K-DBSCAN: Identifying Spatial Clusters With Differing Density Levels

Madhuri Debnath
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas
Email: madhuri.debnath@mavs.uta.edu

Praveen Kumar Tripathi
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas
Email: praveen.tripathi@mavs.uta.edu

Ramez Elmasri
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas
Email: elmasri@cse.uta.edu

***Abstract*—Spatial clustering is a very important tool in the analysis of spatial data. In this paper, we propose a novel density based spatial clustering algorithm called K-DBSCAN with the main focus of identifying clusters of points with similar spatial density. This contrasts with many other approaches, whose main focus is spatial contiguity. The strength of K-DBSCAN lies in finding arbitrary shaped clusters in variable density regions. Moreover, it can also discover clusters with overlapping spatial regions, but differing density levels. The goal is to differentiate the most dense regions from lower density regions, with spatial contiguity as the secondary goal. The original DBSCAN fails to discover the clusters with variable density and overlapping regions. OPTICS and Shared Nearest Neighbour (SNN) algorithms have the capabilities of clustering variable density datasets but they have their own limitations. Both fail to detect overlapping clusters. Also, while handling varying density, both of the algorithms merge points from different density levels. K-DBSCAN has two phases: first, it divides all data objects into different density levels to identify the different natural densities present in the dataset; then it extracts the clusters using a modified version of DBSCAN. Experimental results on both synthetic data and a real-world spatial dataset demonstrate the effectiveness of our clustering algorithm.**

I. Introduction

Clustering is the process of grouping a set of objects into classes or clusters so that the similarity between the objects within the same cluster is maximized. For researchers who work with geographical and other types of spatial data, data mining has offered many useful and promising tools for data analysis. Spatial clustering is one of these tools [1].

Spatial Clustering has a wide range of applications. Some of them include crime hot-spot analysis, identification of similar land usage, earthquake analysis, agricultural environment analysis and merging of regions with similar weather patterns.

Spatial databases have some unique challenges. So, in order to choose a clustering algorithm that is suitable for a particular spatial application, some important issues need to be considered [2].

- Clustering algorithms should identify irregular shapes. Partitioning algorithms like K-means [3] or K-medoids [4] can discover clusters with spherical shapes and similar size. Density-based clustering algorithms like DBSCAN [5] are more suitable to find arbitrary shaped clusters.
- The algorithms should not be sensitive to the order of input. That means, clustering results should be independent of data order. For example, cluster quality and efficiency in K-means [3] depends on the choice of initial seeds, while cluster results in DBSCAN [5] do not depend on the data order.
- Algorithms should handle data with outliers. Density-based algorithms like DBSCAN [5] and OPTICS [6] can handle noise, while K-means [3] cannot.
- Algorithms should not be too sensitive to user specified parameter. For example, existing density-based algorithms like DBSCAN [5], DENCLUE [7] and OPTICS [6] need a careful choice of threshold for density, because they may produce very different results even for slightly different parameter settings.
- Lastly, clustering algorithms should handle spatial data with varying density. DBSCAN [5] fails to cluster this kind of data.

Motivated by these challenges, we propose a new density-based spatial clustering algorithm K-DBSCAN to analyse spatial data that can handle data with different density levels. Unlike the DBSCAN [5] algorithm, it does not depend on the global ϵ parameter to calculate neighbourhood, rather each data point dynamically generates its own parameter to define its neighbourhood. Hence, it has less sensitivity to user specified parameter.

Our proposed K-DBSCAN algorithm can be utilized in several applications. For example, it can be used to find spatial clusters with differing population density levels, even when these clusters are overlapping. Spatial analysis of regions based on population has important application in urban planning, healthcare and economic development. Population density levels of different regions are different.

The rest of the paper is organized as follows. In Section 2, we review some related works. We describe our proposed algorithm in Section 3. Section 4 presents experimental results of our algorithm and compares the quality of the clustering result with three other well-known algorithms. In Section 5, we present a practical application of our algorithm with a real-world spatial dataset. Finally Section 6 concludes the paper.

978-1-4673-8111-6/15 $31.00 © 2015 IEEE
DOI 10.1109/DMIA.2015.14

II. Related Work

Spatial Clustering algorithms can be partitioned into four general categories: Partitioning, hierarchical, density-based and grid-based.

Partitioning algorithms divide the entire dataset into a number of disjoint groups. Each disjoint group is a cluster. K-means [3], EM (Expectation Maximization) [8] and K-medoid [4] are three well-known partitioning based clustering algorithms. These use an iterative approach and try to group the data into K clusters, where K is a user specified parameter. The shortcoming of the algorithms is that they are not suitable for finding arbitrary shaped clusters. Further, they are dependent on the user specified parameter K.

Hierarchical clustering algorithms use a distance matrix as an input and generates a hierarchical set of clusters. This hierarchy is generally formed in two ways: bottom-up and top-down [4]. The top-down approach starts with all the objects in the same cluster. In each successive iteration a bigger cluster is split into smaller clusters based on some distance measure, until each object is in one cluster itself. The clustering level is chosen between the root (a single large cluster) and the leaf nodes (a cluster for each individual object). The bottom-up approach starts with each object as one cluster. It then successively merges the clusters until all the clusters are merged together to form a single big cluster. The weakness of the hierarchical algorithms is that they are computationally very expensive.

BIRCH [9] and CURE [10] are hierarchical clustering algorithms. In BIRCH, data objects are compressed into small sub-clusters, then the clustering algorithm is applied on these sub-clusters. In CURE, instead of using a single centroid, a fixed number of well scattered objects are selected to represent each cluster.

Density-based methods can filter out the outliers and can discover arbitrary shaped clusters. DBSCAN [5] is the first proposed density-based clustering algorithm. This algorithm is based on two parameters: ϵ and $MinPts$. Density around each point depends on the number of neighbours within its ϵ distance. A data point is considered dense if the number of its neighbours is greater than $MinPts$. DBSCAN can find clusters of arbitrary shapes, but it cannot handle data containing clusters of varying densities. Further, the cluster quality in DBSCAN algorithm depends on the ability of the user to select a good set of parameters.

OPTICS [6] is another density based clustering algorithm, proposed to overcome the major weakness of DBSCAN algorithm. This algorithm can handle data with varying density. This algorithm does not produce clusters explicitly, rather computes an augmented cluster ordering such that spatially closest points become neighbours in that order.

The DENCLUE [7] algorithm was proposed to handle high dimensional data efficiently. In this algorithm density of a data object is determined based on the sum of influence functions of the data points around it. DENCLUE also requires a careful selection of clustering parameters which may significantly influence the quality of the clusters.

The Shared Nearest Neighbour (SNN) [11] clustering algorithm was proposed to find clusters of different densities in high dimensional data. A similarity measure is based on the number of shared neighbours between two objects instead of traditional Euclidean distance. This algorithm needs 3 parameters (k, ϵ, $MinPt$).

Grid-based clustering algorithm divides the data space into a finite number of grid cells forming a grid structure on which operations are performed to obtain the clusters. Some examples of grid based methods include STING [12], Wave-Cluster [13] and CLIQUE [14]. The STING [12] algorithm calculates statistical information in each grid cells. The Wave-Cluster [13] algorithm applies wavelet transformation to the feature base. Input parameters include the number of grid cells for each dimension. This algorithm is applicable for low dimensional data space. The CLIQUE [14] algorithm adopts a combination of grid-based and density-based approaches and this algorithm can detect clusters in high-dimensional space.

III. Proposed Algorithm

In this section, we focus on the basic steps of our proposed algorithm. We propose K-DBSCAN algorithm, which works in two phases.

- K Level Density Partitioning: In this phase, we calculate the density of each data point based on its distance from its nearest neighbouring data points. Then we partition all the data points into K groups based on their density value.
- Density Level Clustering: In this phase, we introduce a modified version of DBSCAN algorithm that works on different density levels.

A. K-DBSCAN Phase 1 - K level Density Partitioning

In real world spatial datasets, different data objects may be located in different density regions. So, it is very difficult or almost impossible to characterize the cluster structures by using only one global density parameter [15].

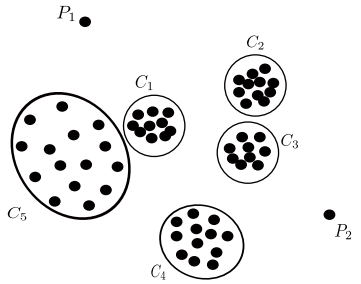


Fig. 1: Points in different density regions

Consider the example from Figure 1. In this example, points in clusters C_1, C_2 and C_3 represents very dense neighbourhoods. Points in cluster C_4 represents a less dense region, while points in cluster C_5 represent a sparse neighbourhood. Point P_1 and P_2 should be considered as noise or outliers. As different data points are located in different density regions, it is impossible to obtain all the clusters simultaneously using one global density parameter. Because, if we consider the density estimation for points located in C_1, we have to choose

a smaller ϵ value and we will find clusters C_1, C_2 and C_3. All other points will be considered as outliers. However, if we want to discover cluster C_5, we have to choose a larger value of ϵ, but this may result in a bigger cluster by including most of the data points as its neighbour (for example, C_1 may merge with C_5 forming a bigger cluster).

Consider another example in Figure 2. Here cluster C_1 is surrounded by another cluster C_2. Points in C_2 represent a less dense region, while points in C_1 are in a high density region. It is also difficult to identify both clusters using *DBSCAN* [5].

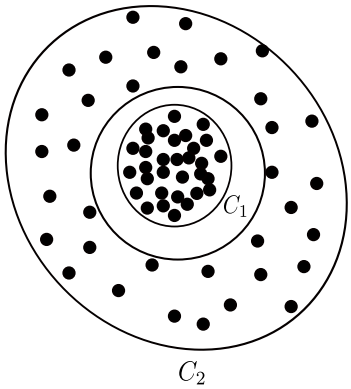


Fig. 2: One cluster surrounding another cluster

To overcome these problems, we aim to partition all the data points based on their density values. The density value of a point depends on the distance of the point from its nearest neighbours.

There are different measures to find the density of a point. In *DBSCAN* [5], density of a point is defined as the total number of neighbours within a given radius(ϵ) of the point. But this does not work well in dataset with varying densities. A large number of points may be considered as noise due to lack of significant neighbourhood, in terms of global ϵ. SNN [11] clustering algorithm used the same definition for density measure, but they did it in terms of shared nearest neighbour similarity, which is relatively insensitive to variation of density. Still we have to estimate a global radius parameter, ϵ.

In this paper, we have defined a new measure to find density. Here we introduce some important definitions required for this algorithm.

Definition 1: l-nearest neighbour distance: The l-nearest neighbour distance of a point P, denoted as $dist_l(P)$ is the distance between the point P and its l^{th} nearest neighbour. Where, $l \geq 1$. If $l\text{-}set(P)$ is the set of l closest neighbours of point P and d_i is the distance from point P to its i^{th} closest neighbour, then

$$dist_l(P) = \max\{d_i\} \tag{1}$$

Definition 2: l-density: The l-density of a point P, denoted as l-density(P) is defined as follows

$$l\text{-}density(P) = \frac{1}{l}\sum_{i=1}^{l} d_i \tag{2}$$

1) Partitioning: We divide the data points into K groups based on their density values. For each point P, we calculate l-density(P). Then, we implement the K-means [3] clustering algorithm to divide all points into K groups based on their l-density values. As a result, each point will be assigned to a number from 1 to K. We denote this number as the density level of a point. The motivation behind this is to divide the points into K density levels.

For example, consider that, we have 10 points from p_1 to p_{10}. We compute their $l\text{-}density(P)$ values which are,

$l\text{-}density$ = $\{1.2, 1.5, 4.5, 4.2, 3.3, 1.9, 2.2, 3.7, 4.0, 4.9\}$.

We implement K-means algorithm on this set with K = 2, we get the resulting density level set.

$density\text{-}level$ = $\{1, 1, 2, 2, 2, 1, 1, 2, 2, 2\}$.

Points p_1, p_2, p_6 and p_7 have the same density level 1. Whereas, points p_3, p_4, p_5, p_8, p_9 and p_{10} have density level 2.

Definition 3: Density-Level: Density-Level of a point P, denoted by *density-level*(P) is an integer number, labelled by K-means algorithm. For two points P and Q, if their density levels are the same, then the l-density of P and Q are approximately similar. Note that, density-level is only a categorical parameter.

2) How to choose the value of K: In this algorithm K is a user specified parameter, which plays an important role in finding a good clustering result. The value of K indicates the number of different dense regions in the whole data space. If we choose value of K = 1, it signifies that all the data points are in the same dense region.

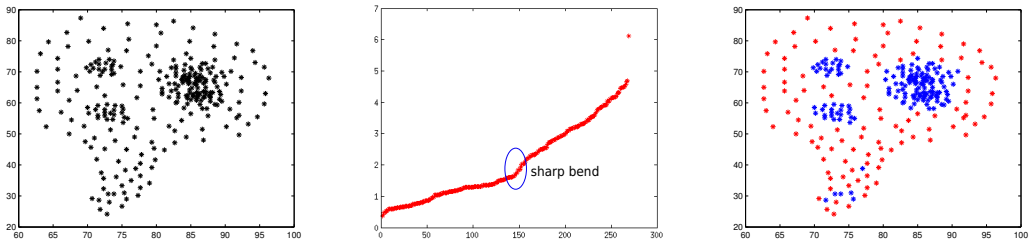
In order to determine the value of K, we sort the l-density value of all points in increasing order and plot them. We look for the strong or slightly strong bends in this graph. Consider a sample dataset in Figure 3a, whose sorted *l-density* plot is given in Figure 3b. There is a clear bend in this graph. So, we can visualize the dataset and divide it into 2 density levels.

In this dataset, we implement K-means [3] clustering algorithm with K = 2. Hence, we get two intermediate clusters based on two density levels. Blue points have level 2 density and red points have level 1 density (see Figure 3c).

B. K-DBSCAN Phase 2 - Density Level Clustering

Step 1 partitions the data points into different density levels. Step 2 is a modified version of DBSCAN algorithm, which considers spatial distance as well as the density level difference between points while clustering.

the K-DBSCAN algorithm does not depend on the single global parameter ϵ, rather each point defines its neighbourhood region dynamically based on its density value. We introduce the idea of *Density level neighbourhood* of a point. *Density*



(a) Dataset-1 (b) Sorted *l-density* plot (c) Two density levels

Fig. 3: K-level Density Partitioning

level neighbours of a point P_i are the points that reside inside the neighbourhood region of P_i and that have the same density level as that of P_i.

Consider the example in Figure 4. We assume that, density level of all points in C_1 is 1 and density level of all points in C_2 is 2. So, point P_1 is assigned with density-value 2. Only the *blue* points inside P_1's neighbourhood radius are defined as *Density level neighbourhood* of P_1.

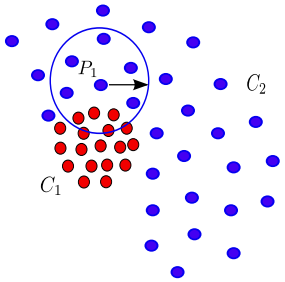


Fig. 4: Neighbourhood of a Point

Definition 4: Neighbourhood radius of a point (ϵ_i): The neighbourhood radius of a point P_i is defined as ϵ_i is $\epsilon_i = dist_l(P_i)$.

Definition 5: Density Level Neighbourhood of a point $N(P_i)$: This is defined by $N(P_i) = \{Q \in D \mid dist(P_i, Q) \leq \epsilon_i$ and density-level(P_i) = density-level(Q)$\}$.

The following definitions from 6 to 9 are similar to the DBSCAN definitions except for the differences in neighbourhood that take density level into account in definitions 6 and 7.

Definition 6: Directly density reachable: A point q is directly density reachable from point P_i wrt. ϵ_i if $q \in N(P_i)$.

Definition 7: Density reachable: A point q is density reachable from point P_i if there is a chain of points $p_1 .....p_n, p_1 = P_i$ and $p_n = q$ such that p_k is directly density reachable from point p_{k+1}.

Definition 8: Density connected: A point P is density connected to point Q if there is an intermediate point o such that both P and Q are density reachable from point o.

Definition 9: Cluster: A cluster C is a non-empty subset of the whole dataset of points satisfying the following conditions:
1. $\forall\ p, q$: if $p \in C$ and q is density reachable from p, then $q \in C$.
2. if $p, q \in C$: p is density connected to q.

Definition 10: Outliers: A cluster must have at least $MinPts$, which is a user specified parameter. If the number of points in a cluster is less than the threshold $MinPts$, we consider the points as outlier, that do not belong to any cluster.

Clustering Algorithm: In this section, we present our density-based clustering algorithm. The structure of K-DBSCAN algorithm is given in Algorithm 1, which invokes ExpandCluster (see Algorithm 2) method and RegionQuery (see Algorithm 3) method. Input D is the set of data points. Another input *DL* is an array containing the *density-levels* of all points generated by a K-means [3] algorithm, that we described in the Partitioning phase (see Section 3.1.1).

Algorithm 1 K-DBSCAN

Inputs:
D: a set of data points ($P_1, P_2, P_3, ...P_n$)
DL: a set of density level of the corresponding data points ($dl_1, dl_2, dl_3....dl_n$)
Outputs:
CL: a set of clusters

```
1:  procedure K-DBSCAN(D, DL)
2:      C ← 0 /* C is cluster id */
3:      for each unvisited point P_i do
4:          mark P_i as visited
5:          Calculate ε_i /* see Definition 4 */
6:          dl_i ← DL[P_i]
7:          NeighborPts ← regionQuery(P_i, dl_i, ε_i)
8:          C ← C + 1
9:          ExpandCluster(P_i, NeighborPts, C, DL)
10:     end for
11: end procedure
```

RegionQuery method returns all the density level neighbours of a point (see Definition 5). ExpandCluster method does the cluster formation. If a point P is assigned to a cluster C, its density level neighbours are also part of the same cluster C. This process continues until all the density connected (see Definition 8) points are found (see Algorithm 2).

Algorithm 2 ExpandCluster

```
1: procedure EXPANDCLUSTER(P_i,Neighbor,C,DL)
2:     Assign P_i to Cluster C
3:     for each point p_j in Neighbor do
4:         Calculate ε_j
5:         dl_j ← DL[p_j]
6:         NeighboursPts_j ← regionQuery(p_j, dl_j, ε_j)
7:         for all points p_k in NeighbourPts_j do
8:             if DL[P_i] = DL[p_k] then
9:                 NeighbourPts ← NeighbourPts ∪ p_k
10:            end if
11:        end for
12:        if p_j is not yet assigned to any cluster then
13:            assign p_j to cluster C
14:        end if
15:    end for
16: end procedure
```

Algorithm 3 RegionQuery

Inputs:
P_i*: i-th data point*
dl_i*: density level of point* P_i
ϵ_i*: Neighbourhood radius of point* P_i
Outputs:
S: a set of neighbour points

```
1: procedure REGIONQUERY(P_i, dl_i, ε_i)
2:     Return all points within P_i's ε_i-neighbourhood (including P_i) and that has the same density-level as dl_i
3: end procedure
```

IV. Experiments and Comparison with other methods

In this section, we evaluate the effectiveness of our clustering algorithm. We used two different datasets. Dataset-1 is a synthetic dataset, whereas, dataset-2 is a real-world spatial dataset. We compare our method with three well known density based clustering algorithm, DBSCAN [5], Shared Nearest Neighbour (SNN) [11] and OPTICS [6].

A. Experiment 1

Dataset-1 is a synthetic dataset, that was also used in [15] (see Figure 3a). We use K = 2 (2 density levels) and $MinPts$ = 5. The result of K-DBSCAN is shown in Figure 5a. In this result, different colors indicate different clusters. We get 5 clusters. cluster C_5 is the big cluster with lower density level that surrounds the other clusters C_1, C_2, C_3, C_4.

We change the value of K to 3 (3 density levels). Figure 5b shows the result. Here we get 8 clusters. Cluster C_8 represents the lowest density level. Cluster C_4, C_5 and C_6 and C_7 are at the middle density level. Whereas cluster C_1, C_2 and C_3 are at the highest density level.

Figure 6a shows the result of DBSCAN algorithm with parameters ϵ = 2 and $MinPts$ = 5. We got 3 clusters here (red, blue and green points). A large number of points are not being clustered as they are considered as outliers. We change the value of ϵ = 4 and Figure 6b shows the result. We get only one big cluster (red points). Obviously one big cluster does not contain any meaningful information.

Figure 7a, Figure 7b and Figure 7c show the results of *Shared Nearest Neighbour* [11] algorithm on the same dataset. In this algorithm, the value nearest neighbour list size, k, is important to determine the granularity of clusters. If k is too small, even a uniform cluster will be broken up into multiple clusters, and the algorithm will produce a large number of small clusters. On the other hand, if k is too large, the algorithm will generate only a few, well separated clusters [11]. We use 3 different values for k and compare the results with our algorithm.

Figure 7a shows the result of SNN algorithm with k = 15. Here this algorithm produces 37 small clusters, with a lot outliers.

Figure 7b shows the clustering result with k = 30. We get 18 clusters. We can see that cluster C_3 (From Figure 5) has been broken into multiple different clusters. Similarly, cluster C_5 (From Figure 5) has also been broken down into multiple clusters. Also, we find the points in different density levels are mixed together and form a single cluster.

Figure 7c shows the result with k = 50. Here, we get 8 clusters. But still one of the dense clusters (Cluster C_1 from Figure 5b) has been broken into multiple small sized clusters. Further, some points from different density levels are merged together to form a single cluster.

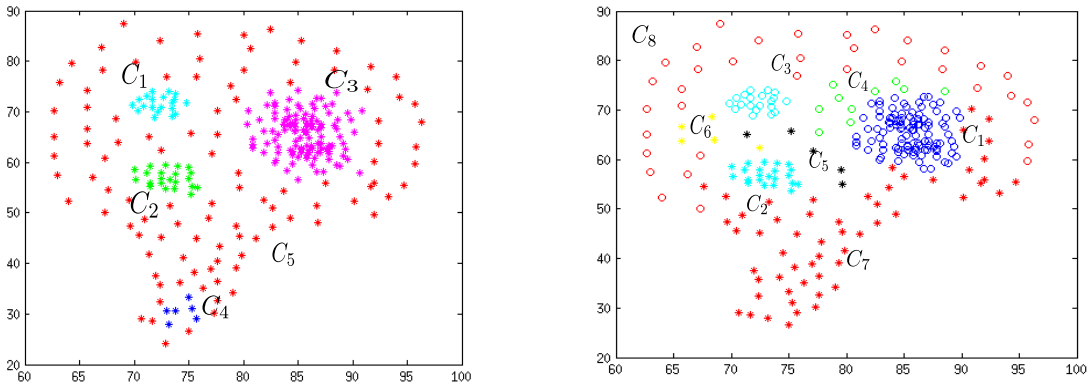
Now, we compare our method with OPTICS [6] algorithm. From a set of points, OPTICS [6] generates an ordering of points and corresponding reachability values [6]. Using the reachability plot, clusters of different densities can be obtained. Figure 8a shows the reachability plot obtained by OPTICS algorithm for Dataset-1. In this plot, x-axis displays the order in which OPTICS visits the points. Whereas, y-axis displays the reachability distance of corresponding points. Each valley represents a cluster. The deeper the valley, the more dense the cluster.

Extracting clusters can be done manually by selecting a threshold on the y-axis. We can clearly see the threshold value = 4.5 will give us a good result. Figure 8b shows the clustering result. It is clear that, clusters contain points of varying densities, but the big overlapping cluster is missing here (cluster C_5 from Figure 5a).

B. Experiment 2

In the second experiment, we use a real-world spatial dataset (Dataset-2), generated from OpenStreetMap [16], which covers a small area of Dhaka city in Bangladesh (see Figure 9a). We parsed all house or apartment locations in this region. This dataset contains 325 points. Figure 9b shows the K-DBSCAN clustering result using K = 2 (2 density levels). Our algorithm generates a total of 8 clusters. Cluster C_1 to C_7 are the most dense clusters, whereas cluster C_8 represents the lower density.

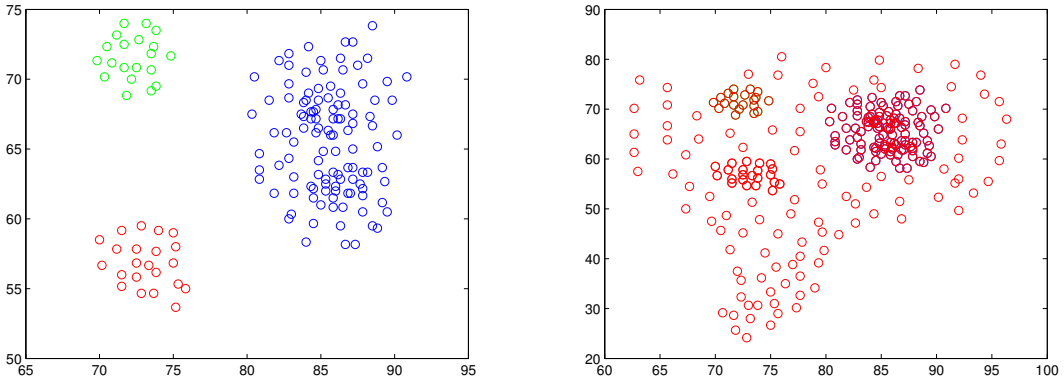
To compare with DBSCAN, we first use value ϵ = 0.004 and $MinPts$ = 5 (see Figure 9c). Points that formed cluster C_8 in our algorithm (see Figure 9b) are not being clustered using these parameters, because they are considered as outliers.



(a) K = 2

(b) K = 3

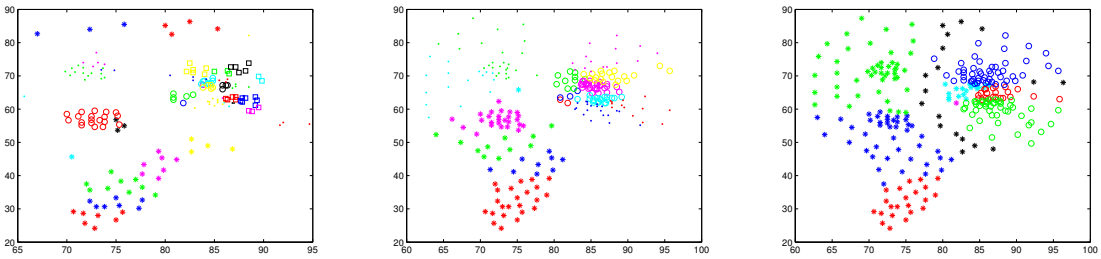
Fig. 5: K-DBSCAN results on Dataset-1



(a) ϵ = 2

(b) ϵ = 4

Fig. 6: DBSCAN results on Dataset-1, $MinPts$ = 5

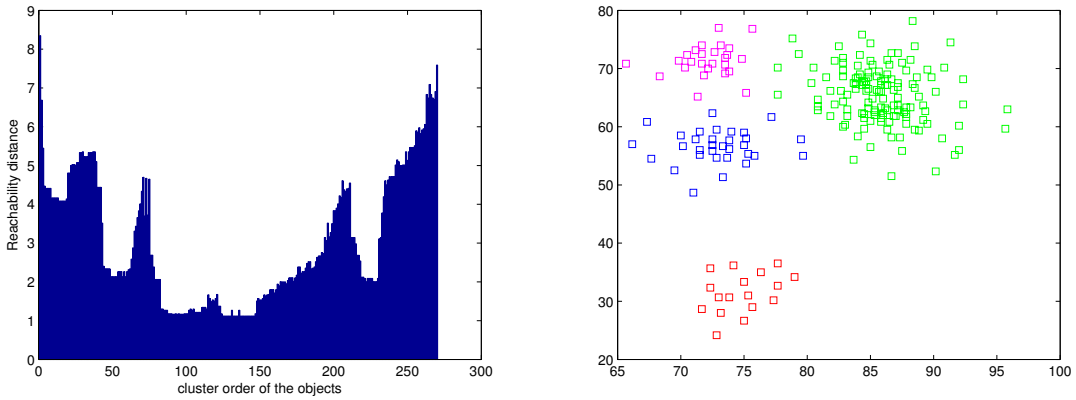


(a) k = 15

(b) k = 30

(c) k = 50

Fig. 7: SNN clustering results on Dataset-1



(a) Reachability distance

(b) OPTICS clustering result

Fig. 8: OPTICS clustering result on Dataset-1

We increase the value of ϵ to 0.009 and implement DBSCAN again (see the result in Figure 9d). Here we find the same cluster(magenta points) as cluster C_8 (Figure 9b), but clusters C_1 to C_5 merged together to form a single cluster (red points).

We implement Shared Nearest Neighbour (SNN) [11] algorithm on dataset 2. We use nearest neighbour size k = 50. Figure 9e shows the result. We get a total of 8 clusters. Points that formed cluster C_8 in our algorithm (see Figure 9b) are not being clustered in this result.

Next we implement OPTICS [6] algorithm on dataset 2. We get a total of 4 clusters. Still the points that formed cluster C_8 in our algorithm (see Figure 9b) are not being clustered in this result (see Figure 9f).

C. Qualitative Measure of Clustering Results

We attempt to measure the qualitative measure of clustering result of K-DBSCAN algorithm. Our goal is to differentiate the most dense regions from lower density regions. Motivated by this goal, we define a cluster quality measure that is based on density variation of the points in the clusters. First, we calculate the average density of all points of a cluster. Then we get the standard deviation of density values of that cluster. In addition to this, we consider the noise penalty to penalize an excessive number outliers. The formulas for these density quality measures are given in (3) - (7).

$$a_i = \frac{1}{|C_i|} \sum_{x \in C_i} density(x) \tag{3}$$

$$\sigma_i = \sqrt{\frac{\sum_{x \in C_i} (a_i - density(x))^2}{|C_i|}} \tag{4}$$

$$NoiseP = \frac{\textit{Total number of outliers}}{\textit{Total number of Points}} \tag{5}$$

$$MDV = \frac{1}{num_{cluster}} \sum_{i=1}^{num_{cluster}} \sigma_i \tag{6}$$

$$QMeasure = MDV + NoiseP \tag{7}$$

Table I compares the qualitative measure of our clustering algorithm on Dataset-1 with the other different algorithms, whereas, Table II shows the qualitative measure on Dataset-2. The lower the QMeasure value, the better the result. We can clearly see that, our algorithm performs better on both cases than all other algorithms.

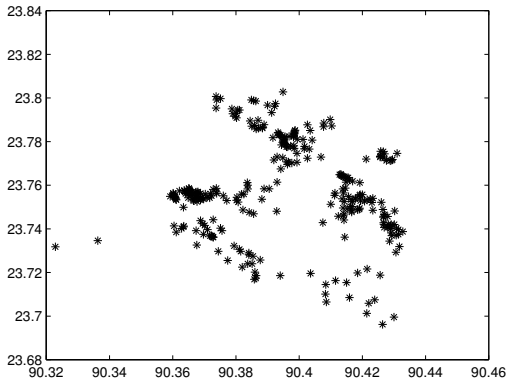
We then incorporates a measure of spatial contiguity of clusters in addition to the density and noise measures, by using SSE (Sum of squared error) [17], normalized to 0 to 1 range, (8) and (9).

$$SSE = \sum_{i=1}^{num_{cluster}} \left(\frac{1}{2|C|} \sum_{x \in C_i} \sum_{y \in C_i} dist(x,y)^2 \right) \tag{8}$$

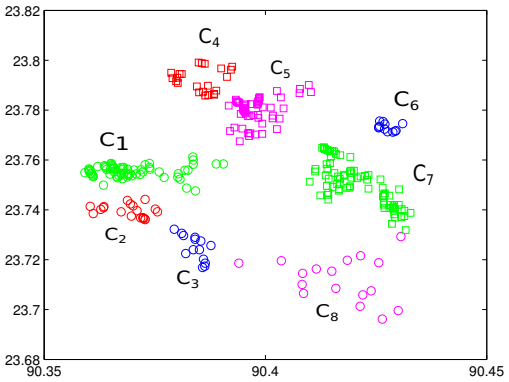
$$Total = MDV + NoiseP + SSE \tag{9}$$

Table III and IV show the results for Dataset-1 and Dataset-2 respectively. Smaller value indicates better result. Dataset-1 gives us better result for K = 3 or K = 4. With 2 density levels (K = 2), we get one big overlapping cluster (see Figure 5a). That is why we get a larger value of SSE. As the density levels increase, the big overlapping cluster is broken into multiple well separated clusters, which gives us a better result (see Figure 5b). Qualitative results of DBSCAN change dramatically for change of parameter values. Whereas, our method gives comparatively consistent results when changing the parameter value (K).

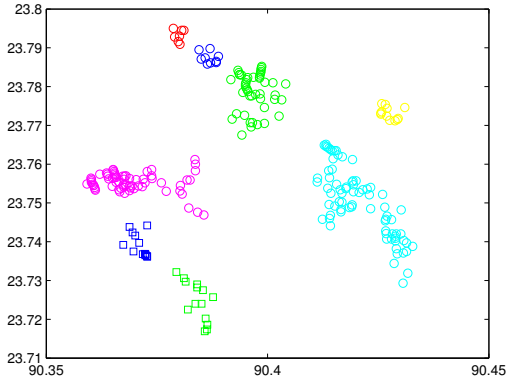
Our method gives us the best result in Dataset-2. As there are no overlapping clusters, we get 8 well separated clusters (see Figure 9b).



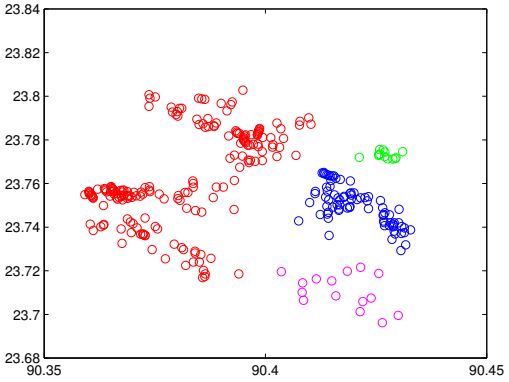
(a) Dataset-2



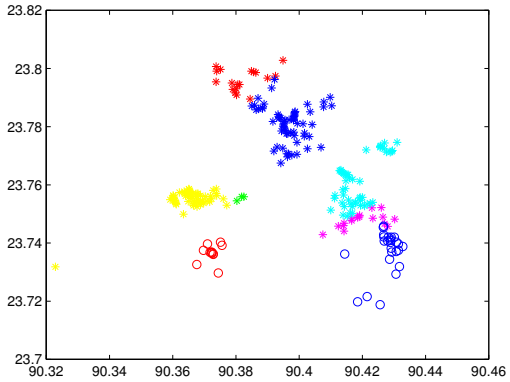
(b) K-DBSCAN, K=2



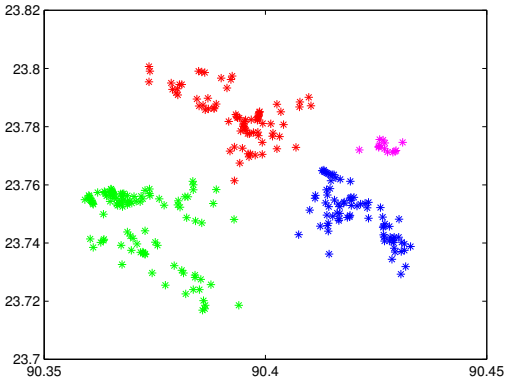
(c) DBSCAN, $\epsilon = 0.004$



(d) DBSCAN, $\epsilon = 0.009$



(e) SNN, $k = 50$



(f) OPTICS

Fig. 9: Clustering results on Dataset-2

Algorithm	MDV	NoiseP	QMeasure
K-DBSCAN (K=2)	0.3503	0.0037	0.354
K-DBSCAN (K=3)	0.3080	0.029	0.337
K-DBSCAN (K=4)	0.2451	0.085	0.330
SNN (k=15)	0.4354	0.263	0.699
SNN (k=30)	0.4966	0.103	0.597
SNN (k=50)	0.7328	0	0.733
OPTICS	0.6364	0.222	0.859
DBSCAN ($\epsilon = 0.2$)	0.2278	0.333	0.561
DBSCAN($\epsilon = 0.4$)	0.8861	0.0111	0.897

TABLE I: QMeasure for Dataset-1

Algorithm	MDV	NoiseP	QMeasure
K-DBSCAN(K=2)	0.00083	0.0565	0.0573
SNN(k=50)	0.0017	0.0598	0.0615
OPTICS	0.0011	0.664	0.675
DBSCAN($\epsilon = 0.004$)	0.0006	0.698	0.6986

TABLE II: QMeasure for Dataset-2

Algorithm	MDV	NoiseP	SSE	Total
K-DBSCAN (K=2)	0.350	0.004	0.404	0.758
K-DBSCAN (K=3)	0.308	0.029	0.299	0.636
K-DBSCAN (K=4)	0.245	0.085	0.273	0.603
SNN (k=15)	0.435	0.263	0.004	0.703
SNN (k=30)	0.497	0.103	0.0512	0.649
SNN (k=50)	0.733	0	0.0512	0.784
OPTICS	0.636	0.222	0.0570	0.916
DBSCAN ($\epsilon = 0.2$)	0.228	0.333	0.024	0.585
DBSCAN ($\epsilon = 0.4$)	0.887	0.011	0.511	1.401

TABLE III: Total Qualitative Measure for Dataset-1

Algorithm	MDV	NoiseP	SSE	Total
K-DBSCAN (K=2)	0.0008	0.057	0.020	0.078
SNN (k=50)	0.0017	0.059	0.045	0.106
OPTICS	0.0011	0.664	0.021	0.697
DBSCAN ($\epsilon = 0.004$)	0.0006	0.698	0.131	0.830

TABLE IV: Total Qualitative Measure for Dataset-2

V. Application of K-DBSCAN

In order to illustrate the practical application of our algorithm, we apply K-DBSCAN to a real-world population dataset [18]. This dataset gives the population density for Texas state for 1990. This dataset is gridded with 0.25×0.25 degree resolution. Each grid cell contains the count of the number of people inside it. For the experiments in this paper, we represent each population count of 50 persons by 1 point. Within each grid, if there were n people residing in it, we generated $n/50$ locations (latitude, longitude) so that each point represents 50 persons.

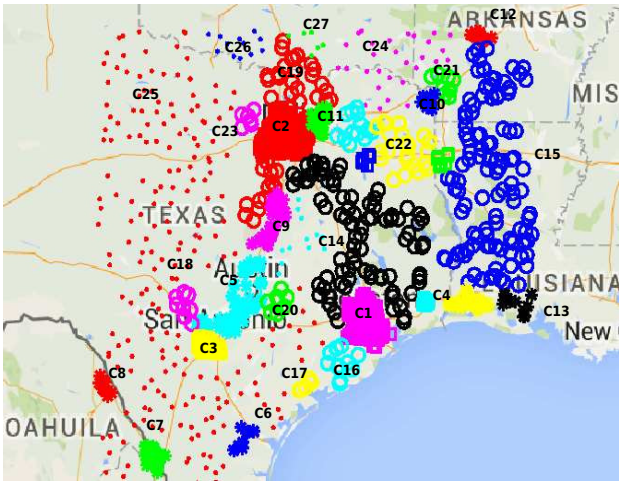


Fig. 10: Spatial Clusters based on population density

We use density level parameter $K = 4$. Figure 10 shows the result. We get total 27 clusters with 4 population density levels. 3 clusters (C_1 to C_3) among them have the highest density levels. We can see from Figure 10 that these regions are in Houston, Dallas and San-Antonio. Clusters C_4 to C_{13} have the second highest population density. Clusters C_{14} to C_{23} have the third highest density levels. Clusters C_{24} to C_{27} are the regions with lowest population density.

VI. Conclusion

In this paper, we propose a density-based clustering algorithm which can handle arbitrary shaped clusters and datasets with varying densities. We did experiments on both synthetic data and real-world spatial data. Unlike the DBSCAN algorithm, K-DBSCAN does not depend on a single global density threshold ϵ, which is difficult to determine. Rather, we have to find a threshold for density level K, which can be easily determined from the points density distribution. Experimental results show that our algorithm can correctly cluster the data points that have different density levels and different shapes. We compared the clustering quality of our algorithm with 3 other well known algorithms.

References

[1] J. Han, M. Kamber, and A. K. H. Tung, "Spatial clustering methods in data mining: A survey," in *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*. Taylor and Francis, 2001. [Online]. Available: http://www-faculty.cs.uiuc.edu/ hanj/pdf/gkdbk01.pdf

[2] E. Kolatch, "Clustering algorithms for spatial databases: A survey," *PDF is available on the Web*, 2001.

[3] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281-297. California, USA, 1967, p. 14.

[4] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, vol. 96, 1996, pp. 226–231.

[6] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *ACM SIGMOD Record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.

[7] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, vol. 98, 1998, pp. 58–65.

[8] A. P. Dempster, N. M. Laird, D. B. Rubin *et al.*, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[9] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM SIGMOD Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[10] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

[11] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data." in *SDM*. SIAM, 2003, pp. 47–58.

[12] W. Wang, J. Yang, and R. Muntz, "Sting: A statistical information grid approach to spatial data mining," in *VLDB*, vol. 97, 1997, pp. 186–195.

[13] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: a wavelet-based clustering approach for spatial data in very large databases," *The VLDB Journal*, vol. 8, no. 3-4, pp. 289–304, 2000.

[14] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*. ACM, 1998, vol. 27, no. 2.

[15] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "A local-density based spatial clustering algorithm with noise," *Information Systems*, vol. 32, no. 7, pp. 978–986, 2007.

[16] *http://www.openstreetmap.org*.

[17] J. Han and M. Kamber, "Data mining: Concepts and techniques."

[18] *http://webmap.ornl.gov*.