Journal of Information & Computational Science 9: 10 (2012) 2739–2749
Available at http://www.joics.com

# Multi-density DBSCAN Algorithm Based on Density Levels Partitioning

Zhongyang Xiong, Ruotian Chen*, Yufang Zhang, Xuan Zhang

*College of Computer Science, Chongqing University, Chongqing 400044, China*

**Abstract**

DBSCAN is a typical density-based clustering algorithm, it has an advantage of discovering clusters of different shapes and sizes along with detection of outliers. However, the parameter Eps and MinPts are hard to determine but directly influence the clustering result. Furthermore, the adoption of global parameters makes it an unsuitable one for datasets with varied densities. To address these problems, this paper proposes a multi-density clustering method called DBSCAN-DLP (Multi-density DBSCAN based on Density Levels Partitioning). DBSCAN-DLP partitions a dataset into different density level sets by analyzing the statistical characteristics of its density variation, and then estimates Eps for each density level set, finally adopts DBSCAN clustering on each density level set with corresponding Eps to get clustering results. Extensive theoretical analysis and experimental results on both synthetic and real-world datasets confirm that proposed algorithm is efficient in clustering multi-density datasets.

*Keywords*: Density-based Clustering; DBSCAN; Multi-density; Density Levels; Statistical Characteristics

## 1 Introduction

Data clustering is an important and extremely challenging problem. It groups data into meaningful subclasses such that the points in each subclass have high intra-class similarity and low inter-class similarity [1]. As an attractive research area in data mining, clustering is extensively used in varieties of applications such as pattern recognition, image processing, business intelligence etc.

With exponential growth of data scale and the enrichment of data types, some new requirements [2] have been put forward on clustering algorithms: scalability, noise handling, dealing with multi-dimensional data, ability of discovering clusters with arbitrary shapes, minimum dependence of domain knowledge to determine input parameters.

The density-based clustering methodology satisfies some of the above requirements. They can find arbitrarily shaped and differently sized clusters, which are considered to be the dense regions

*Corresponding author.
*Email address:* `crt_310@163.com` (Ruotian Chen).

separated by low-density regions. Density-based clustering requires no prior information regarding the number of clusters. Besides, noise is well handled.

The DBSCAN [3] algorithm is one of the most popular density-based clustering techniquesit carries forward all advantages of density-based clustering family. In the meanwhile, due to adoption of global parameters, it fails to identify clusters with varied densities unless the clusters are clearly separated by sparse regions.

In this paper, we propose a new clustering algorithm which generalizes DBSCAN to discover clusters of different densities. It automatically estimates parameters for each cluster so as to reduce the requirement for human intervention or domain knowledge. The experimental results show the superiority of this algorithm.

The rest of the paper is organized as follows: Section 2 reviews DBSCAN algorithm and presents related work on multi-density clustering. Section 3 firstly presents the required definitions and notations, and then states the details of proposed algorithm. Experimental results are given in Section 4. At the end, we summarize the conclusions and future work in Section 5.

## 2 DBSCAN Algorithm and Related Work

### 2.1 Introduction to DBSCAN Algorithm

DBSCAN is a classic density-based clustering algorithm, it groups data points which are sufficiently dense into clusters, the discovery process is based on the fact that a cluster can be expanded by any of its core objects. In DBSCAN, the density associated with a point is obtained by counting the number of points in a region of specified radius called Eps around this point. Points with a density above a specified threshold called MinPts are identified as core points.

The discovery process starts from an arbitrary point, if it's a core point, the neighborhood query [4] recursively continues and stops at the border points, then another arbitrary ungrouped object is selected and this process is repeated until all data points in the dataset have been placed in clusters or labeled as noise.

DBSCAN doesn't need the number of final clusters to be given in advance. Besides, by applying neighborhood expansion, DBSCAN can accurately recognize clusters of arbitrary shape and different size along with noise filtering. However, DBSCAN also has following three main flaws [2, 5]: (1) It's in high demand for memory and I/O resource especially when input dataset is enormously large, and a significant amount of time is consumed in iterative neighborhood query; (2) It has the limitation of parameter tuning because the crucial parameters which directly influence the clustering result must be specified by users; (3) It's unable to deal with varied densities because of the adoption of global parameters.

### 2.2 Existing Extended Work

In view of the fact that DBSCAN doesn't perform well under multi-density circumstance and requires much subjective intervention in the parameter estimation, many improvements have been proposed in recent years.

Early efforts like OPTICS [6] (Ordering Points to Identify the Clustering Structure) is an enhanced method upon DBSCAN. OPTICS creates an ordering of the objects augmented by reachability distance and makes a reachability-plot out of this ordering. The reachability-plot, which contains information about the intrinsic clustering structure, is the basis for interactive cluster analysis, but it doesn't produce the clustering result explicitly.

To reduce I/O cost, The PDBSCAN [7] (Data-partitioning-based DBSCAN) extends DBSCAN by dataset partitioning to reduce the search space of the neighborhood query. For each partition, local DBSCAN is adopted with corresponding Eps, which is chosen from the kdist plot of this partition. And then a merging process is required to get the final clusters. PDBSCAN not only reduces memory and I/O cost but also has the ability to deal with multi-density datasets. Nevertheless, this algorithm needs much human intervention in dataset partitioning and it fails to deal with ring-shaped datasets.

To determine appropriate parameters for DBSCAN, an interactive way seems effective. The GADAC [8] (A new density-based scheme for clustering based on genetic algorithm) adopts diverse radiuses which are adjusted by a genetic algorithm to expand clusters of varied densities. GADAC produces more precise clustering results than DBSCAN. The VDBSCAN [9] (Varied Density Based Spatial Clustering of Applications with Noise), computes kdist value for each object and sort them in ascending order, then make a visualization of the sorted values. The sharp changes in the kdist plot correspond to a list of radiuses for different density varied clusters.

To estimate parameters automatically, a method, called KDDClus [10], is designed to identify clusters of varied densities automatically. It estimates the density of a pattern by averaging the distances of all its k nearest neighbors, and uses 1-dimension clustering on these density values to get a partition of different levels. Then, the KDDClus obtains radiuses for different densities, with which DBSCAN is adopted to find out clusters of varied densities. Nevertheless, this method brings in the instinct defects of the clustering method for reason that the input parameters for 1- dimension clustering is still a hard nut to crack. For the same purpose, another improvement of the DBSCAN, proposed by HuanLiang Sun [11], uses 1inear regression technique to find the sharp changes of kdist plot for a multi-density dataset, but the parameter Mindis is much data-dependent. In addition, both two methods above have trouble in distinguishing noise.

Although the clustering techniques stated above can deal with clusters of different densities to some extent, the goal of less subjectivity and more intelligence still remains challenging. To achieve this goal, we attempt to provide such a solution via DBSCAN-DLP.

## 3 Proposed Algorithm

In this paper, we propose DBSCAN-DLP algorithm which attempts to generalize the typical DBSCAN algorithm to automatically discover clusters of different densities via density level partitioning. To start with, some basic definitions are required:

**Definition 1** *k nearest neighbor distance [12]. The k nearest neighbor distance of a point p, denoted by kdist* $(p, k)$*, is defined as the distance between p and its k-th nearest neighbor o, where k is an arbitrary positive integer, always selected from range [3, 10].*

**Definition 2** *k neighborhood density. The k neighborhood density (for short density) of p, denoted* *by $Den(p,k)$, is defined as follows:*

$$Den(p,k) = \frac{k}{kdist(p,k)}. \tag{1}$$

where $k$ defines the number of $p$'s neighbors. Given a fixed $k$, the smaller $kdist\,(p,k)$ is, the denser $p$ is.

**Definition 3** *density variation. The density variation of point $p_i$ with respect to $p_j$, denoted by $DenVar(p_i,p_j)$, is defined as how much denser or sparser $p_i$ than $p_j$. It's computed as follows:*

$$\begin{aligned} DenVar(p_i,p_j) &= \frac{|Den(p_i,k) - Den(p_j,k)|}{Den(p_j,k)} \\ &= \frac{|kdist(p_j,k) - kdist(p_i,k)|}{kdist(p_i,k)}. \end{aligned} \tag{2}$$

According to Eq. (1) and Eq. (2), we can see that the *kdist* value of a certain point can indicate its density appropriately. Thus, for simple calculation, rest definitions use term *kdist* instead of density. Note that all k in our definitions have the same value, which is specified by users.

**Definition 4** *density level set. A density level set (for short DLS) consists of points whose densities are approximately the same. In other words, the density variations of data points within the same DLS should be relatively small. Point $p_i$ and $p_j$ belong to the same DLS if they satisfy the following condition:*

$$p_i, p_j \in DLS_n \quad iff \quad DenVar(p_i,p_j) \le \tau.$$

where $\tau$ is a density variation threshold. A multi-density dataset can be divided into several density level sets, each of which stands for a density distribution.

**Definition 5** *inter-level density gradient. Assumed $DLS_i$ has a higher density level than $DLS_j$, the inter-level density gradient between $DLS_i$ and $DLS_j$, denoted by $DenGrad\,(DLS_i, DLS_j)$, presents how much $DLS_i$ is denser than $DLS_j$. It's computed as follows:*

$$DenGrad(DLS_i, DLS_j) = \frac{meankdist(DLS_j)}{meankdist(DLS_i)} - 1. \tag{3}$$

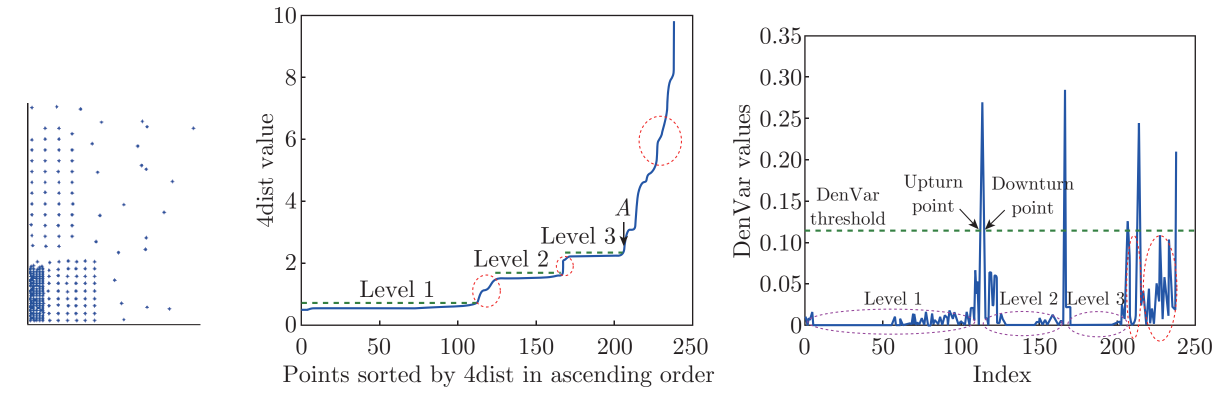
where $meankdist(DLS_j)$ is the mean *kdist* value of all objects in $DLS_j$.

**Definition 6** *intra-level density scatterness. The intra-level density scatterness of $DLS_i$, denoted by $Scatterness\,(DLS_i)$, indicates the uniformity of local density distribution within $DLS_i$. The DLS, which obtains more uniform local distribution, gets smaller scatterness. Scatterness is computed as follows:*

$$Scatterness(DLS_i) = \frac{1}{\sqrt{\|DLS_i\|}} \cdot \frac{SD(DLS_i)}{meankdist(DLS_i)}. \tag{4}$$

where $\|DLS_i\|$ is the size of $DLS_i$, $SD\,(DLS_i)$ is the Standard Deviation (SD) of *kdist* values in $DLS_i$. Scatterness is defined based on coefficient of variability (CV) (refer to the right multiplier in Eq. (4)), which is always used to compare the dispersion of two same sized sample sets. We use the size of a *DLS* to tune its CV value, because the SD value of a bigger sized but more even-distributed *DLS*, may be bigger than a smaller sized but less even-distributed *DLS*. Thus, Scatterness is appropriately estimated with the tuning part.

To illustrate our algorithm easily, we construct a noisy sample dataset(as shown in Fig. 1 (a)) which contains three clusters of different densities. Compute the *kdist* (select $k = 4$) value for each point of this dataset, sort them in ascending order, and plot them out as shown in Fig. 1 (b). From Fig. 1 (b), it can be seen that there are three relatively smooth lines which accordingly represent three density levels. For noise or border points, the corresponding line (circled in red) rockets, it's called level-turning line [9]. DBSCAN chooses the *kdist* value of point A as global *Eps*, however, it results in merging of the bottommost two clusters. Apparently, a solution can be: interactively choose corresponding Eps for each cluster.



(a) A sample dataset (b) Kdist plot of sample dataset (c) DenV ar plot of sample datset

Fig. 1: *kdist* and *DenVar* plot of sample dataset

For datasets with widely varied densities, there will be some distinct variation, depending on the densities of the clusters, but for points of the same density level, the range of variation may not be huge while a sharp change is expected to see between two density levels. Thus, we deduce that we can acquire all density level sets by detecting these sharp changes of density. Besides, some clusters may be involved in the same density level set, so a local clustering process on each density level set is needed to get final clusters.

The proposed algorithm is divided into following main steps:

a. Partition input dataset into different density level sets and refine these sets.

b. Estimate *Eps* value for each density level set.

c. Adopt DBSCAN clustering on each density level set with corresponding *Eps*.

## 3.1 Partitioning Density Levels

Density levels partitioning is a key step in proposed algorithm. Here, we try to implement this by investigating the statistical characteristics of density variation.

To assist our statement, we construct a plot of density variation values (for short *DenVar* plot): select a fixed $k$, compute $kdist$ value for each point, and sort them in ascending order. With sorted $kdist$ list (denoted by $kdistList$), assume its size is $n$, for each sequence-adjacent $kdist_i$ and $kdist_{i+1}$ in $kdistList$, compute $DenVar_i = (kdist_{i+1} - kdist_i)/kdist_i$ (refer to Eq. (2)), then we get $DenVar$ list (denoted by $DenVarList$) of size $n-1$. Note that, each element in $DenVarList$ corresponds to two data points in input dataset. Plot $DenVarList$ out, we get the $DenVar$ plot (as shown in Fig. 1 (b)).

In the $DenVar$ plot, there are some sharp waves between two relatively smooth lines. It can be seen that the smooth lines and sharp waves in $DenVar$ plot correspond to the density level lines and level-turning lines in $kdist$ plot respectively, i.e. a smooth line represents a density level set and a sharp wave indicates a sharp change of two density levels connected by this wave. In order to get these density level sets, each smooth line should be separated out, a partitioning method can be: using a horizontal straight line to split $DenVar$ plot into several small parts, and the parts lying lower than this straight line can be considered as density level sets.

Based on above analysis, we implement partitioning method on $DenVarList$ directly, but not on $DenVar$ plot. Given a density variation threshold $\tau$ (refer to *Definition 4*), remove $DenVar$ values which is bigger than $\tau$ out of $DenVarList$, then the underlying points of remaining separated segments are considered as different density level sets. Nevertheless, threshold is hard to determine without any prior knowledge. Here, we specify $\tau$ according to the statistical characteristics of the $DenVarList$, it's computed as follows:

$$\tau = EX(DenVarList) + \omega \cdot SD(DenVarList). \tag{5}$$

where EX is mathematical expectation, SD is Standard Deviation, $\omega$ is a tuning coefficient. According to the $DenVar$ plot, it can be seen that there are only a small part of points which result in sharp waves, so the smooth lines fluctuate slightly around the value EX ($DenVarList$). Thus, we just need an extra positive tuning part on the basis of EX ($DenVarList$) to ensure effective partitioning. In terms of probability theory and statistics [13], Eq. (5) can estimate a suitable $\tau$. According to large numbers of experiments, for most datasets, $\omega$ can be chosen from range (0, 3]. Especially, for multi-density datasets, $\omega = 2.5$ is an ideal value.

After density levels partitioning, we acquire a list of density level sets (for short $DLSList$). However, the sets consisting of noise and border points should be removed out of $DLSList$; the sets which almost have the same density level should be merged, refinements are required as follows:

a. Removal process: According to *Definition 6*, the sets which consist of noise and border points have few members or huge $Scatterness$ values. So we remove the set whose member size is less than 0.6% (empirical value) of the whole dataset or its $Scatterness$ value is bigger than 0.012 (empirical value).

b. Merging process: In partitioning process, some levels may be separated into smaller parts if the threshold is too small. So we merge each two sets between which the $DenGrad$ (refer to *Definition 5*) value is less than 0.2 (empirical value).

## 3.2 Estimating Parameters

In this step, our object is to find representative Eps for each density level set. For a certain $DLS$, its corresponding Eps will be magnified by simply choosing the maximum $kdist$. To deal with this problem, we compute $Eps_i$ for $DLS_i$ as follows:

$$Eps_i = maxkdist(DLS_i) \cdot \sqrt{\frac{mediankdist(DLS_i)}{meankdist(DLS_i)}}. \tag{6}$$

where $maxkdist$, $meankdist$ and $mediankdist$ are the maximum, mean and the median $kdist$ of $DLS_i$ respectively.

As we know, for a density level set, the end-elements may correspond to border objects or noise, they have some influence on the mean $kdist$, and then the median $kdist$ is always smaller than mean $kdist$. According to Eq. (6), $Eps$ is appropriately tuned compared to the unsuitable max $kdist$. In this manner, we get a list of $Eps$ (for short $EpsList$).

## 3.3 Forming Clusters

Table 1: DBSCAN-DLP algorithm

```
Algorithm: DBSCAN-DLP.
Input: k, dataset D
Procedure:
Begin
1) Initialize (D);    // Initialize all objects in D as unlabeled and unprocessed.
2) distMat = CompDistMat(D);    // Get distance matrix.
3) kdistList = CompKDsit(k, D);    //Get k nearest neighbor distances.
4) SortInAsc(kdistList);    //Sort kdistList in ascending order.
5) DenVarList = CompDenVar(kdistList);    // Get density variation values list.
6) = EX(DenVarList) + SD(DenVarList);    // Compute density variation threshold.
7) DLSList = PartitionDL(DenVarList, D);    // Partition D into a list of density level sets.
8) RefineDLS(DLSList);    //Refine DLSList via Removal Process and Merging Process.
9) EpsList = EstimateEPS(DLSList);    //Estimate Eps for each DLS.
10) For each Epsi EpsList,
    DBSCAN(k, Epsi, DLSi, D);    //Adopt DBSCAN on each $DLS_i$ with $Eps_i$
    End iteration;
11) Return all clustering results.
End
```

With density levels partitioning and parameters estimation done, we carry on the clustering process: initialize global parameter $MinPts = k$, adopt DBSCAN for each $Eps_i$ in $EpsList$ (already in ascending ordered according to previous steps) on corresponding $DLS_i$. During each iteration, discovery process starts only from the objects within $DLS_i$ rather than the whole input dataset, but neighborhood counting use all unprocessed objects in whole dataset. It means the initial seeds are only from $DLS_i$, but unprocessed objects in input dataset, e.g. some border objects, do have the chance to be neighborhood expanded by these seeds. In this case, the frequency of neighborhood query has been reduced compared to checking all objects in dataset for each $Eps_i$ [5]. Finally, non-marked points after all iterations are recognized as noise. Combine clusters discovered in each iteration, we get the final clusters.

The pseudocode of DBSCAN-DLP algorithm is presented in Table 1. Note that *kdist* plot and *DenVar* plot are not parts of our algorithm, they are just assistant analysis tools.

### 3.4 Computational Complexity

The proposed algorithm contains three main steps, most time-consuming steps are the first and third step. Assume there are n objects in dataset D. In the first step, the calculation of *kdist* values leads to a time complexity of $O(nlogn)$ via using a spatial access method, such as R*-tree [5, 14]; the ordering of *kdist* values consumes $O(nlogn)$; the other processes have the same runtime complexity of $O(n)$. In the third step, the clustering process adopts DBSCAN, whose time required for neighborhood query of a single object is $O(logn)$ with R*-tree structure used, neighborhood query is performed for objects in all $DLS$, so the runtime complexity of forming clusters is $O(mlogn)$, $m(\leq n)$ is the total number of objects in all $DLS$.

Totally, the runtime complexity of DBSCAN-DLP is $O(nlogn)$, which is the same as DBSCAN.

## 4 Experiments and Analysis

In this section, we evaluate the performance of DBSCAN-DLP on both synthetic and real-world datasets and compare the results with DBSCAN. We implement the algorithm in C++. Experiments are conducted on a 2.1 GHz laptop with core dual processor, 2048 MB RAM.

**Experiment I.** Performance on synthetic datasets

For intuitive illustration, the synthetic datasets are all in 2-dimensional space. As shown in Fig. 2, the first dataset(for short DS1) is a multi-density dataset used in article [15], which contains 3147 data points and 4 clusters of different densities, sizes; DS2 is also a multi-density dataset used in SNN [16], which contains 3873 data points and 6 clusters of different shapes and sizes from 3 density levels; DS3 is a noisy single-density dataset called t4.8k [17], it's one of the famous Chameleon algorithm datasets, which contains 8000 data points and 6 clusters of different shapes.

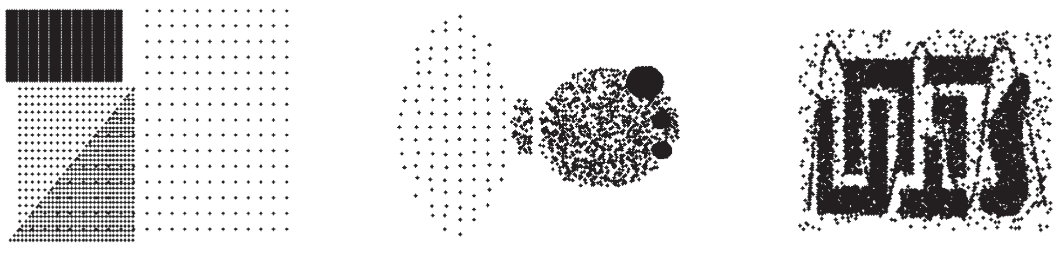

(a) DS1 (b) DS2 (c) DS3

Fig. 2: Datasets

It can be seen from Fig. 2 (a) (b), a particularly challenging feature of DS1 and DS2 is that they are both density-connected or density-overlapping.

Fig. 3 and Fig. 4 respectively show the comparative clustering results between DBSCAN-DLP and DBSCAN on DS1 and DS2. Different symbols are used to indicate different clusters, the same

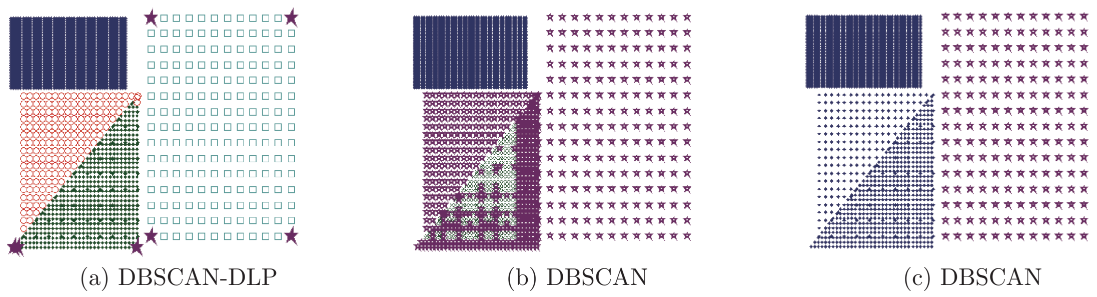

(a) DBSCAN-DLP (b) DBSCAN (c) DBSCAN

Fig. 3: Clustering results on DS1: (a) $k = 4, \omega = 2.5$; (b) $MinPts = 4, Eps = 0.3$; (c) $MinPts = 4, Eps = 0.7$ (noise is symbolized by star)

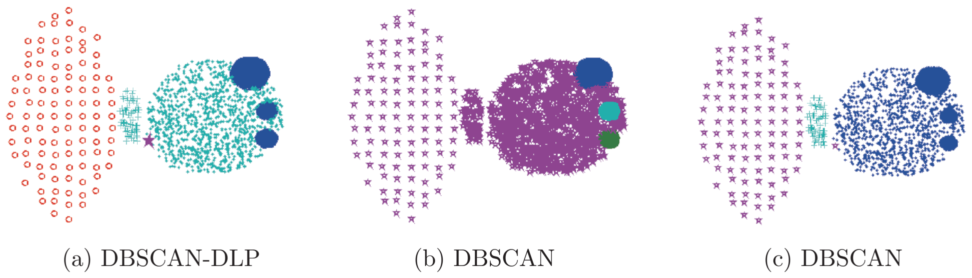

(a) DBSCAN-DLP (b) DBSCAN (c) DBSCAN

Fig. 4: Clustering results on DS2: (a) $k = 4, \omega = 2.5$; (b) $MinPts = 4, Eps = 0.05$; (c) $MinPts = 4, Eps = 0.22$ (noise is symbolized by star)

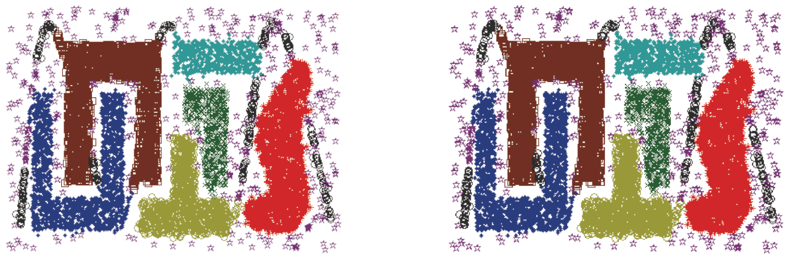
colors are used to indicate the same density levels, note that the term density level is meaningful only to proposed algorithm.

In Fig. 3 (a), DBSCAN-DLP detects 4 density level sets, corresponding to 4 *Eps*: 0.2, 0.3, 0.51 and 1, with these *Eps*, 4 clusters are discovered. Especially, the two triangle-shaped density-connected clusters are clearly separated into two clusters by our algorithm. In Fig. 4 (a), DBSCAN-DLP discovers 3 density levels, corresponding *Eps* are: 0.048, 0.22 and 0.56, then 6 clusters are found appropriately. Especially, the rightmost three smaller clusters are successfully distinguished from the bigger overlapping ellipse-shaped cluster. However, both in Fig. 3 and Fig. 4, DBSCAN fails to detect clusters correctly, a relatively small *Eps* makes some clusters identified as outliers as shown in Fig. 3 (b) and Fig. 4 (b); a relatively big *Eps* makes some clusters merged into a single cluster as shown in Fig. 3 (c) and Fig. 4 (c). According to the experimental results on DS1 and DS2, we can deduce that proposed DBSCAN-DLP is able to discover clusters of different densities, shapes and sizes efficiently even if they are density-connected or density-overlapping.

We also conduct experiments on the single-density dataset DS3, clustering results are depicted in Fig. 5, proposed algorithm detect a single density level with corresponding $Eps = 7.46$. From this test, we can see both proposed algorithm and DBSCAN can appropriately discover clusters in single-density noisy datasets.

**Experiment II.** Performance on real-word datasets

In order to further confirm our conclusions in Experiment I, we evaluate the performance of proposed algorithm and DBSCAN on two real-word datasets called Iris and Wine, they are both from the famous UCI Machine Learning Repository. Rand-Index [18] is used to compare the clustering outputs of the two methods.



(a) DBSCAN-DLP: $k$=6; $\omega$=1:2 (b) DBSCAN: $MinPts$=6; $Eps$=7:5

Fig. 5: Clustering results on DS3 (noise is symbolized by star)

Table 2 gives the comparison accuracy of the two algorithms. Good accuracy of experiments on real-world data also shows that DBSCAN-DLP has good performance in identifying clusters from real datasets.

Table 2: Comparison of accuracy

| Dataset | Algorithm | Clusters | Attributes | Parameters | Rand_Index(%) |
|---|---|---|---|---|---|
| Iris | DBSCAN-DLP | 3 | 4 | $k = 4, \omega = 0.5$ | 84.1 |
| | DBSCAN | | | $MinPts = 4, Eps = 0.13$ | 76.1 |
| Wine | DBSCAN-DLP | 3 | 13 | $k = 4, \omega = 1$ | 72.3 |
| | DBSCAN | | | $MinPts = 4, Eps = 0.49$ | 73.1 |

## 5 Conclusions

In this article, proposed DBSCAN-DLP algorithm tries to achieve multi-density clustering via density levels partitioning according to the statistical characteristics of density variation. Excellent performance on both synthetic and real-word datasets confirms its effectiveness. However, proposed algorithm needs k nearest neighbor distances and density variation values computed, stored as well. It's time and I/O consuming when input dataset is enormously large. To attack this problem, we can adopt sampling techniques before density levels partitioning for future research.

## References

[1] Xutao Li, Yunming Ye, Mark Junjie Li, Michael K. Ng, On cluster tree for nested and multi-density data clustering, Pattern Recognition, 43(9), 2010, 3130-3143

[2] Xuegang Hu, Dongbo Wang, Xindong Wu, Varying density spatial clustering based on a hierarchical tree, in: Proc. 5th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'2007), Leipzig Germany, 18(20), 2007, 188-202

[3] M. Ester, H. P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proc. 2nd International Conference on Knowledge Discovery and Data Mining, Portland OR, AAAI Press, 2006, 912-915

[4] P. Viswanath, V. Suresh Babu, Rough-DBSCAN: A fast hybrid density based clustering method for large data sets, Pattern Recognition Letters, 30(16), 2009, 1477-1488

[5] Yangqiang Yu, Tianqiang Huang, Gongde Guo, Semi-supervised clustering algorithm for multi-density and complex shape dataset, in: Proc. 2008 Chinese Conference on Pattern Recognition (2008), IEEE Press, 2008, 30-35

[6] M. Ankerst, M. Bruenig, H. P. Kreigel, J. Sander, OPTICS: Ordering points to identify the clustering structure, in: Proc. Int. Conf. Management of Data (SIGMOD'99), Philadelphia, PA, 1999, 49-60

[7] Shuigeng Zhou, Aoying Zhou, Jing Cao, A data-partitioning-based DBSCAN algorithm, Journal of Computer Research and Development, 37(10), 2000, 1153-1159

[8] Chihyang Lin, Chinchen Chang, A new density-based scheme for clustering based on genetic algorithm, Fundamenta Informaticae, 68(4), 2005, 315-331

[9] Peng Liu, Dong Zhou, Naijun Wu, Varied density based spatial clustering of applications with noise, in: Proc. 2007 International Conference on Service Systems and Service Management, Changdu, China, 2007, 528-531

[10] Sushmita Mitra, Jay Nandy, KDDClus: A simple method for multi-density clustering, in: Proc. 2011 International Workshop on Soft Computing Applications and Knowledge Discovery (SCAKD'2011), Moscow, 2011, 72-76

[11] Huanliang Sun, Zhanju Bi, Junlin Liu, A density-based clustering algorithm for multi-level density, Journal of Shenyang Jianzhu University (Natural Science), 22(2), 2006, 329-333

[12] Lian Duan, Lida Xu, Feng Guo, A local-density based spatial clustering algorithm with noise, Information Systems, 32(7), 2007, 978-986

[13] Shihong Yue, Ping Li, Jidong Guo, Shuigeng Zhou, Using greedy algorithm: DBSCAN revisited II, Journal of Zhejiang University SCIENCE, 5(11), 2004, 1405-1412

[14] B. Borah, D. K. Bhattacharyya, DDSC: A density differentiated spatial clustering technique, Journal of Computers, 3(2), 2008, 72-79

[15] Ahmed Fahim, Abd-Elbadeeh Scalem, Fawzy Torkey, Scalable varied density clustering algorithm for large datasets, Journal of Software Engineering & Applications, 3(6), 2010, 539-602

[16] L. Ertoz, M. Steinbach, V. Kumar, Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data, in: Proc. 2nd SIAM International Conference on Data Mining, San Francisco, 2003, 1-12

[17] G. Karypis, E. H. Han, V. Kumar, CHAMELEON: A hierarchical clustering algorithm using dynamic modeling, Computer, 32(8), 1999, 68-75

[18] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association, 66(336), 1971, 846-850