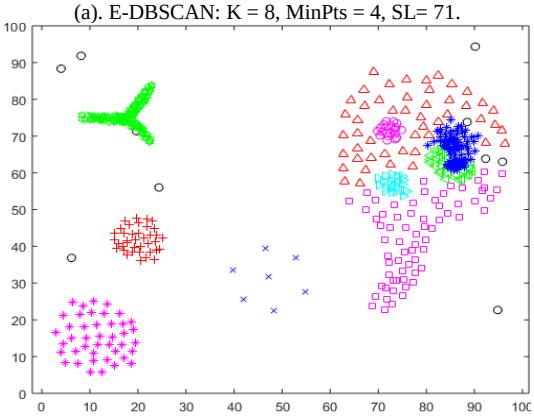
DBSCAN fails to handle this data as shown in Fig 3(d), it returned the smallest cluster as noise, and the right region is returned as a single cluster. There is no proper Eps value to detect the correct clusters in this dataset.

In CMDD method, the denser cluster may take some objects from the adjoining cluster as shown in Fig. 3(e) where the cyan crosses cluster and red pluses cluster take six objects from the green triangles cluster. In addition, this method returns noise as small or singleton clusters as displayed in Fig. 3(e), Fig. 3(f).

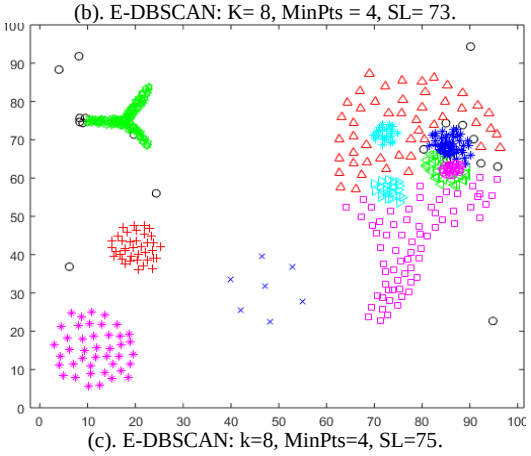
The method in [24] returned the right region as three clusters as shown in Fig. 3(g). It merged the denser cluster with its low-density surrounding cluster since it does not use the concept of a core object as in the proposed method. Also, this method discards the very small (less than 0.006 of the dataset size) clusters as outliers.



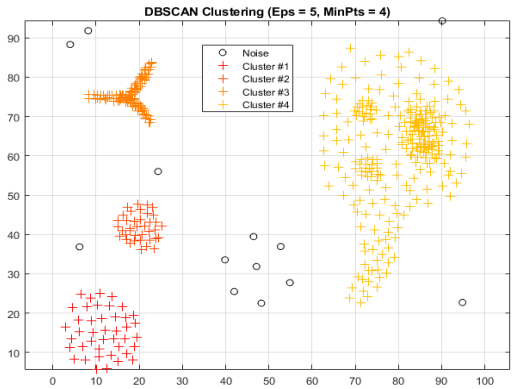
(a). E-DBSCAN: K = 8, MinPts = 4, SL= 71.



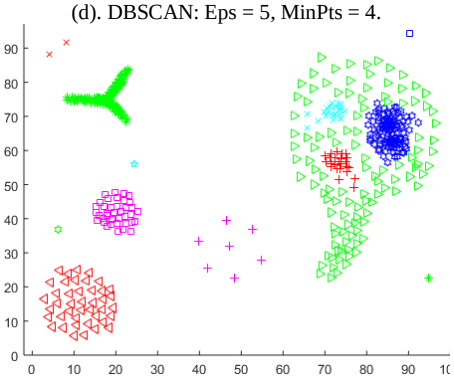
(b). E-DBSCAN: K= 8, MinPts = 4, SL= 73.



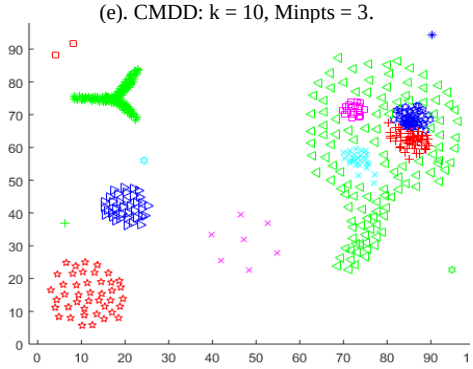
(c). E-DBSCAN: k=8, MinPts=4, SL=75.



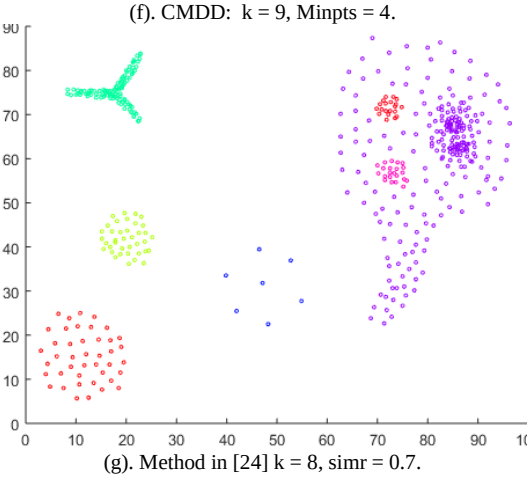
(d). DBSCAN: Eps = 5, MinPts = 4.



(e). CMDD: k = 10, Minpts = 3.

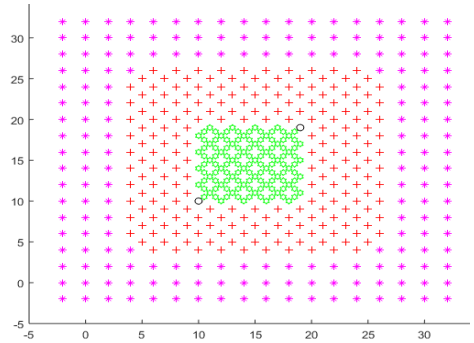


(f). CMDD: k = 9, Minpts = 4.

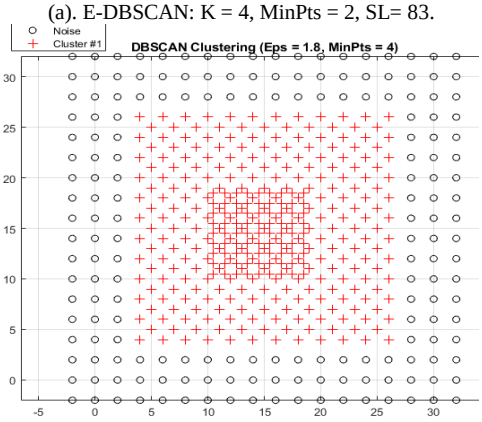


(g). Method in [24] k = 8, simr = 0.7.

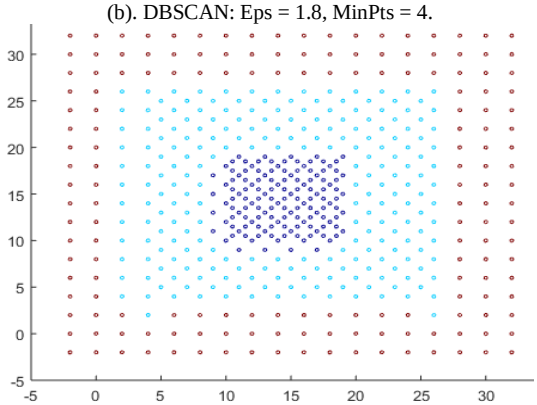
Fig. 3. Dataset 1and the Resulting Clusters.



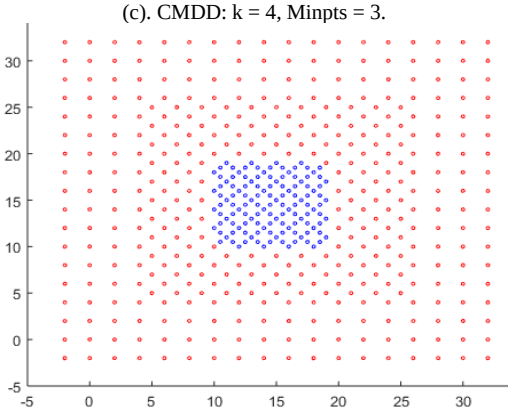
(a). E-DBSCAN: K = 4, MinPts = 2, SL= 83.



(b). DBSCAN: Eps = 1.8, MinPts = 4.



(c). CMDD: k = 4, Minpts = 3.

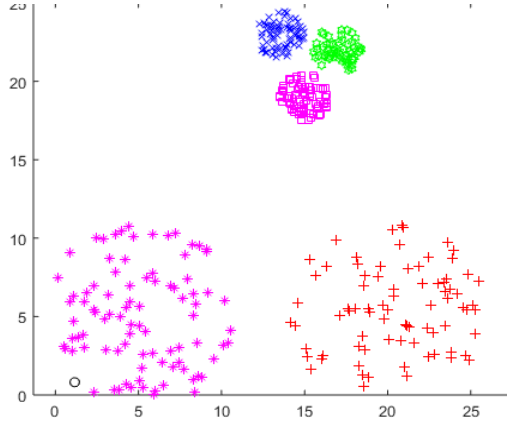


(d). Method in [24]: k = 4, simr = 0.80.

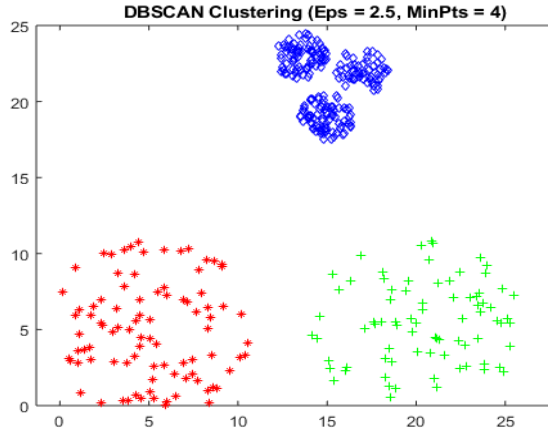
Fig. 4. Dataset 2 and the Resulting Clusters.

Dataset 2 has 526 objects, which is pictured in Fig. 4. The consistent density gradient in this data poses a hurdle. The outer cluster has the lowest density, with two interlaced inner more dense clusters. The E-DBSACN discovered the clusters correctly and assigned two objects noise as shown in Fig. 4(a). DBSCAN merged the two inner clusters and assigned the outer cluster noise as displayed in Fig. 4(b). So, there is no single Eps value to find the correct clusters from this dataset. CMDD has trouble that is shown in Fig. 4(c) where the inner densest cluster took some objects from the middle dense cluster and the middle cluster took some objects from the outer less density cluster. The method in [24] returned two clusters as shown in Fig. 4(d). It merged the outer cluster with the middle one.

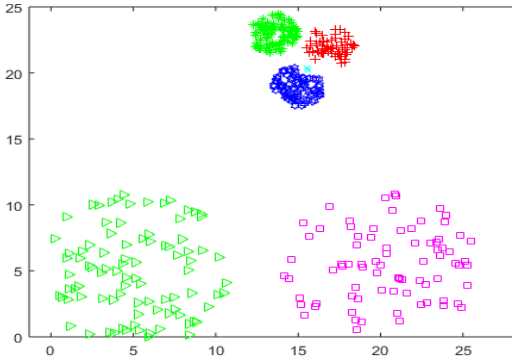
Dataset 3 has 383 objects constituting five convex clusters with two levels of density as presented in Fig. 5. The E-DBSACN discovered the actual five clusters from data as displayed in Fig. 5(a). DBSCAN cannot detect the correct clusters using a single Eps value as portrayed in Fig. 5(b). It merged the three dense clusters since they are close to each other. CMDD returned a good result, but it assigned the two cyan objects a new cluster as shown in Fig. 5(c). The method in [24] returned a good result, but it has one misclassified object (the yellow cluster takes one object from the magenta cluster) as depicted in Fig. 5(d).



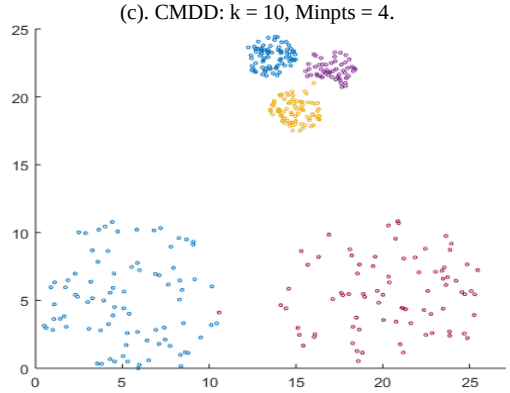
(a). E-DBSCAN: K = 8, MinPts = 4, SL= 52.



(b). DBSCAN: Eps = 2.5, MinPts = 4.



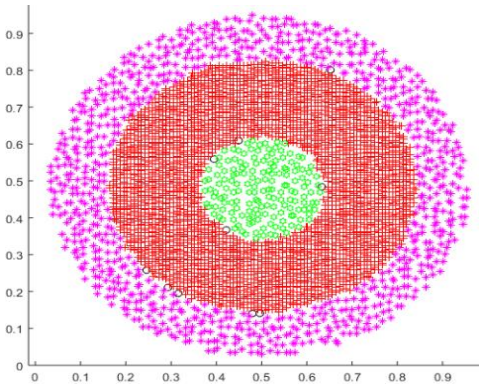
(c). CMDD: k = 10, Minpts = 4.



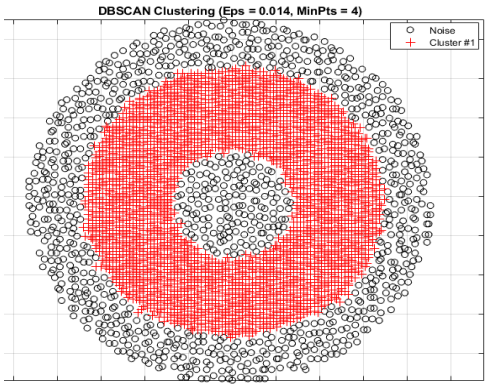
(d). Method in [24] k = 10, simr = 0.75.

Fig. 5. Dataset 3 and the Resulting Clusters.

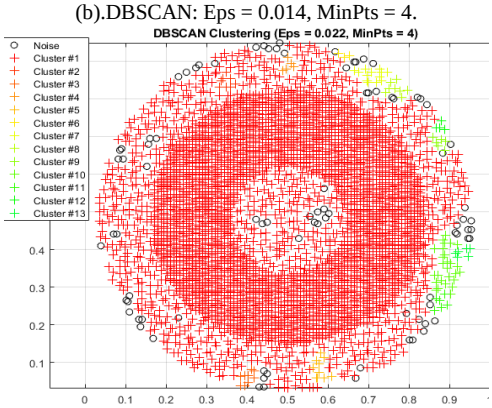
Dataset 4 contains 4600 objects that form three clusters as shown in Fig. 6. The challenge here there is no separation between clusters. Each cluster touches its nearest cluster. The proposed method discovered the accurate clusters and assigned ten objects noise as displayed in Fig. 6(a). Using a small value for Eps, DBSCAN detected the densest cluster and returned the others as noise as shown in Fig 6(b). Increasing the Eps, DBSCAN divided the dataset into 13 clusters. Since objects in the outer and the inner cluster are not of the same density as shown in Fig. 6(c). The problem of CMDD appeared again where the densest green cluster took some border objects from the inner and the outer cluster as shown in Fig. 6(d). This problem emerges when clusters are in touch. The method in [24] produced three clusters as appeared in Fig. 6(e), Fig. 6(f), but it returned 95, 82 objects outlier, respectively.



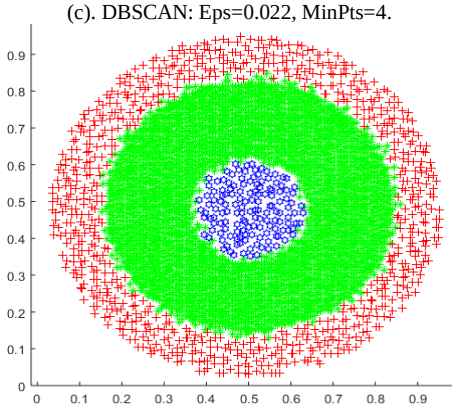
(a). E-DBSCAN: K = 15, MinPts = 7, SL= 80.



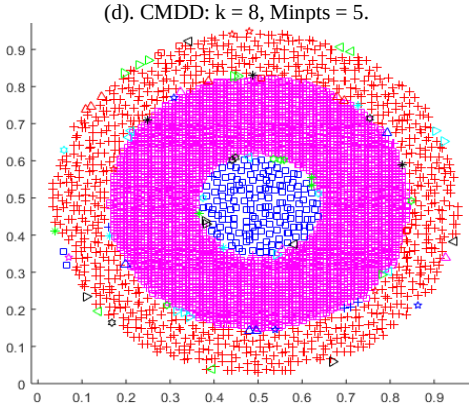
(b).DBSCAN: Eps = 0.014, MinPts = 4.



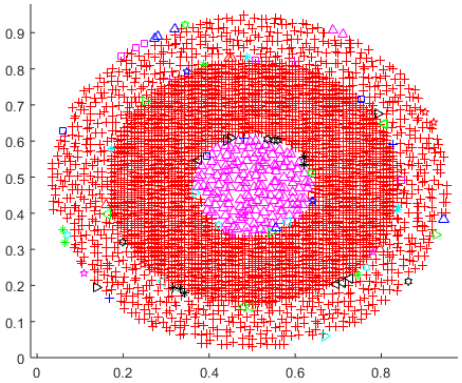
(c). DBSCAN: Eps=0.022, MinPts=4.



(d). CMDD: k = 8, Minpts = 5.



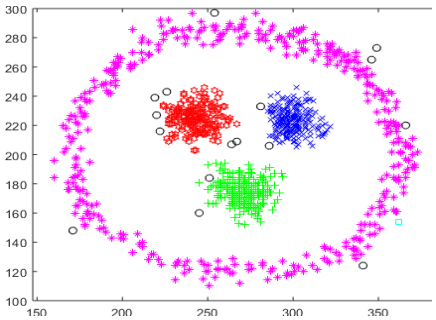
(e). Method in [24] k = 8, simr = 0.89.



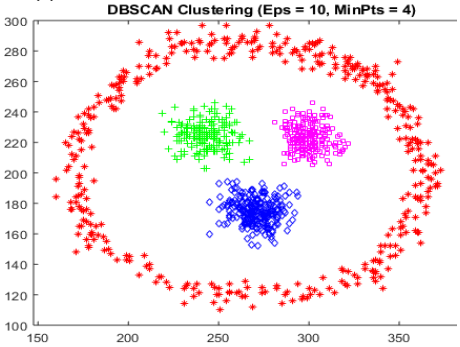
(f). Method in [24] k = 8, simr = 0.88.

Fig. 6. Dataset 4 and the Resulting Clusters.

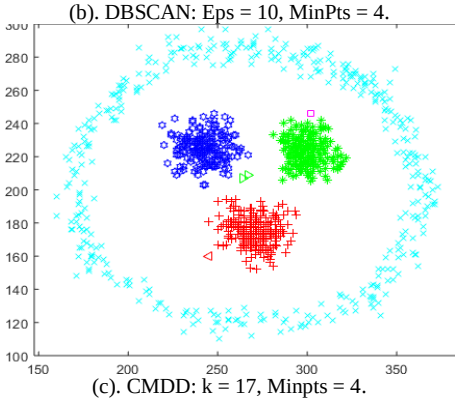
As seen in Fig. 7, Dataset 5 has 1016 objects. As demonstrated in Fig. 7(a), the suggested technique successfully locates the four accurate clusters as well as the noise objects. Other methods failed to discard the noise objects well. But all of them returned good results as shown in Fig. 7(b), (c), (d).



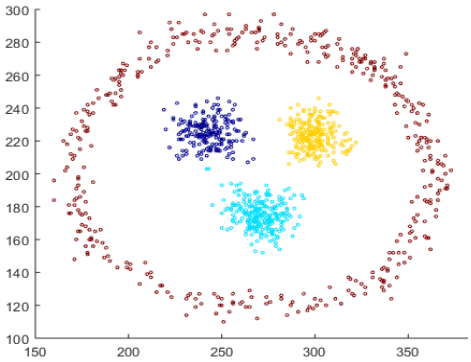
(a). E-DBSCAN: K = 7, MinPts = 3, SL = 50.



(b). DBSCAN: Eps = 10, MinPts = 4.



(c). CMDD: k = 17, Minpts = 4.

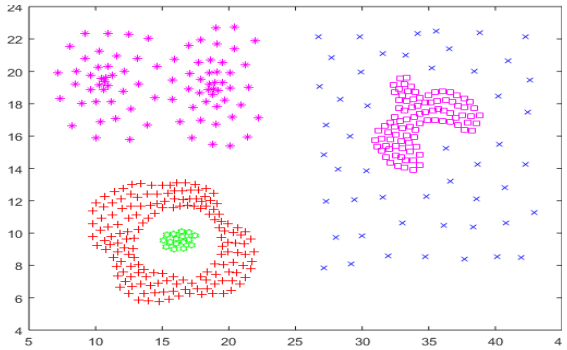


(d). Method in [24]: k =12, simr = 0.6.

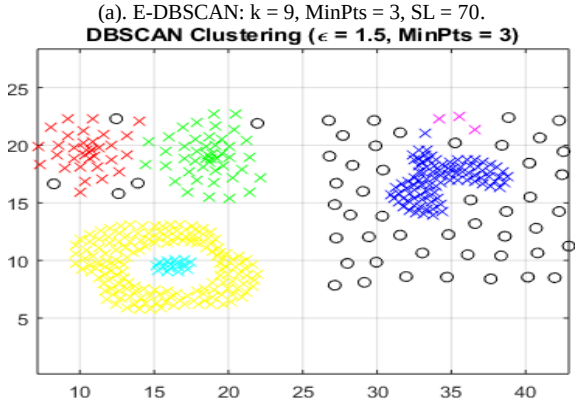
Fig. 7. Dataset 5 and the Resulting Clusters.

Fig. 8 depicts Dataset 6, which has 399 objects. Because it comprises interconnected clusters that are quite close to one other, this dataset is extremely difficult to analyze. The suggested method returned five clusters as shown in Fig. 8(a). In the upper left cluster, there is a smooth gradient in density in this group. Furthermore, the right cluster contains an inner high dense cluster with no separation between them, which the method effectively detects.

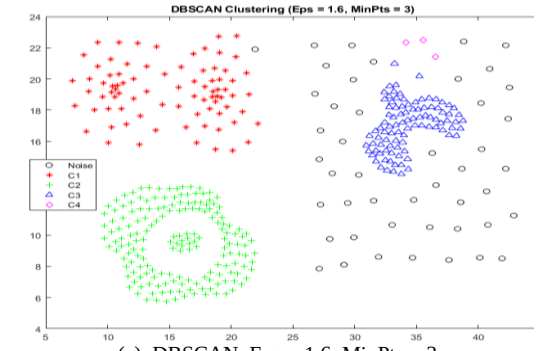
DBSCAN returned five clusters and considered the low-density cluster as noise besides discarding five objects as noises from the two upper left clusters, and two objects are misclassified as shown in Fig 8(b). Raising the Eps value implies decreasing the quality of the outcome as displayed in Fig. 8(c) by combining the two lower left clusters into one and simultaneously merging the two upper left clusters.



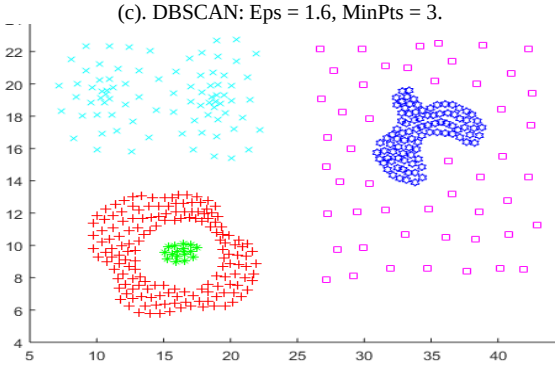
(a). E-DBSCAN: k = 9, MinPts = 3, SL = 70.



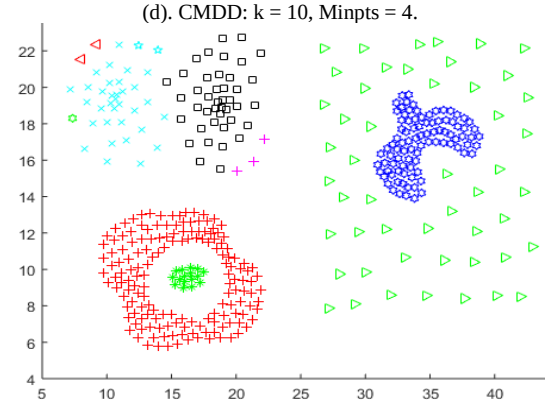
(b). DBSCAN: Eps = 1.5, MinPts = 3.



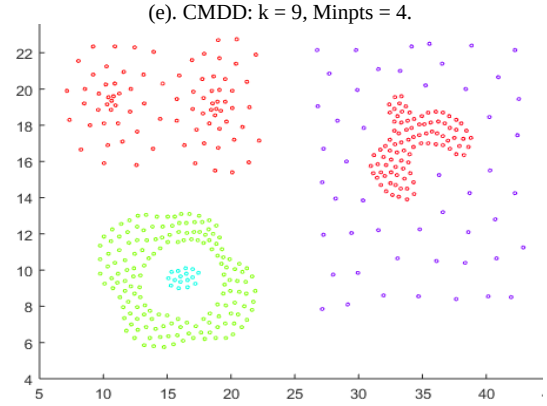
(c). DBSCAN: Eps = 1.6, MinPts = 3.



(d). CMDD: k = 10, Minpts = 4.



(e). CMDD: k = 9, Minpts = 4.

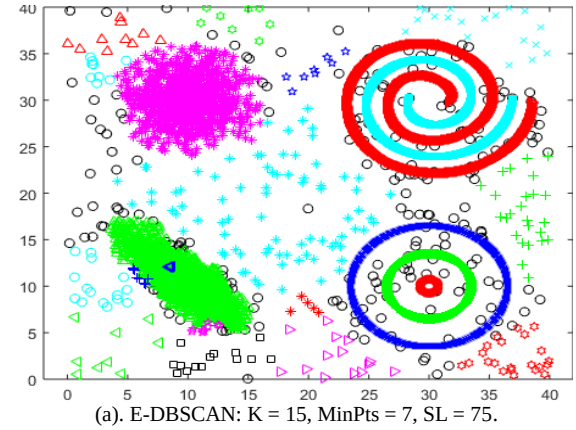


(f). Method in [24] k = 4, simr = 0.72.

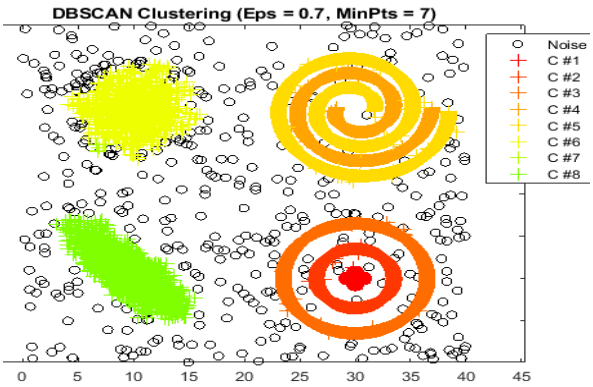
Fig. 8. Dataset 6 and the Resulting Clusters.

CMDD removed five objects from the upper left cluster and three objects from the black cluster as displayed in Fig. 8(e). The suggested method, CMDD, and the method in [24] returned the same result as displayed in Fig. 8(a), (d), (f).

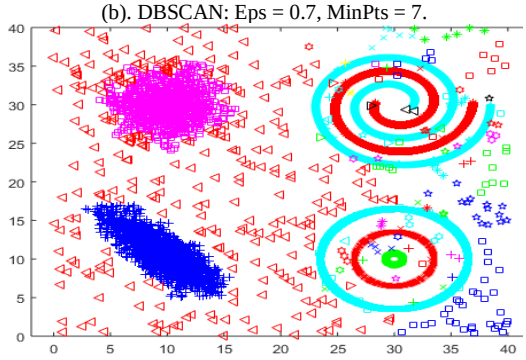
Dataset 7 contains 8537 objects, which is displayed in Fig. 9. The suggested approach discovered the basic seven clusters, some noise objects, and the other objects are allocated to different fifteen clusters as displayed in Fig. 9(a). These fifteen clusters are considered noise in DBSCAN as displayed in Fig. 9(b), but the suggested technique treated them as clusters as depicted in Fig. 9(a). DBSCAN discovered 8 clusters, the eighth cluster is tiny and can be considered noise. The other objects are treated as noise, as shown in Fig. 9(b). CMDD discovered the basic seven clusters, but it can't handle noise, there are many singleton clusters as displayed in Fig 9(c). The technique in [24] discovered eight clusters as displayed in Fig. 9(d), but some sparser objects are assigned to the top left cluster. Also, this method does not handle noise. Comparing the results in subfigures a, b, c, and d, you find that the result in a is more accurate than the others.



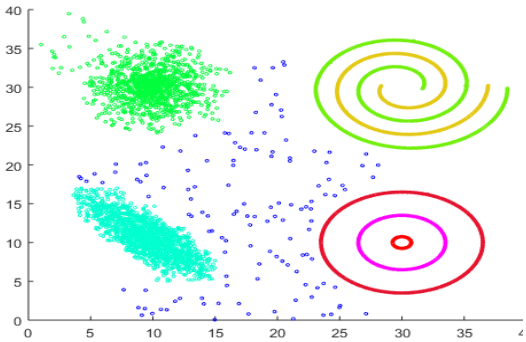
(a). E-DBSCAN: K = 15, MinPts = 7, SL = 75.



(b). DBSCAN: Eps = 0.7, MinPts = 7.



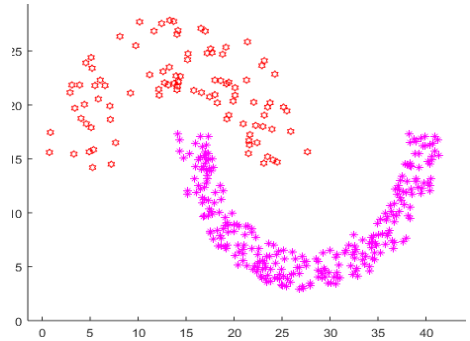
(c). CMDD: k = 30, Minpts = 4.



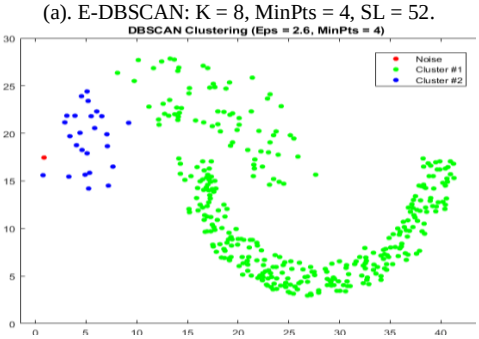
(d). Method in [24]: k = 8, simr = 0.72.

Fig. 9. Dataset 7 and the Resulting Clusters.

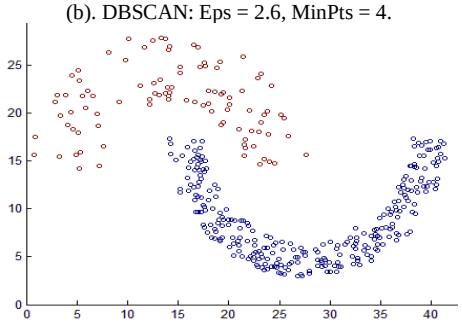
Dataset 8 contains 373 objects that form two clusters of various sizes, forms, and densities. The suggested approach produces the correct clusters accurately as displayed in Fig. 10(a). DBSCAN divided the upper cluster into two clusters and merged one of them with the lower cluster in addition to discarding one noise object as displayed in Fig. 10(b). CMDD and the technique in [24] discovered the correct clusters as displayed in Fig. 10(c), and Fig. 10(d).



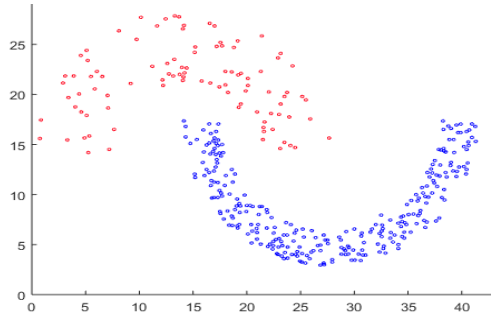
(a). E-DBSCAN: K = 8, MinPts = 4, SL = 52.



(b). DBSCAN: Eps = 2.6, MinPts = 4.



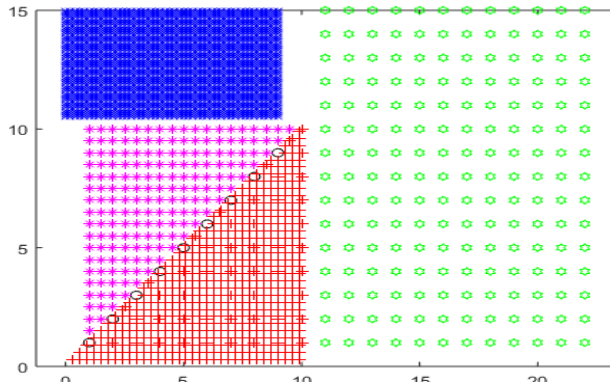
(c). CMDD: k = 13 to 20, Minpts = 4.



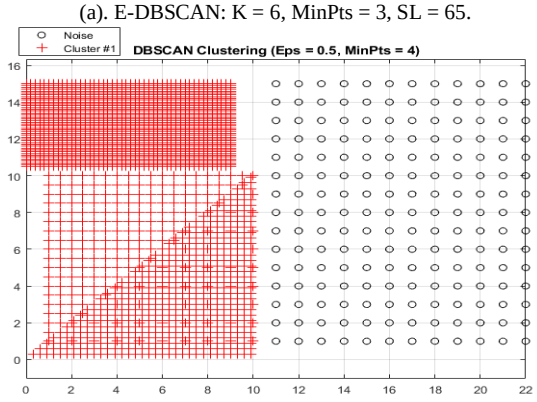
(d). Method in [24]: k = 9, simr = 0.66.

Fig. 10. Dataset 8 and the Resulting Clusters.

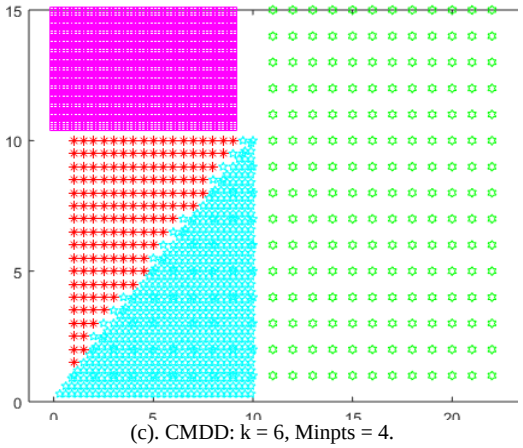
Dataset 9 contains 3147 objects as shown in Fig. 11. It contains four clusters of different sizes, forms, and densities. The suggested method discovered the four clusters from data accurately and nine noise objects as displayed in Fig. 11(a). DBSCAN failed to locate the clusters from this dataset when Eps = 0.5, it merged three clusters into one and treated the fourth cluster as noise as displayed in Fig. 11(b). The same problem of CMDD appeared again where the cyan cluster took 12 objects from the red cluster. These objects are treated as borders for the cyan cluster as shown in Fig. 11(c). The method in [24] produced a better result than that of CMDD, but it has 2 misclassified objects and has removed one object from the lower-left corner of the red cluster as displayed in Fig. 11(d).



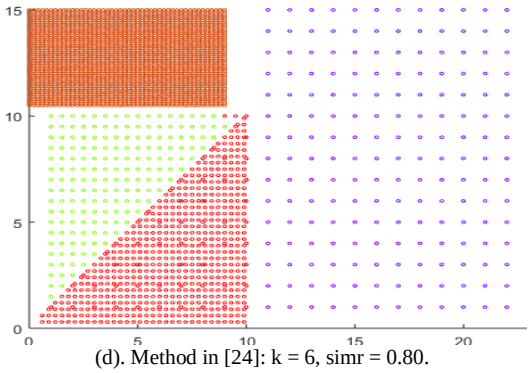
(a). E-DBSCAN: K = 6, MinPts = 3, SL = 65.



(b). DBSCAN: Eps = 0.5, MinPts = 4.



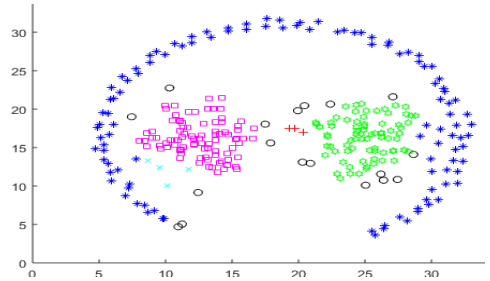
(c). CMDD: k = 6, Minpts = 4.



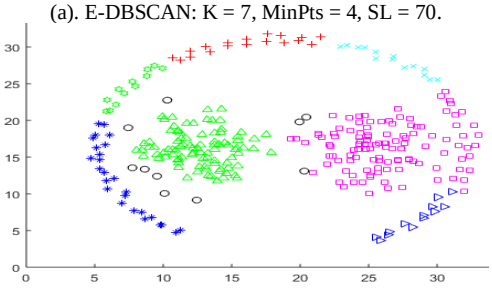
(d). Method in [24]: k = 6, simr = 0.80.

Fig. 11. Dataset 9 and the Resulting Clusters.

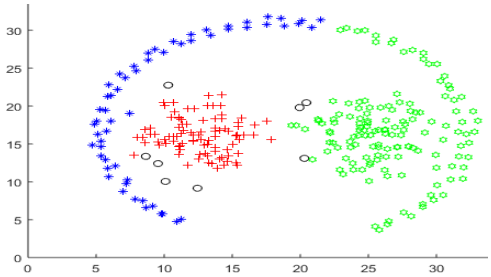
Dataset 10 contains 300 objects as displayed in Fig. 12. The suggested technique discovered the correct three clusters from data in addition to two small clusters (the red and cyan) and assigned eighteen objects noise as displayed in Fig. 12(a). DBSCAN failed to locate the correct clusters. There is no appropriate Eps value to discover the clusters in this data as displayed in Fig. 12(b) and Fig. 12(c). CMDD discovers the three main clusters, it does not distinguish noise objects as in DBSCAN, see Fig. 12(d). The method in [24] failed to discover the correct clusters as displayed in Fig. 12(e).



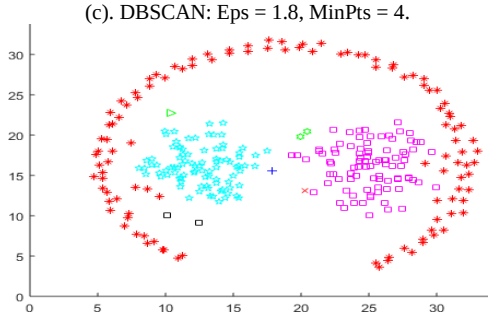
(a). E-DBSCAN: K = 7, MinPts = 4, SL = 70.



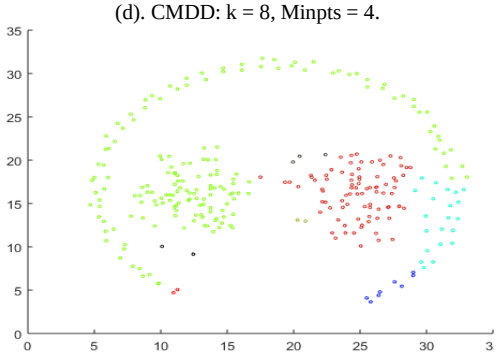
(b). DBSCAN: Eps = 1.6, MinPts = 4.



(c). DBSCAN: Eps = 1.8, MinPts = 4.



(d). CMDD: k = 8, Minpts = 4.



(e). Method in [24]: k = 5, simr = 0.7

Fig. 12. Dataset 10 and the Resulting Clusters.

V. Conclusion

In this work, we have introduced an extended version of the DBSCAN method, the proposed method can handle clusters of diverse densities. This method uses the distance to the k^{th} neighbor to be Eps, this idea makes the Eps vary from one object to another. So, this solves the problem of fixed Eps in the basic DBSCAN. In the proposed method, MinPts is used as in the basic DBSCAN and it must be smaller than the value of *k* in k-nearest neighbors. Since the suggested approach uses k-nearest neighbors it needs a termination condition for cluster expansion, it uses similarity level (SL) as a termination condition. The similarity between two objects relies on their local density. The local density of an object is equal to the total of the lengths to its MinPts-nearest neighbors.

The practical results indicate that the suggested approach (E-DBSCAN) can find clusters of diverse sizes, forms, and densities. All of these qualities for clusters prompted the researchers to propose several changes to the DBSCAN algorithm to handle these challenges jointly, particularly clusters of varying densities.

MinPts parameter ranges from 2 to 7, as a general rule it is near from half of the k-nearest neighbor, *k* for k-nearest neighbors ranges from 4 to 15. The value of SL (similarity level) ranges from 52 to 83.

ACKNOWLEDGMENT

The author extends his appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (IF-PSAU-2021/01/17758).

REFERENCES

[1] M. Ester, H.-P. Kriegel, J. Sander, and X. Xiaowei, "A Density-based algorithm for discovering clusters in large spatial databases with noise," in Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996, pp. 226–231.

[2] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure," in ACM SIGMOD Record, 1999, vol. 28, no. 2, pp. 49–60.

[3] A. Hinneburg and D. A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 1998, vol. September, pp. 58–65.

[4] A. M. Fahim, A. M. Salem, F. A. Torkey, and M. A. Ramadan, "Efficient enhanced k-means clustering algorithm," J. Zhejiang Univ. Sci., vol. 7, no. 10, pp. 1626–1633, 2006, doi: 10.1631/jzus.2006.A1626.

[5] J. E. Gentle, L. Kaufman, and P. J. Rousseuw, Finding Groups in Data: An Introduction to Cluster Analysis., vol. 47, no. 2. 1991.

[6] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," Proc. 20th Int. Conf. Very Large Data Bases, pp. 144–155, 1994.

[7] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," IEEE Trans. Knowl. Data Eng., vol. 14, no. 5, pp. 1003–1016, 2002, doi: 10.1109/TKDE.2002.1033770.

[8] A. Fahim, "K and starting means for k-means algorithm," J. Comput. Sci., vol. 55, no. October, p. 101445, 2021, doi: 10.1016/j.jocs.2021.101445.

[9] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method," Comput. J., vol. 16, no. 1, pp. 30–34, 1973.

[10] D. Defays, "An efficient algorithm for a complete link method," Comput. J., vol. 20, no. 4, pp. 364–366, 1977, doi: https://doi.org/10.1093/comjnl/20.4.364.

[11] S. Guha, R. Rastogi, and K. S. Cure, "An efficient clustering algorithm for large databases," Proc. ACM SIGMOD Int. Conf. Manag. Data, vol. 2, no. 1, pp. 73–84, 1998, [Online]. Available: https://dl.acm.org/citation.cfm?id=276312.

[12] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in ACM SIGMOD int. conference on Management of data, 1996, pp. 103–114, doi: 10.1093/nq/s9-II.32.108-a.

[13] A. Fahim, "Homogeneous Densities Clustering Algorithm," Int. J. Inf. Technol. Comput. Sci., vol. 10, no. 10, pp. 1–10, 2018, doi: 10.5815/ijitcs.2018.10.01.

[14] M. Debnath, P. K. Tripathi, and R. Elmasri, "K-DBSCAN: Identifying spatial clusters with differing density levels," in Proceedings - 2015 International Workshop on Data Mining with Industrial Applications, DMIA 2015: Part of the ETyC 2015, 2015, pp. 51–60, doi: 10.1109/DMIA.2015.14.

[15] A. Fahim, "A Clustering Algorithm based on Local Density of Points," Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 12, pp. 9–16, 2017, doi: 10.5815/ijmecs.2017.12.02.

[16] C. Jungan, C. Jinyin, Y. Dongyong, and L. Jun, "A k-Deviation Density Based Clustering Algorithm," Math. Probl. Eng., vol. 2018, 2018, doi: 10.1155/2018/3742048.

[17] C. Xiaoyun, M. Yufang, Z. Yan, and W. Ping, "GMDBSCAN: Multi-density DBSCAN cluster based on grid," in IEEE International Conference on e-Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08, 2008, pp. 780–783, doi: 10.1109/ICEBE.2008.54.

[18] M. A. Alhanjouri and R. D. Ahmed, "New Density-Based Clustering Technique: GMDBSCAN-UR," Int. J. Adv. Res. Comput. Sci., vol. 3, no. 1, pp. 1–9, 2012.

[19] P. Liu, D. Zhou, and N. Wu, "Varied Density Based Spatial Clustering of Application with Noise," Proc. IEEE Conf. ICSSSM 2007 pg 528, vol. 531, pp. 528–531, 2007.

[20] Z. Xiong, R. Chen, Y. Zhang, and X. Zhang, "Multi-density DBSCAN algorithm based on Density Levels Partitioning," J. Inf. Comput. Sci., vol. 9, no. 10, pp. 2739–2749, 2012.

[21] S. Louhichi, M. Gzara, and H. Ben Abdallah, "A density based algorithm for discovering clusters with varied density," 2014 World Congr. Comput. Appl. Inf. Syst. WCCAIS 2014, no. 1, 2014, doi: 10.1109/WCCAIS.2014.6916622.

[22] J. Hou, H. Gao, and X. Li, "DSets-DBSCAN: A Parameter-Free Clustering Algorithm," IEEE Trans. Image Process., vol. 25, no. 7, pp. 3182–3193, 2016, doi: 10.1109/TIP.2016.2559803.

[23] A. Fahim, "Clustering Algorithm for Multi-density Datasets," Rom. J. Inf. Sci. Technol., vol. 22, no. 3–4, pp. 244–258, 2019, doi: 10.1007/978-3-030-38501-9_2.

[24] A. Fahim, "A Clustering Algorithm for Varied Density Clusters based on Similarity of Local Density of Objects," in Proceedings of the International Conference on Intelligent Computing and Control Systems, ICICCS 2020, 2020, no. Iciccs, pp. 26–31, doi: 10.1109/ICICCS48265.2020.9121008.

[25] A. Ghanbarpour and B. Minaei, "EXDBSCAN: An extension of DBSCAN to detect clusters in multi-density datasets," 2014 Iran. Conf. Intell. Syst. ICIS 2014, pp. 1–5, 2014, doi: 10.1109/IranianCIS.2014.6802561.

[26] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," Science (80-.)., vol. 344, no. 6191, pp. 1492–1496, 2014, doi: 10.1126/science.1242072.

[27] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," Inf. Sci. (Ny)., vol. 450, pp. 200–226, 2018, doi: 10.1016/j.ins.2018.03.031.

[28] X. X. Wang, Y. F. Zhang, J. Xie, Q. Z. Dai, Z. Y. Xiong, and J. P. Dan, "A density-core-based clustering algorithm with local resultant force," Soft Comput., vol. 24, no. 9, pp. 6571–6590, 2020, doi: 10.1007/s00500-020-04777-z.

[29] W. Wang, J. Yang, and R. Muntz, "STING : A statistical information grid approach to spatial data mining," Proc. 23rd Int. Conf. Very Large Databases, VLDB 1997, pp. 186–195, 1997.

[30] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," Proc. Int. Conf. Very Large Data Bases, no. 24, pp. 428–439, 1998.

[31] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," ACM SIGMOD Iternational Conf. Manag. Data, vol. 27, no. 2, pp. 94–105, 1998.