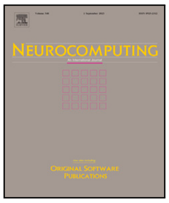
Neurocomputing 576 (2024) 127329

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



MDBSCAN: A multi-density DBSCAN based on relative density

Jiaxin Qian[a], You Zhou[b], Xuming Han[c,1], Yizhang Wang[a,d,1,*]

[a] Yangzhou University, Yangzhou, 225009, China
[b] Jilin University, Changchun, 130015, China
[c] Jinan University, Guangzhou, 510632, China
[d] Institute of Scientific and Technical Information of China, Beijing, 100038, China

ARTICLE INFO

Communicated by R. Vimieiro

Dataset link: https://github.com/mlyizhang/Clustering-Datasets

Keywords:
Relative density
DBSCAN
Multiple density clustering

ABSTRACT

DBSCAN is a widely used clustering algorithm based on density metrics that can efficiently identify clusters with uniform density. However, if the densities of different clusters are varying, the corresponding clustering results may be not good. To address this issue, we propose a multi-density DBSCAN based on the relative density (MDBSCAN), which can achieve better results on clusters with multiple densities. The intuition of our work is simple but effective, we first divide the dataset into two parts: low density and high density, and then we take a divide and conquer method to deal with the respective parts to avoid them interfering with each other. Specifically, the proposed MDBSCAN consists of three steps: (i) extract the low-density data points in the dataset by relative density. (ii) find natural clusters among the identified low-density data points. (iii) clustering the remaining data points (except the data points of natural clusters in a dataset) by using DBSCAN and assigning the noises (generated by DBSCAN) to the nearest clusters. To verify the effectiveness of the proposed MDBSCAN algorithm, we conduct experiments on ten synthetic datasets and six real-world datasets. Experimental results demonstrate that the proposed MDBSCAN algorithm outperforms the original DBSCAN and six extends of DBSCAN, especially including two state-of-the-art algorithms (DRL-DBSCAN and AMD-DBSCAN) in most cases.

1. Introduction

Clustering is a common technique in machine learning and data analysis [1], it divides the objects in the dataset into different clusters, as a result, the objects in the same cluster are as similar as possible, and the objects between different clusters are as different as possible. According to the characteristics of existing clustering algorithms, they can be divided into the following categories: hierarchical-based [2,3], partition-based [4], density-based [5–7], graph-based [8–11] [12], *etc*. Among them, density-based clustering algorithms can dynamically determine the number of clusters based on the density change of the dataset without necessarily specifying the number of clusters beforehand. In addition, density-based algorithms can identify clusters with arbitrary shapes, sizes, and densities.

DBSCAN [6] is a typical density-based clustering algorithm that has been widely used in image processing [13,14], bioinformatics [15], and other fields. It can help identify outliers, discover the patterns of datasets, *etc*. However, DBSCAN may also have the following problems in practical applications: the parameters need to be manually adjusted, and the values of two user-defined parameters (distance threshold Eps, and the minimum number of points required to form a dense region *MinPts*) have a great impact on the clustering results [16,17]. The time complexity of the DBSCAN algorithm is $\mathcal{O}(n^2)$, so it is challenging for DBSCAN to handle big data.

Among the above problems, an important problem is that DBSCAN is very sensitive to the density change of data distribution [18,19]. For a dataset with different densities, it is not easy to find global parameter values (Eps and *MinPts*) to identify clusters with different densities because data points with different densities are relatively close in Euclidean space. Specifically, if Eps is set too large, the high-density clusters will contain the low-density data points (see Fig. 1(b)), otherwise, the low-density clusters will be ignored (see Fig. 1(c)).

The intuition of our work is simple, if we divide the dataset into two parts: low density and high density, then we take a divide and conquer method to deal with the respective parts to avoid them interfering with each other, the clustering results will be significantly improved.

According to the above idea, we propose a multi-density DBSCAN based on the relative density (MDBSCAN). The process of MDBSCAN mainly consists of three steps: (i) extract the low-density data points in the dataset. (ii) find natural clusters among the identified low-density

* Correspondence to: College of Information Engineering, Yangzhou University, Yangzhou, 225009, China.
E-mail addresses: hanxuming@jnu.edu.cn (X. Han), wyzhang_new@sina.com (Y. Wang).

1 Xuming Han and Yizhang Wang contributed equally.

https://doi.org/10.1016/j.neucom.2024.127329
Received 25 October 2023; Received in revised form 6 January 2024; Accepted 21 January 2024
Available online 26 January 2024

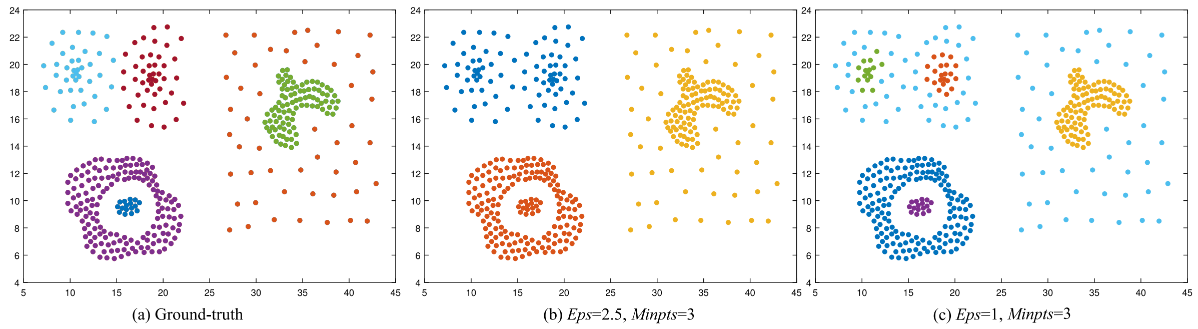


Fig. 1. Clustering results on dataset Compound by using DBSCAN: (a) the ground-truth of data Compound. (b) $Eps = 2.5, Minpts = 3$, the high-density clusters will contain the low-density data points (on the left). (c) $Eps = 1, Minpts = 3$, the low-density clusters will be ignored (on the right).

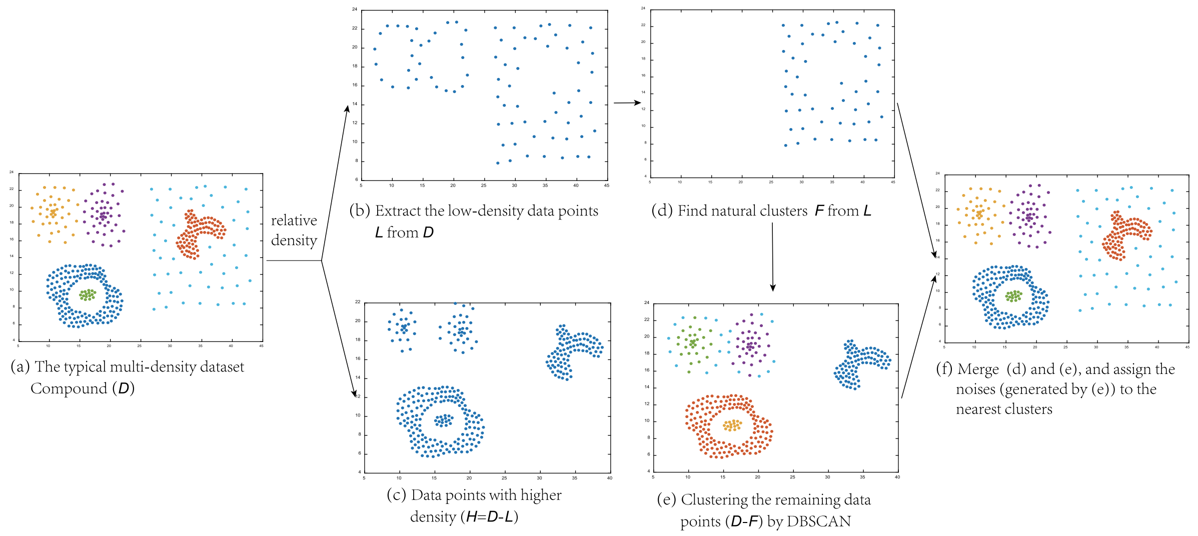

relative density
(a) The typical multi-density dataset Compound (D)
(b) Extract the low-density data points L from D
(c) Data points with higher density (H=D-L)
(d) Find natural clusters F from L
(e) Clustering the remaining data points (D-F) by DBSCAN
(f) Merge (d) and (e), and assign the noises (generated by (e)) to the nearest clusters

Fig. 2. Illustration on how MDBSCAN handles clusters with multiple densities.

data points. (iii) clustering the remaining data points (except data points of natural clusters in a dataset) by using DBSCAN and assigning the noises (generated by DBSCAN) to the nearest clusters. To further understand our idea easily, we give an example of how MDBSCAN handles clusters with multiple densities as shown in Fig. 2.

The main contributions of this paper are as follows:

(I) The concept of relative density is introduced to divide the dataset into two parts: low density and high density. Clustering the data points from two parts respectively is better than simultaneously clustering all the data points with mixed densities.

(II) We propose a new Share Nearest Neighbors-based Clustering method (SNNC) to efficiently identify natural clusters among low-density data points.

(III) The proposed MDBSCAN algorithm outperforms the original DBSCAN and six extends of DBSCAN, especially including two state-of-the-art algorithms (DRL-DBSCAN [20] and AMD-DBSCAN [21]) in most cases.

The rest of this paper is organized as follows: In Section 2, we introduce the DBSCAN algorithm and the related work done by the researchers and introduce the concept of relative density. In Section 3, we describe the specific implementation process of MDBSCAN. In Section 4, we evaluate the proposed algorithm based on the experimental results and make discussions. In Section 5, we summarize the article and put forward some ideas for future work.

2. Related work

In this section, we introduce the specific process of DBSCAN and its related work. In Section 2.1, we describe the implementation process of the DBSCAN and the major problems of the DBSCAN algorithm. In Section 2.2, we describe the concept of relative density. In Section 2.3, we present the related work of DBSCAN.

2.1. DBSCAN

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise [6]) algorithm is a typical density-based algorithm. Given N data points $\{p_1, p_2, \dots, p_N\}$ in dataset D, DBSCAN can divide the data points into clusters and mark the noises. All the data points are divided into core points, density reachable points, and noises by using two parameters Eps and $MinPts$. If p_i is the core point, it forms a cluster with all the density reachable points, each cluster has at least one core point, and the non-core points can also be part of the cluster.

DBSCAN does not have to declare the number of clusters in advance, and it can effectively identify clusters with uniform densities. However, if the density of data points in the dataset is quite different, the DBSCAN algorithm will not provide a good clustering result. In this research, we propose a multi-density DBSCAN (MDBSCAN) to solve the above important problem.

2.2. Relative density (rd)

In 2021, Wang et al. introduced the concept of relative density (rd) into the clustering process [22], which not only inherits the original advantages of the algorithm but also effectively solves the problem that the density-based clustering algorithms are sensitive to parameter values. In addition, relative density is an effective tool for revealing

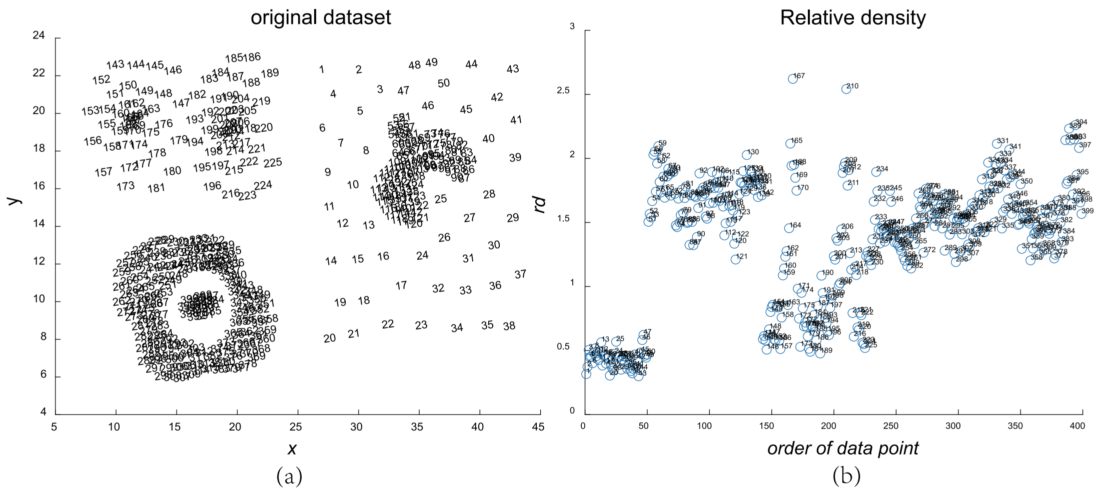


Fig. 3. Taking dataset Compound as an example, the number in the graph is the order of each data point. (a) Shape of the original dataset Compound. (b) Relative density of all the data points in the dataset Compound when we set k to 5 (corresponds to Fig. 3(a) one by one).

the density hierarchy of data points. In this research, we use relative density as a measure to divide the dataset into two parts: low density and high density. The relative density (rd) of data point p_i is defined as follows:

$$rd(i) = \frac{k}{\sum_{n=1}^{k} dist_{(p_i,p_n)}}, \tag{1}$$

where $dist(p_i, p_n)$ represents the Euclidean distance between the point p_i and p_n and k denotes the number of nearest neighbors of data point p_i. As shown in Fig. 3, we assign a number (Fig. 3(a)) to each data point in the Compound dataset so that they can correspond to each point in the relative density map (Fig. 3(b)). With this operation, we can clearly find the relative density value of each point. In our experience, the low-density region in Fig. 3(a) must contain the sparse part on the right of the figure. Their distribution on the relative density map (the dense region in the bottom left) is also consistent with our intuition. Low-density points can be well separated if a suitable relative density threshold is chosen.

2.3. Extensions of DBSCAN

DBSCAN algorithm is a density-based clustering algorithm that has many advantages such as identifying clusters with the same density. However, there are some problems with the DBSCAN algorithm in real applications: parameters are sensitive to densities of datasets, difficult to deal with large-scale datasets or datasets with continuous spatial–temporal properties, difficult to process datasets with different densities between clusters, *etc*, many researchers have improved the DBSCAN from the above problems.

To address the parameter-sensitive problem of the DBSCAN algorithm, Campello et al. proposed HDBSCAN [23] algorithm. It extends DBSCAN by transforming it into a hierarchical clustering algorithm and then extracts a flat cluster with a stable clustering technique. Compared to the DBSCAN algorithm, the HDBSCAN algorithm does not manually select the neighborhood radius Eps and *Minpts* and can automatically recommend the optimal clustering results and handle clusters with different densities. Lai et al. proposed a new method to optimize the DBSCAN parameters [24], the method designs a new mechanism for the variable update of MVO and uses it to optimize the DBSCAN parameter. It can be used to quickly find the highest clustering accuracy and corresponding parameters without manual settings. Pakdehi et al. proposed a hierarchical clustering method (DBHC) [25] based on DBSCAN. The main idea of the algorithm is to first determine the values of these parameters using the concept of a k-nearest neighbor and the concept of a k-dist graph, and then the DBSCAN algorithm was executed several times according to the previously generated *Eps* number. If the number of clusters generated is greater than the number of clusters estimated by users, the obtained clusters are merged. DBHC algorithm avoids the problems caused by manually setting the parameters by DBSCAN. Smiti et al. proposed the DBSCAN-GM [26] algorithm that combines the Gaussian mean algorithm and the DBSCAN algorithm. The main idea is to cover the limitations of DBSCAN by exploring the advantages of the Gaussian mean: small clusters with cluster centers can be generated by running the Gaussian mean, and DBSCAN parameter values can be estimated. Wang et al. proposed an adaptive Multi-density DBSCAN (AMD-DBSCAN) [21] and introduced an improved parameter adaptation method within the algorithm to search for multiple parameter pairs (i.e., Eps and *Minpts*), which are the key parameters to determine the clustering results and performance. Moreover, the algorithm requires only one hyperparameter and avoids the complicated repetitive initialization operations.

For the problems of the DBSCAN algorithm in clustering large-scale datasets, Ankerst et al. proposed the OPTICS [5] algorithm, which does not produce a clustering of a dataset explicitly, but instead creates an augmented ordering of the database representing its density-based clustering structure. For large datasets, the algorithm graphically visualizes this enhanced ordering by introducing appropriate visualization techniques. Kumar et al. proposed a novel graph-based index structure method Groups [27], which can expand the hierarchical index structure in datasets above 20 dimensions, and accelerate the neighbor search operation, increase the speed of the DBSCAN algorithm. Moreover, the method is robust to noise by prepruning the noise and eliminating unnecessary computations. In addition, some scientists have made improvements to the problem that the DBSCAN algorithm is too slow to perform on large-scale datasets, Kim et al. proposed approximate adaptive DBSCAN (AA-DBSCAN) and kAA-DBSCAN [17] algorithms. The basic idea is to define the dataset density layer with a new quadruple tree-based tree structure and focus on minimizing the additional computation required to determine the parameters by using the approximate adaptive Eps for each density. While inheriting the advantages of the DBSCAN algorithm, it significantly reduces the time required to perform clustering. Gholizadeh et al. proposed K-DBSCAN [28], the core idea is to initially classify the data through the k-means++ algorithm, and then use the DBSCAN algorithm to cluster each group separately (merge border clusters if necessary), which reduces the computational burden of performing the DBSCAN algorithm and significantly improves the speed of cluster execution.

To solve the problem that the DBSCAN algorithm makes it difficult to process datasets with different densities between clusters, Zhu et al. proposed a density-ratio-based method [29] to overcome this weakness

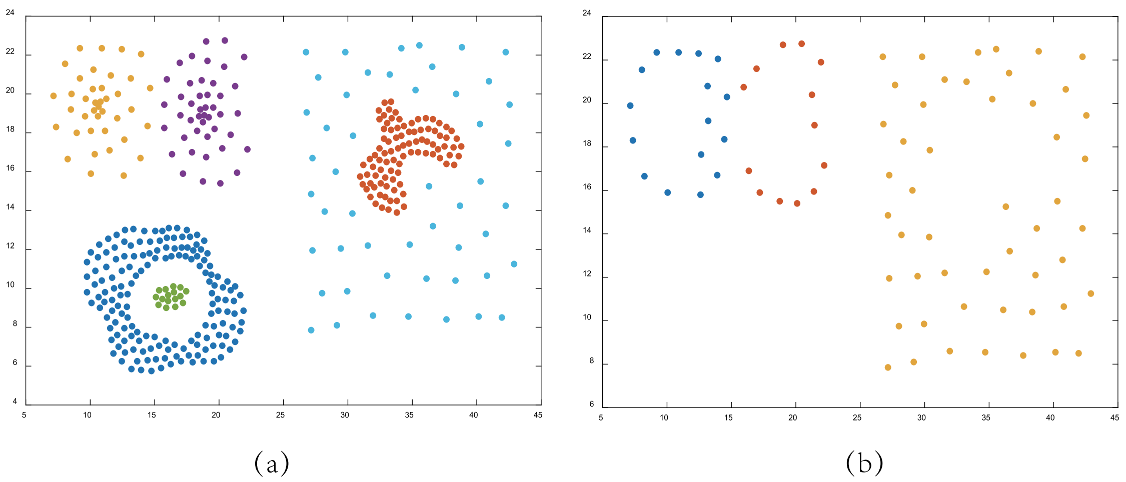


Fig. 4. (a) The ground-truth of the Compound. (b) Low-density points extracted from dataset Compound, when $k = 5$, $t = 0.7$.

and provided two implementation approaches. One approach is to modify a density-based clustering algorithm to do density-ratio-based clustering, utilizing its density estimator to compute density ratios, while the other approach focuses on rescaling the given dataset. Experiments based on DBSCAN algorithm verify the effectiveness of these two approaches. Darong et al. proposed a new algorithm GRPDBSCAN (Grid-based DBSCAN Algorithm with Referential Parameters) [30], The core idea of the GRPDBSCAN algorithm is to combine the grid partition technology with the clustering algorithm based on multi-density, which improves the efficiency of the algorithm in dealing with datasets with different densities between clusters. Scitovski et al. proposed an improvement based on DBSCAN algorithm [31], this algorithm groups the radius of every data point (with the point as the center of the circle and the radius of the circle containing the *Minpts* data points) into a suitable number of clusters and then obtains a series of parameters. Through the combination of parameters *Minpts* and radius, the initial partition is constructed, and new clusters are gradually added. Finally, the clustering results are optimized through the merging process. Compared with the standard DBSCAN algorithm, this algorithm shows significant advantages in processing data of different densities.

Also, for special datasets, Birant et al. proposed the ST-DBSCAN [32] algorithm, the core idea of this algorithm is to represent data points as spatial–temporal objects, i.e., spatial–temporal points, and then cluster according to the density accessibility of the points. In the ST-DBSCAN algorithm, the density of points is defined as the spatial density weighted by time, namely the number of data points in a certain time in the spatial range. In the meantime, the ST-DBSCAN algorithm also needs to consider the timing relationship of the points and process the time stamp of each point, so that only the points with accessible density within a certain time interval can be divided into the same cluster. Thus, the ST-DBSCAN algorithm can more efficiently handle datasets with continuous spatial–temporal properties.

The previous research never made DBSCAN identify clusters with different densities from the perspective of dividing the dataset into low density and high density, we propose a brand-new divide and conquer method to deal with respective parts to avoid them interfering with each other, the clusters with different densities are identified. The proposed MDBSCAN naturally achieves promising results on multiple-density datasets.

3. Multi-density DBSCAN based on relative density (MDBSCAN)

In this section, we introduce the proposed Multi-density DBSCAN based on the relative density. As mentioned in Section 2.1, DBSCAN is not good at identifying clusters with multiple densities because data points with multiple densities are close in Euclidean space. To solve this problem, we first divide the dataset into two parts: low density and high density, then we design a divide and conquer method to deal with the respective part. Specifically, MDBSCAN consists of the following three key steps: (i) extract the low-density data points in the dataset. (ii) find natural clusters among the identified low-density data points. (iii) clustering the remaining data points (except data points of natural clusters in a dataset) by using DBSCAN and assigning the noises (generated by DBSCAN) to the nearest clusters. We will introduce these three steps in detail in the following subsections, and at the end of this section, we will analyze the complexity of the algorithm.

3.1. *Extract low-density data points in the dataset*

In Section 2.2, we introduce the basic concept of relative density. To obtain the low-density data points in the dataset, we define two parameters, namely k (the number of nearest neighbors) and t (density threshold parameter), where k determines the number of nearest neighbors involved in the calculation of relative density, and t determines which data points have low densities. For a given dataset, we will calculate the relative density $rd(i)$ of each data point p_i in the dataset by using Eq. (1), and compare it with t. If $rd(i)$ is less than t, as shown in Fig. 4, the data point will be merged into low-density points set L, which is expressed as:

$$L = \{p_i | rd(i) < t\} \tag{2}$$

3.2. *Find natural clusters with low-density*

In this research, as shown in Fig. 4(b), the low-density data points L obtained by Section 3.1 contain some data points belonging to the high-density clusters, if data points in L are directly merged into a cluster, the clustering accuracy is greatly reduced. To obtain more accurate and reliable clustering results, it is often necessary to further extract natural clusters from the extracted low-density data points L, so we developed a new Share Nearest Neighbors-based Clustering method to extract natural clusters and named it SNNC, as shown in Fig. 5, it can find the natural clusters in Fig. 4(b), and the remaining data points (upper-left part of Fig. 4(b)) participate in the execution of DBSCAN algorithm together with high-density data points.

In the algorithm SNNC, we use $knn(i)$ to represent the k-nearest neighbor set of data point p_i. In step (i), when the k-nearest neighbor sets ($knn(i)$ and $knn(j)$) of two points (p_i and p_j) have intersections, the two data points are stored in $kc(i)$. The specific process of SNNC is as follows:

(i) For the data points in L, we loop check whether $knn(i)$ and $knn(j)$ have intersections ($i < j$), if so, the data points p_i and p_j are merged into $kc(i)$.

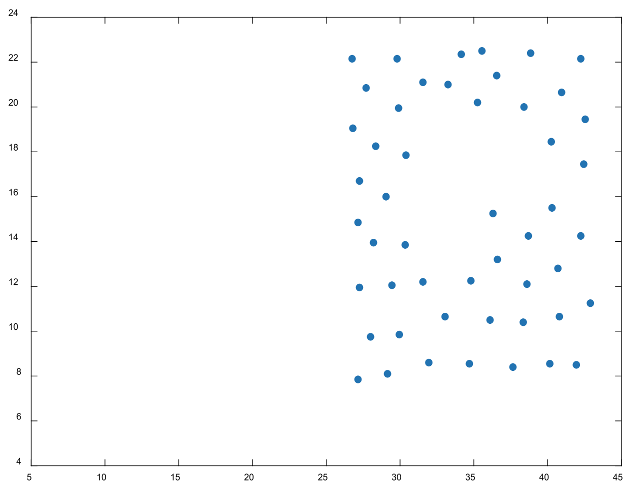


Fig. 5. Natural cluster in Fig. 4(b) extracted by using SNNC.

(ii) If there is an intersection between $kc(i)$ and $kc(j)$ $(i < j)$, the data points $kc(j)$ and $kc(i)$ are merged into $kc(i)$, and let $kc(j)$ become empty. Any pair of $kc(i)$ and $kc(j)$ that have intersections are merged in this step.

(iii) For d non-empty $\{kc(1), kc(2), \ldots, kc(n)\}$ obtained in the previous step, we use nkc_i to denote the number of data points contained in each $kc(i)$. Then, the mean value of nkc is defined as:

$$mean = \left(\sum_{i=1}^{n} nkc_i\right)/d. \tag{3}$$

For all the data points set kc, if nkc_i of $kc(i)$ is greater than the $mean$, we merge $kc(i)$ into F, which is defined as:

$$F = \cup\{kc(i) | nkc_i > mean\}, \tag{4}$$

where the F is the natural clusters we want to obtain. To understand our method easily, we present the implementation process of SNNC in Algorithm 1.

3.3. Clustering the remaining data points by DBSCAN

After obtaining a cluster F, for the remaining data points $(D - F)$, we use the DBSCAN algorithm to process them. Taking the dataset Compound as an example, Fig. 6(a) shows the results of clustering the remaining points by the DBSCAN algorithm. For noises obtained by DBSCAN algorithm execution, as shown in Fig. 6(b), we assign them to the nearest clusters. To show the process of the proposed MDBSCAN algorithm more clearly, the flowchart is shown in Fig. 7, and the pseudo-code of MDBSCAN is given in Algorithm 2.

3.4. Complexity analysis

For a dataset containing n data points, in the first step of the algorithm, we extract low-density data points in the dataset, which mainly involves the calculation of the distance matrix of data points and the relative density. The time complexity of distance matrix calculation is $\mathcal{O}(n^2)$. The time complexity of ascending sort each row of the distance matrix is $\mathcal{O}(n * log(n))$, and the complexity of calculating the relative density of each data point is $\mathcal{O}(n)$, so the total time complexity of the first step is $\mathcal{O}(n^2 + n * log(n) + n) = \mathcal{O}(n^2)$. In the second step of the algorithm, we used SNNC to extract natural cluster in low-density data points, which involves three main steps: (i) merging data points iteratively, the time complexity is $\mathcal{O}(m^2)$, where m represents the number of low-density points extracted and $m < n$. (ii) merging data point sets iteratively, the time complexity is $\mathcal{O}(m^2)$. (iii) extracting

Algorithm 1 Share Nearest Neighbors-based Clustering method (SNNC)

```
1: input: Low-density data points..
2: output: Low-density natural clusters $F$.
3: $knn(i)$ ⇐The $k$-nearest neighbor set for each data point $p_i$.
4: $num$ ⇐The number of low-density points.
5: $i$ ⇐1;
6: while $i <= num$ do
7:     $j \Leftarrow i+1$;
8:     while $j <= num$ do
9:         if $|knn(i) \cap knn(j)| >= 1$ then
10:            merge data points $p_i$ and $p_j$ into $kc(i)$;
11:        end if
12:        $j \Leftarrow j + 1$;
13:    end while
14:    $i \Leftarrow i + 1$; $j \Leftarrow i + 1$;
15: end while
16: for $t$=1 to 100 do
17:     for $i$=1 to $num$ do
18:         for $j$=$i$+1 to $num$ do
19:             if $|kc(i) \cap kc(j)| >= 1$ then
20:                 merge data points $kc(i) \cup kc(j)$ into $kc(i)$;
21:                 empty $kc(j)$;
22:             end if
23:         end for
24:     end for // If no more $kc$ are merged, break.
25: end for
26: $nkc_i$ ⇐Number of data points in $kc(i)$;
27: $mean$ ⇐The mean of $nkc_i$ (for $d$ non-empty $kc(i)$);
28: for $i$=1 to $num$ do
29:     if $nkc_i >= mean$ then
30:         add data points in $kc(i)$ into $F$;
31:     end if
32: end for
```

Algorithm 2 MDBSCAN

```
1: input: Dataset $D$.
2: output: Clustering results.
3: $rd(i)$ ⇐ Relative density of each data point.
4: $t$ ⇐ Relative density threshold parameter.
5: $L$ ⇐Temporary empty dataset.
6: if $rd(i) < t$ then
7:     Copy point $p_i$ from $D$ to $L$
8: end if
9: Execute the Algorithm 1 on dataset $L$, get a low-density cluster $F$.
10: $Eps$ ⇐ Distance threshold parameter.
11: $Minpts$ ⇐ Neighborhood sample threshold parameter.
12: DBSCAN($D - F, Eps, MinPts$).
13: The outliers obtained by the DBSCAN algorithm are assigned to the
    nearest clusters.
```

natural cluster, the time complexity is $\mathcal{O}(m * d + d) = \mathcal{O}(m * d)$ (d is explained in Section 3.2). The total time complexity of the second step is $\mathcal{O}(m^2 + m * d + m^2) = \mathcal{O}(m^2)$. Finally, we executed the DBSCAN algorithm on the remaining data points, and the time complexity is $\mathcal{O}(n^2)$. In summary, the time complexity of MDBSCAN is $\mathcal{O}(n^2 + m^2)$. Since the value of m is obviously less than n, the total time complexity $\mathcal{O}(n^2)$ is finally obtained.

4. Performance evaluation and discussions

In this section, we verify the effectiveness of the MDBSCAN algorithm. We conduct experiments on ten synthetic datasets and six UCI datasets and evaluate the clustering results of the proposed MDBSCAN

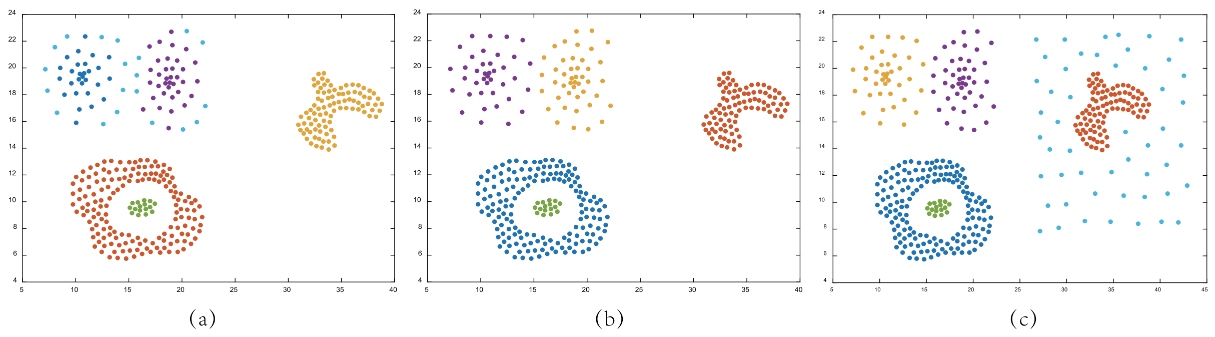


Fig. 6. Taking dataset Compound as an example. (a) The results of clustering remaining points (*D-F*) by DBSCAN. (b) Assign noise points in Fig. 6(a) to the nearest clusters. (c) The final clustering results.

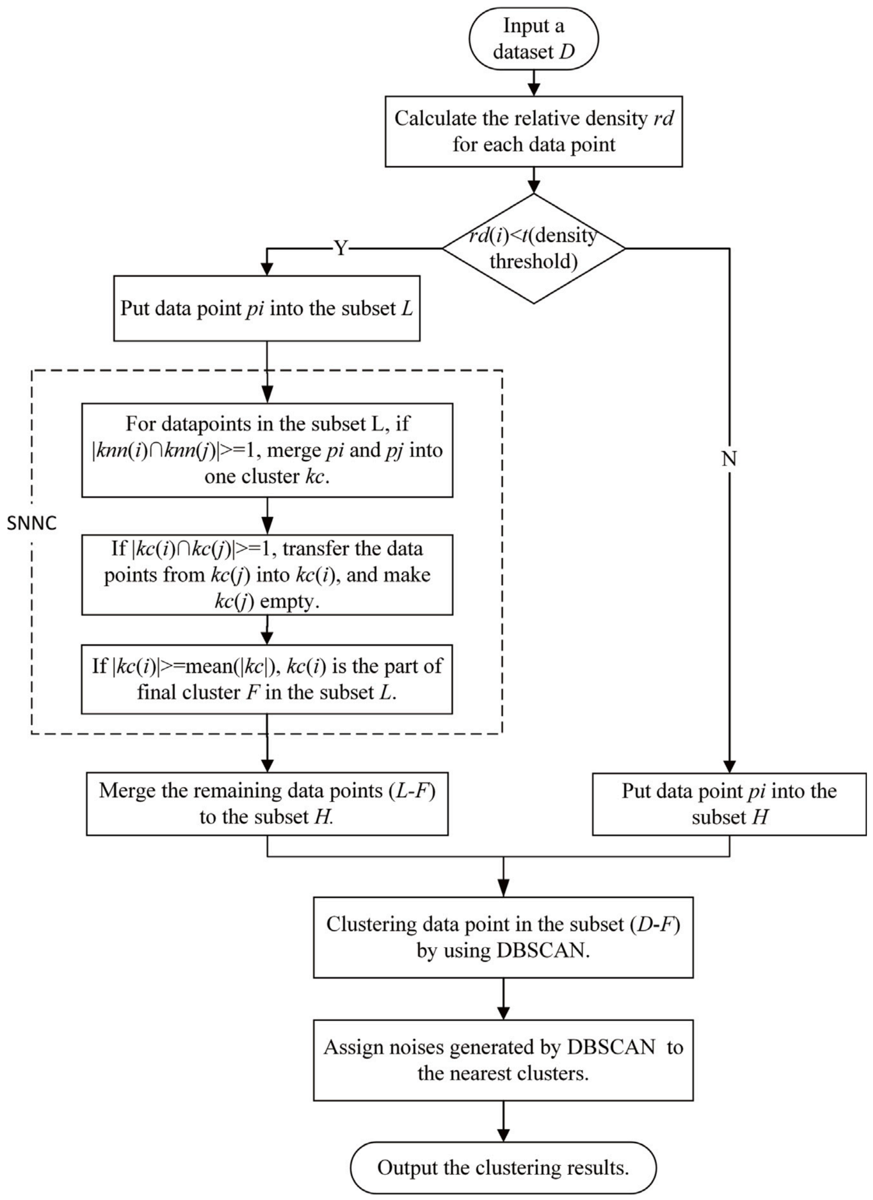

Input a dataset *D*
Calculate the relative density *rd* for each data point
rd(*i*)<*t*(density threshold)
Y
N
Put data point *pi* into the subset *L*
SNNC
For datapoints in the subset L, if |*knn*(*i*)∩*knn*(*j*)|>=1, merge *pi* and *pj* into one cluster *kc*.
If |*kc*(*i*)∩*kc*(*j*)|>=1, transfer the data points from *kc*(*j*) into *kc*(*i*), and make *kc*(*j*) empty.
If |*kc*(*i*)|>=mean(|*kc*|), *kc*(*i*) is the part of final cluster *F* in the subset *L*.
Merge the remaining data points (*L-F*) to the subset *H*.
Put data point *pi* into the subset *H*
Clustering data point in the subset (*D-F*) by using DBSCAN.
Assign noises generated by DBSCAN to the nearest clusters.
Output the clustering results.

Fig. 7. Flowchart of the MDBSCAN algorithm.

Table 1
The features of used dataset.

Type	ID	Datasets 3	Samples	Dimensions	Natural clusters
Synthetic	1	2circles	600	2	2
Synthetic	2	Aggregation	788	2	7
Synthetic	3	Compound	399	2	6
Synthetic	4	D31	3100	2	31
Synthetic	5	Flame	240	2	2
Synthetic	6	Jain	373	2	2
Synthetic	7	Pathbased	300	2	3
Synthetic	8	D1	87	2	3
Synthetic	9	Zelink6	1238	2	3
Synthetic	10	2circles-noise	634	2	3
UCI	1	Ecoli	336	7	8
UCI	2	Glass	214	9	2
UCI	3	Ionosphere	351	34	2
UCI	4	Iris	150	4	3
UCI	5	Vote	435	16	2
UCI	6	Leaf	340	15	30

Table 2
Parameters of selected clustering algorithms.

Algorithms	Required parameters with descriptions
DBSCAN	Eps: scanning radius *MinPts*: minimum number of data points contained
OPTICS	Eps: scanning radius *MinPts*: minimum number of data points contained
STDBSCAN	Eps: scanning radius *MinPts*: minimum number of data points contained t: time range of spatial–temporal data
HDBSCAN	*msc*: minimum number of samples in the cluster
DRL-DBSCAN	//
AMD-DBSCAN	*balancenum*: the number of occurrences of the same cluster
RNN-DBSCAN	K: nearest neighbor parameter
Ng'work	*n_cluster*: cluster dataset into n_cluster subset *sigma*: a parameter of the affinity matrix

algorithm using NMI [33] and ARI [34]. Section 4.1 introduces the five clustering algorithms used for the comparison. The comparison results of the six algorithms are presented in Section 4.2, Section 4.3 examines the effect of four parameters on the effect of clustering, and we will share our experience in determining the four parameters during the experimental process in Section 4.4. The features of all the datasets used in this paper are shown in Table 1, and all the datasets can be downloaded online.[2]

4.1. Benchmarking models

We select a classical clustering algorithm DBSCAN [6] and six extensions of DBSCAN namely OPTICS [5], ST-DBSCAN [32], HDBSCAN [23], AMD-DBSCAN [21], DRL-DBSCAN [20], and RNN-DBSCAN [35] as benchmarking models. In addition, we also choose a traditional clustering algorithm as the benchmark model, namely spectral clustering (proposed by Ng [12]). Most algorithms need to configure parameters, the details are shown in Table 2. Therefore, to ensure the effectiveness of comparison results, we adjust the parameters in advance to achieve the best results. We use the same evaluation indexes NMI [33] and ARI [34] to evaluate the clustering results of different clustering algorithms. The NMI and ARI index is defined as follows:

$$NMI(X,Y) = \frac{2MI(X,Y)}{H(X)+H(Y)}, \tag{5}$$

where X represents the non-duplicate value of the real label, and Y represents the non-duplicate value of the cluster label, $H(X)$ is the entropy of X, $H(Y)$ is the entropy of Y, $MI(X, Y)$ defined as:

$$MI(X,Y) = \sum_{i=1}^{|X|}\sum_{j=1}^{|Y|} P(i,j)log\frac{P(i,j)}{P(i)P(j)}, \tag{6}$$

where $P(i)$ denote the probability distribution of i.

$$ARI = \frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)}, \tag{7}$$

where a represents the number of point pairs belonging to the same cluster in both ground-truth and clustering results; b represents the number of point pairs belonging to the same cluster in ground-truth but not in clustering results; c represents the number of point pairs not belonging to the same cluster in ground-truth but in clustering results. d represents the number of point pairs that do not belong to the same cluster under both ground-truth and clustering results. The value range of ARI is [−1, 1], and the value range of NMI is [0, 1]. Larger values of ARI and NMI indicate better clustering results, the MATLAB code of ARI and NMI can be downloaded online.[3]

4.2. Experiments on synthetic and UCI datasets

Table 3 and Figs. 8–17 show the clustering results and corresponding parameter values of 10 synthetic datasets. Compared to the eight clustering algorithms used for comparison, MDBSCAN produces the best results on most (seven) synthetic datasets. Especially for the 2circles, Aggregation, Compound, D1, zelink6, and 2circles-Noise datasets, MDBSCAN reached 1.000 in ARI and NMI evaluation index. For the Jain dataset (including two clusters), the two evaluation indexes of the MDBSCAN algorithm are both smaller than the DBSCAN algorithm, because we cannot select a qualified t value to completely distinguish the two clusters, and low-density points in high-density clusters will cause interference to our extraction of natural classes when we set a large k value. However, the DBSCAN algorithm directly identifies all data points contained in one of the clusters as noise to achieve the best clustering effect.

Table 4 shows the clustering results and the corresponding parameter values for the six UCI datasets. On the dataset Glass, Iris, Ionosphere, and Ecoli, two evaluation indexes of the proposed algorithm, ARI and NMI, achieve the best clustering effect, while on the Vote and Leaf datasets, only one index achieves the best clustering effect.

4.3. Sensitivity tests on various parameters of MDBSCAN

Taking three UCI datasets and three synthetic datasets as an example, we investigate the influence of four parameters k, t, Eps, and $Minpts$ on the clustering results (keep other values optimal while evaluating one parameter). As shown in Tables 3 and 4, it is not difficult to find that the value of parameters Eps and $Minpts$ of the MDBSCAN algorithm is the same as that of the DBSCAN algorithm when it achieves the optimal results in most cases. Considering Figs. 18–23, when the DBSCAN algorithm achieves the optimal results, the proposed MDBSCAN can improve the clustering results within a certain range by adjusting the values of the parameters k and t.

4.4. Parameter selection

The choice of four parameters does introduce additional challenges to the clustering experiments, and we will share some of our experiences in determining the values of four parameters in this section.

During the experiment, we first additionally executed the data point relative density calculation part of the code to determine the values of parameters k and t. The purpose of this is to generate a relative density map of the dataset similar to Fig. 3(b), which enables the selection of an appropriate value of k that can clearly distinguish high-density points from low-density points. The value of parameter k is a positive integer and should not exceed the number of data points in the dataset.

[2] https://github.com/mlyizhang/Clustering-Datasets
[3] https://github.com/mlyizhang/Clustering-evaluation

Table 3
Comparison of clustering results on ten synthetic datasets.

Algorithms	Par	Val	NMI	ARI	Val	NMI	ARI
Dataset 2circles					**Dataset Aggregation**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	6/3/1.8/5	**1.0000**	**1.0000**	55/0.15/1.6/10	**1.0000**	**1.0000**
DBSCAN	*Eps/Minpts*	1.8/5	**1.0000**	**1.0000**	1.6/10	0.9841	0.9877
OPTICS	*Eps/Minpts*	1/12	**1.0000**	**1.0000**	1.6/9	0.9822	0.9878
STDBSCAN	*Eps/T/Minpts*	1.8/1.1/5	**1.0000**	**1.0000**	1.6/1.35/10	0.9840	0.9887
HDBSCAN	*ms*	5	**1.0000**	**1.0000**	11	0.9063	0.8389
DRL-DBSCAN	/	/	**1.0000**	**1.0000**	/	0.9752	0.9793
AMD-DBSCAN	*balancenum*	3	0.0000	0.0000	3	0.9742	0.9781
RNN-DBSCAN	*K*	11	**1.0000**	**1.0000**	7	0.9957	0.9978
Ng'work	*n_clusters/sigma*	2/1	**1.0000**	**1.0000**	4/1.7	0.8359	0.7338
Dataset Compound					**Dataset D31**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	5/0.7/1.3/3	**1.0000**	**1.0000**	150/0.272/1.2/64	**0.9626**	**0.9451**
DBSCAN	*Eps/Minpts*	1.5/3	0.9411	0.9713	1.3/66	0.9245	0.8818
OPTICS	*Eps/Minpts*	1.47/3	0.9205	0.9567	1.3/64	0.8813	0.7338
STDBSCAN	*Eps/T/Minpts*	1.43/1.47/3	0.9106	0.8964	1.15/1.15/65	0.9155	0.8343
HDBSCAN	*ms*	1	0.8836	0.9172	6	0.8418	0.5608
DRL-DBSCAN	/	/	0.8162	0.8237	/	0.6966	0.3113
AMD-DBSCAN	*balancenum*	3	0.8741	0.8935	3	0.8863	0.7564
RNN-DBSCAN	*K*	13	0.9144	0.8590	22	0.9567	0.9163
Ng'work	*n_clusters/sigma*	3/0.8	0.7912	0.7375	31/3.2	0.9239	0.8575
Dataset Flame					**Dataset Jain**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	10/0.3/1.2/9	0.9200	0.9608	38/0.3/2.2/14	0.9476	0.9775
DBSCAN	*Eps/Minpts*	1.3/8	0.8998	0.9550	2.2/14	**1.0000**	**1.0000**
OPTICS	*Eps/Minpts*	1.3/8	0.8264	0.9077	2.2/15	0.9469	0.9775
STDBSCAN	*Eps/T/Minpts*	1.25/0.94/9	0.8957	0.9447	2/1.2/12	0.9303	0.9664
HDBSCAN	*ms*	1	0.8504	0.9129	19	0.8438	0.9378
DRL-DBSCAN	/	/	0.8691	0.9337	/	0.9714	0.9887
AMD-DBSCAN	*balancenum*	3	0.6692	0.7215	3	0.8430	0.9265
RNN-DBSCAN	*K*	9	**0.9635**	**0.9833**	16	**1.0000**	**1.0000**
Ng'work	*n_clusters/sigma*	2/0.1	0.9354	0.9666	3/17	0.5783	0.4824
Dataset Pathbased					**Dataset D1**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	180/0.108/1.8/5	0.9315	0.9498	3/2.1/0.7/10	**1.0000**	**1.0000**
DBSCAN	*Eps/Minpts*	2.1/10	0.8875	0.9195	0.7/10	**1.0000**	**1.0000**
OPTICS	*Eps/Minpts*	2.1/10	0.8211	0.8506	0.5/5	0.9061	0.9315
STDBSCAN	*Eps/T/Minpts*	1.75/1.7/9	0.8988	0.9294	0.5/0.5/1	**1.0000**	**1.0000**
HDBSCAN	*ms*	15	0.7522	0.7869	2	**1.0000**	**1.0000**
DRL-DBSCAN	/	/	0.8494	0.8909	/	**1.0000**	**1.0000**
AMD-DBSCAN	*balancenum*	3	0.6174	0.5812	3	0.0000	0.0000
RNN-DBSCAN	*K*	7	**0.9335**	**0.9543**	6	**1.0000**	**1.0000**
Ng'work	*n_clusters/sigma*	4/0.5	0.8426	0.8330	3/0.2	**1.0000**	**1.0000**
Dataset Zelink6					**Dataset 2circles-noise**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	6/40/0.1/5	**1.0000**	**1.0000**	30/0.5/0.9/5	**1.0000**	**1.0000**
DBSCAN	*Eps/Minpts*	0.05/55	**1.0000**	**1.0000**	0.9/4	**1.0000**	**1.0000**
OPTICS	*Eps/Minpts*	0.05/55	0.9587	0.9771	0.9/4	0.9640	0.9873
STDBSCAN	*Eps/T/Minpts*	0.02/0.02/6	**1.0000**	**1.0000**	0.9/0.8/6	0.9935	0.9970
HDBSCAN	*ms*	7	0.7438	0.7860	4	0.9857	0.9939
DRL-DBSCAN	/	/	**1.0000**	**1.0000**	/	**1.0000**	**1.0000**
AMD-DBSCAN	*balancenum*	3	0.6206	0.5943	3	0.9672	0.9817
RNN-DBSCAN	*K*	13	0.6888	0.7228	11	0.9869	0.9939
Ng'work	*n_clusters/sigma*	3/4.1	0.8705	0.9092	2/1	0.9212	0.9392

The best results are highlighted in boldface.

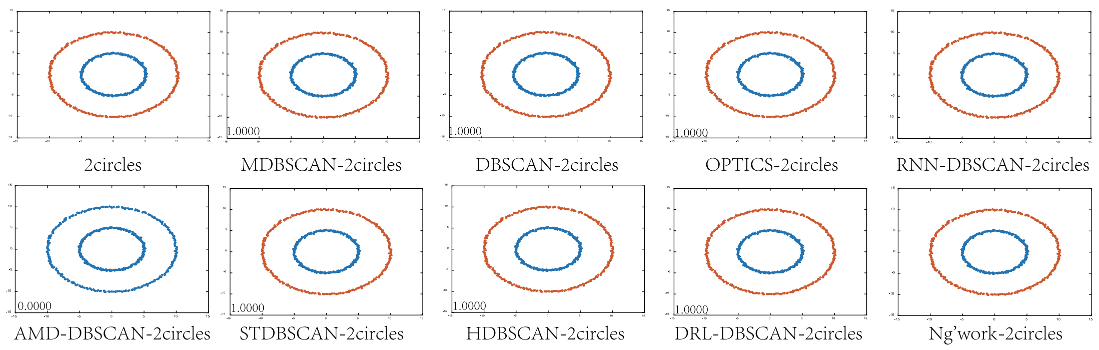


Fig. 8. Result comparisons on 2circles (the numbers in the subfigures represent the NMI index).

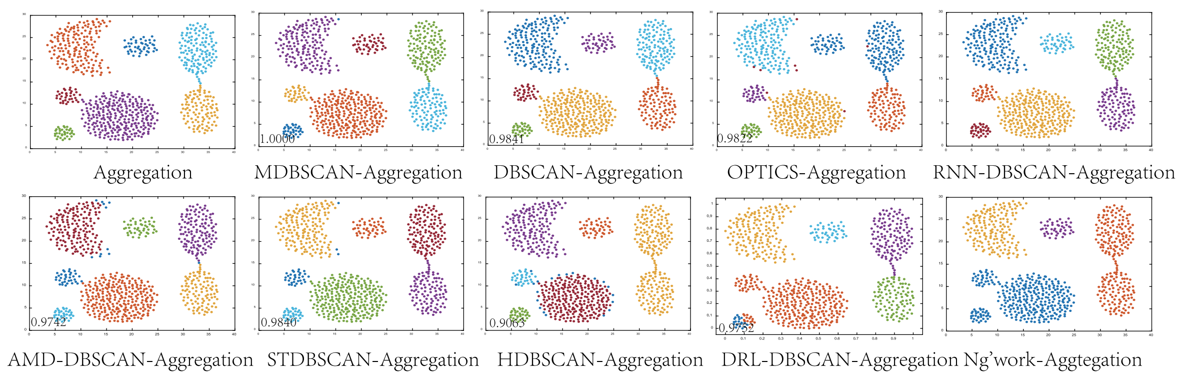


Fig. 9. Result comparisons on Aggregation (the numbers in the subfigures represent the NMI index).

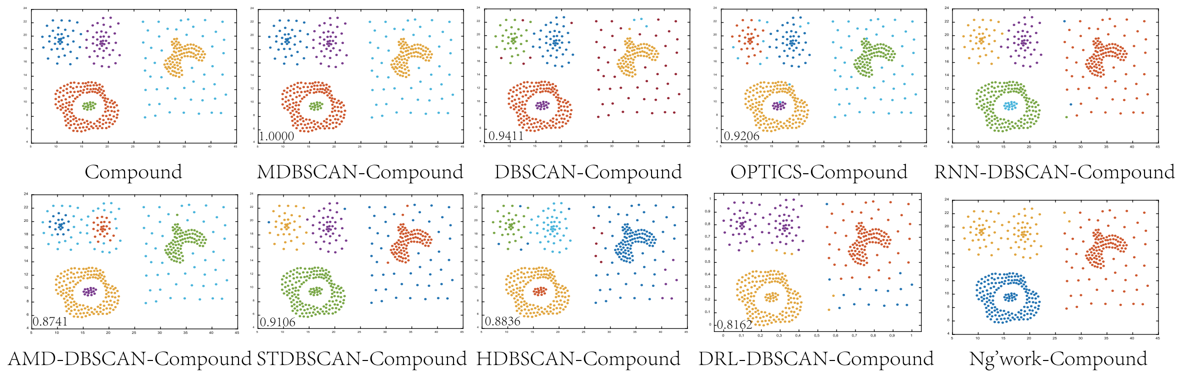


Fig. 10. Result comparisons on Compound (the numbers in the subfigures represent the NMI index).

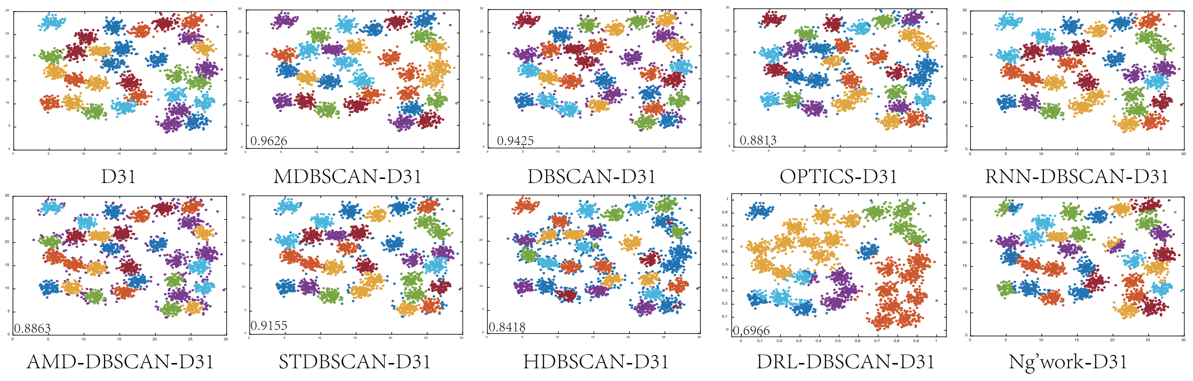


Fig. 11. Result comparisons on D31 (the numbers in the subfigures represent the NMI index).

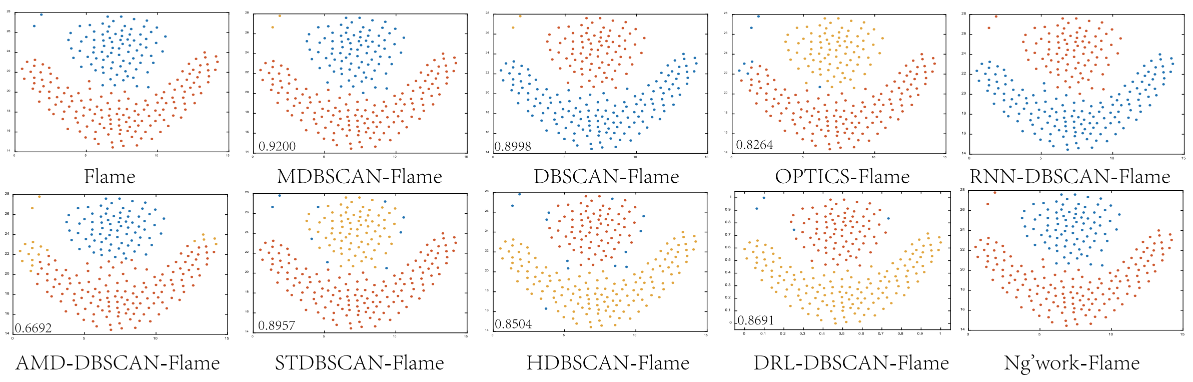


Fig. 12. Result comparisons on Flame (the numbers in the subfigures represent the NMI index).

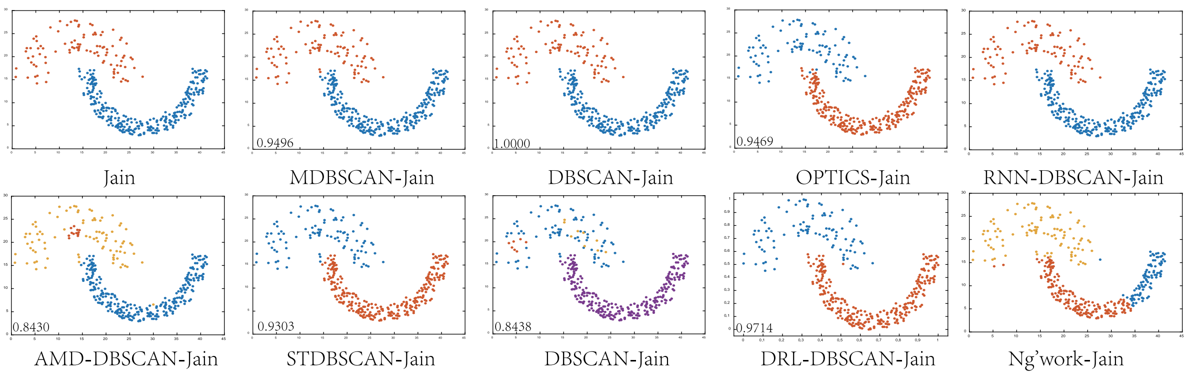


Fig. 13. Result comparisons on Jain (the numbers in the subfigures represent the NMI index).

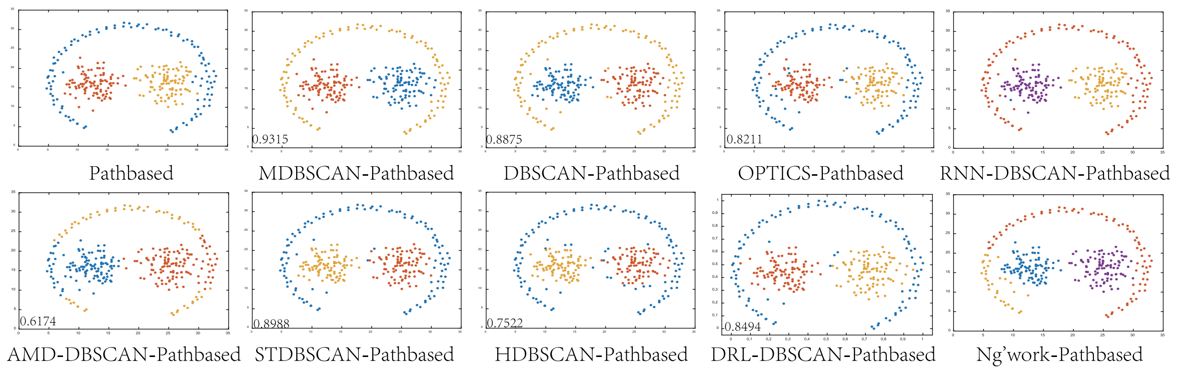


Fig. 14. Result comparisons on Pathbased (the numbers in the subfigures represent the NMI index).

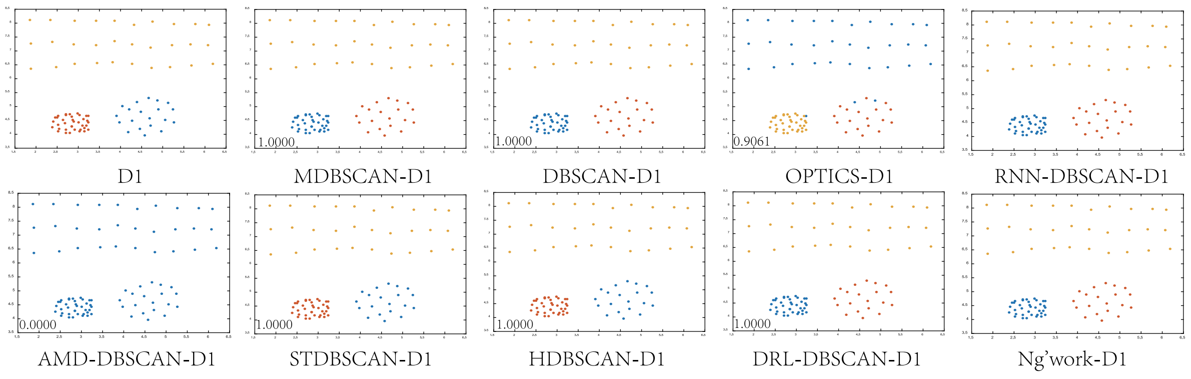


Fig. 15. Result comparisons on D1 (the numbers in the subfigures represent the NMI index).

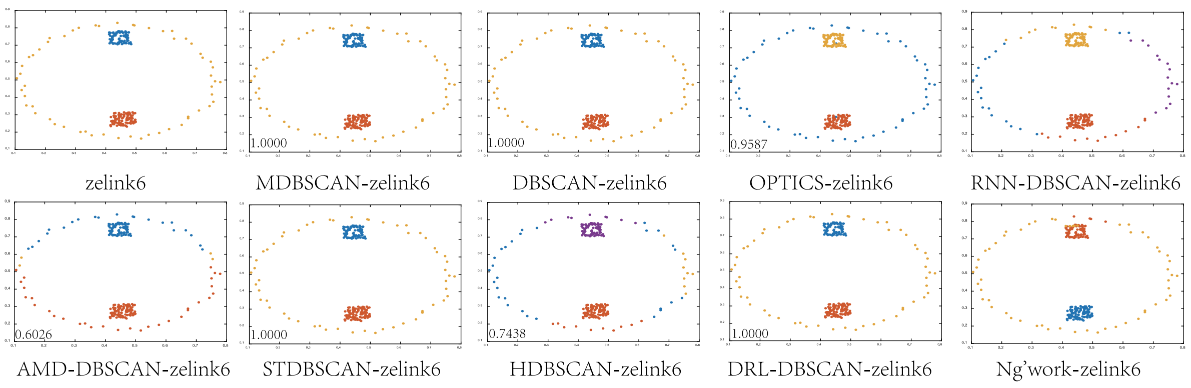


Fig. 16. Result comparisons on Zelink6 (the numbers in the subfigures represent the NMI index).

Table 4
Comparison of clustering results on six UCI datasets.

Algorithms	Par	Val	NMI	ARI	Val	NMI	ARI
Dataset Glass					**Dataset Ionosphere**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	75/0.14/0.9/1	**0.5258**	**0.2929**	3/0.66/1.3/3	**0.6402**	**0.7286**
DBSCAN	*Eps/Minpts*	0.9/1	**0.5258**	**0.2929**	1.8/12	0.5995	0.6820
OPTICS	*Eps/Minpts*	1/2	0.4493	0.2274	1.8/4	0.5820	0.6724
STDBSCAN	*Eps/T/Minpts*	0.9/2/6	0.4630	0.2830	1.8/0.8/12	0.6352	0.7214
HDBSCAN	*ms*	17	0.3690	0.2461	14	0.2874	0.3059
DRL-DBSCAN	/	/	0.4996	0.2624	/	0.3296	0.5560
AMD-DBSCAN	*balancenum*	3	0.0000	0.0000	3	0.0000	0.0000
RNN-DBSCAN	*K*	6	0.3985	0.2319	4	0.1247	0.1142
Ng'work	*n_clusters/sigma*	5/21.7	0.3718	0.2797	2/0.2	0.1747	0.2501
Dataset Iris					**Dataset Vote**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	4/1/0.39/4	**0.8366**	**0.8517**	44/0.21/1.5/3	0.5156	**0.5968**
DBSCAN	*Eps/Minpts*	1/6	0.7337	0.5681	1/3	0.3955	0.2999
OPTICS	*Eps/Minpts*	1/6	0.7037	0.5558	1.2/3	0.3919	0.2952
STDBSCAN	*Eps/T/Minpts*	1/0.9/6	0.7337	0.5681	1.1/1.8/3	**0.6284**	0.5958
HDBSCAN	*ms*	10	0.7337	0.5681	14	0.3801	0.3773
DRL-DBSCAN	/	/	0.7337	0.5681	/	0.2925	0.2322
AMD-DBSCAN	*balancenum*	3	0.7074	0.5601	3	0.4023	0.4511
RNN-DBSCAN	*K*	6	0.7884	0.7504	2	0.1952	0.0223
Ng'work	*n_clusters/sigma*	3/0.3	0.7782	0.7860	2/0.2	0.4733	0.5777
Dataset Ecoli					**Dataset Leaf**		
MDBSCAN (ours)	*k/t/Eps/Minpts*	4/4.2/0.11/5	**0.6595**	**0.7414**	4/0.85/0.075/1	0.7381	**0.1278**
DBSCAN	*Eps/Minpts*	0.11/1	0.5473	0.5004	0.05/1	0.7356	0.0000
OPTICS	*Eps/Minpts*	0.11/1	0.4408	0.3701	0.43/1	0.3147	0.0094
STDBSCAN	*Eps/T/Minpts*	0.09/1/1	0.5571	0.5481	0.07/6/1	**0.7387**	0.0128
HDBSCAN	*ms*	11	0.4216	0.4065	1	0.2367	0.0276
DRL-DBSCAN	/	/	0.5423	0.4979	/	0.7360	0.0104
AMD-DBSCAN	*balancenum*	3	0.4974	0.4704	3	0.0000	0.0000
RNN-DBSCAN	*K*	4	0.5622	0.5154	3	0.4084	0.0785
Ng'work	*n_clusters/sigma*	3/0.3	0.4700	0.4873	13/12.3	0.3379	0.0599

The best results are highlighted in boldface.

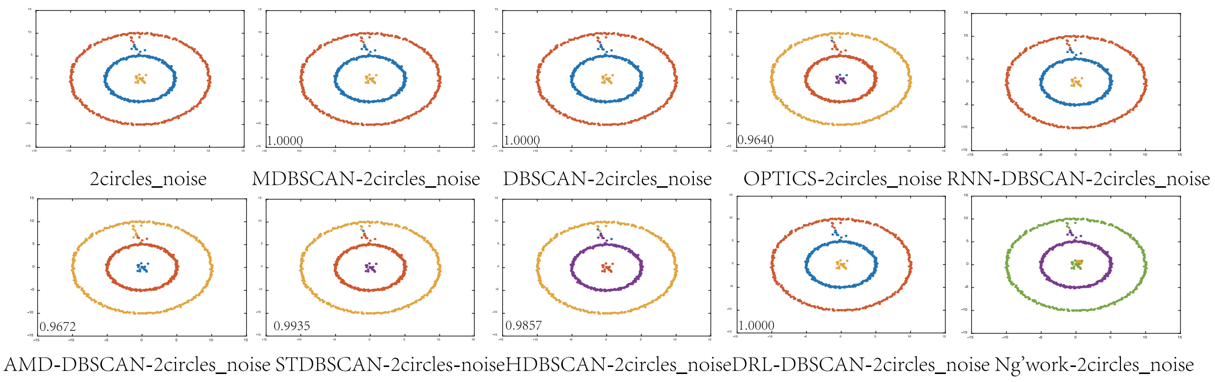


Fig. 17. Result comparisons on 2circles_noise (the numbers in the subfigures represent the NMI index).

Utilizing a small piece of code for selecting the k value is an efficient approach, far more time-effective than running the entire program to adjust the k value.

After determining the value of k, we select the relative density value with a clear demarcation as the value of parameter t by observing the vertical axis of the generated relative density map. Taking Fig. 3(b) as an example, when the value of k is set to 5, by carefully observing the relative density map, we find that when the parameter t is set to 0.6, a rough distinction can be made between low-density points and high-density points. This approach can help us make informed decisions when adjusting the values of parameters k and t.

For the adjustment of $Minpts$ and Eps parameters, many authors have given methods before [6,31,36]. In our experiment, we adopted a unique optimization method based on the K-dist graph [6]: (i) calculate the distance from each point in the dataset to its Kth nearest point (K is chosen around 2*dataset dimension-1), and then sort these distances in descending order to plot the K-dist graph of the dataset. (ii) observe the vertical coordinate value at the inflection point of the K-dist graph, and preliminarily determine the value range of parameter Eps (if the vertical coordinate is 0.5, set the range for the parameter Eps between 0.4 and 0.6). Simultaneously, set the $Minpts$ parameter range to integers around K. (iii) combine with the visualized clustering effects, iteratively adjust the parameters Eps and $Minpts$ to find the optimal combination. The advantage of this method is that the range of the two parameters is limited to a relatively small interval, which effectively avoids too many invalid attempts in the process of parameter adjustment and improves the efficiency of clustering.

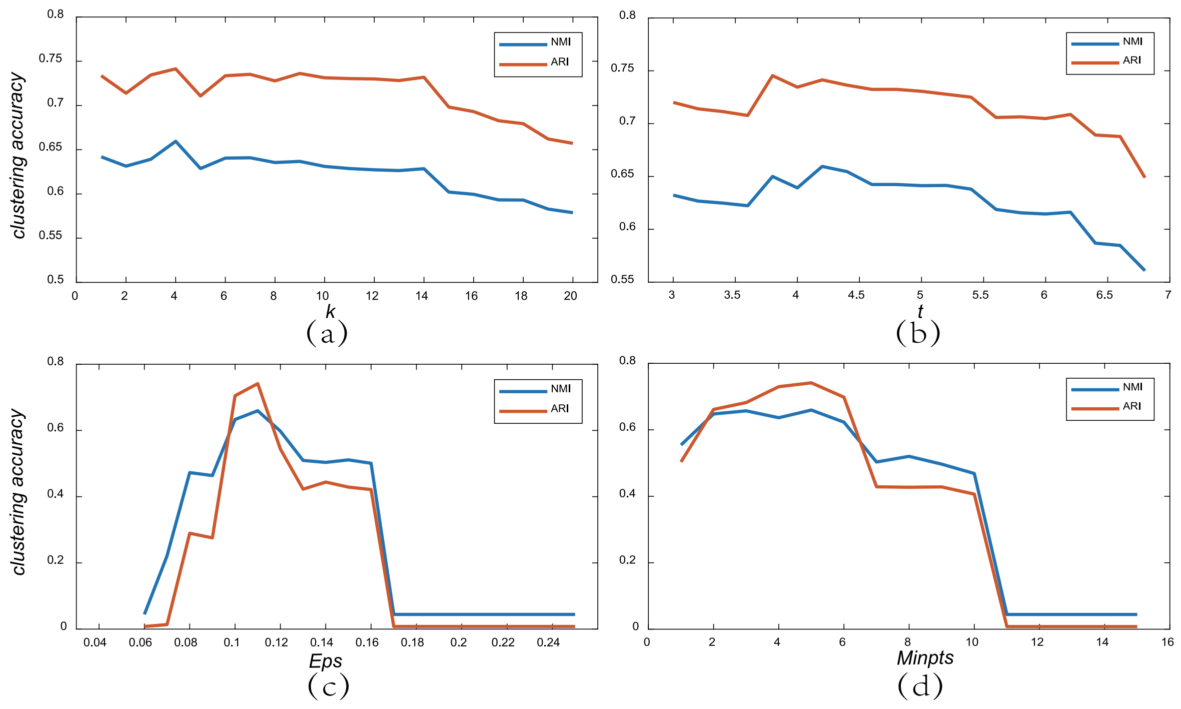


Fig. 18. For the dataset Ecoli: (a) effect of k value on clustering results. (b) effect of t value on clustering results. (c) effect of Eps value on clustering results. (d) effect of $Minpts$ value on clustering results.

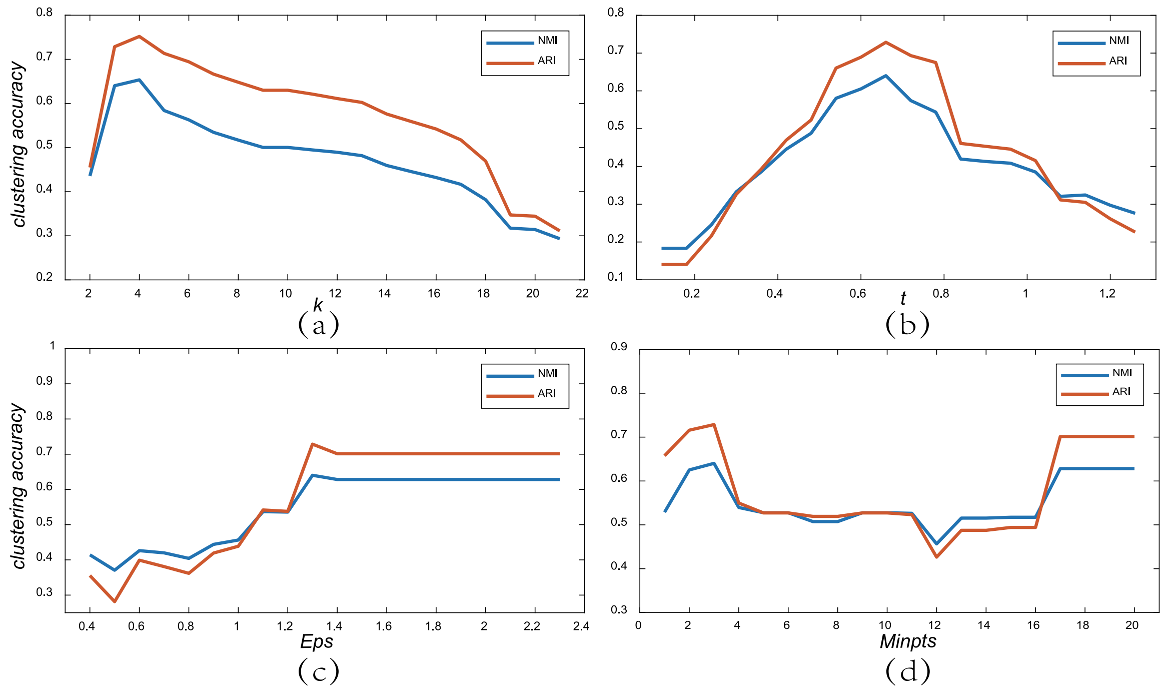


Fig. 19. For the dataset Ionosphere: (a) effect of k value on clustering results. (b) effect of t value on clustering results. (c) effect of Eps value on clustering results. (d) effect of $Minpts$ value on clustering results.

5. Conclusions and future work

In this paper, we present a novel improved DBSCAN based on the relative density (MDBSCAN), which can significantly solve the problems encountered when the DBSCAN algorithm identifies low-density clusters. Based on this algorithm, we also propose an effective method of SNNC to extract natural clusters. In the experimental section, we use sixteen datasets to verify the effectiveness of the MDBSCAN algorithm, and the experimental results show that the MDBSCAN algorithm outperforms the other five clustering algorithms in most cases when processing the datasets with obviously low-density clusters.

In the subsequent research, we can make improvements in two aspects of the MDBSCAN algorithm. The first is the adaptive selection of parameter k. When the MDBSCAN algorithm processes large-scale

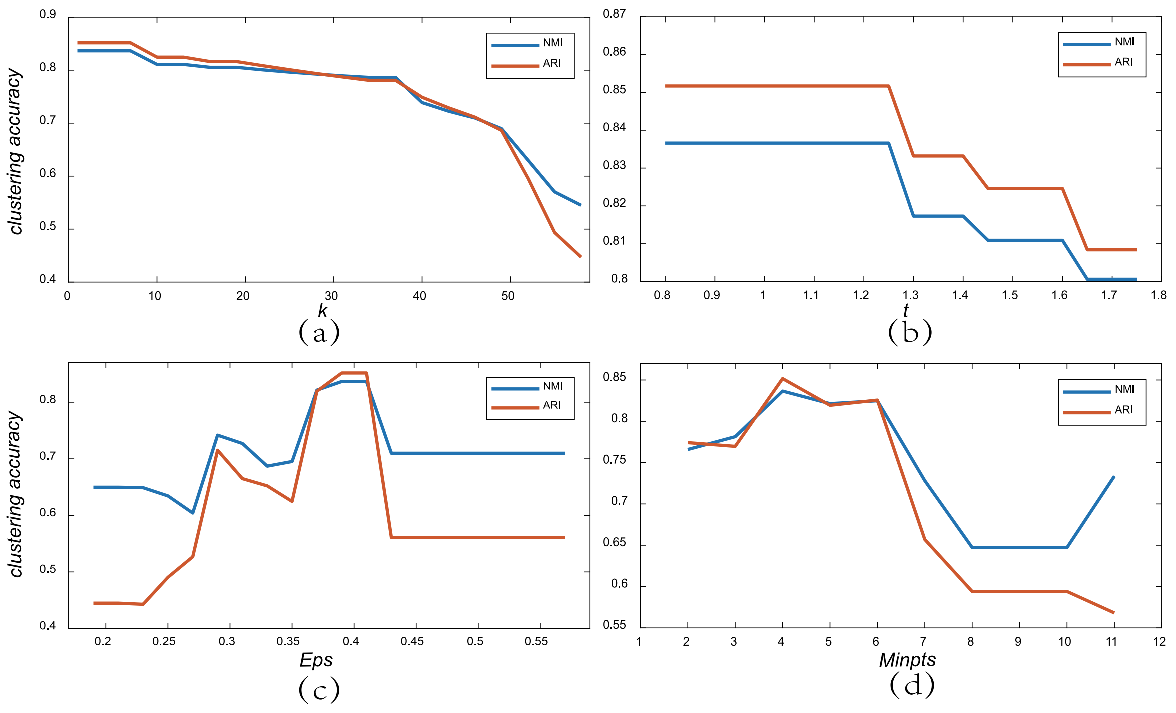


Fig. 20. For the dataset Iris: (a) effect of k value on clustering results. (b) effect of t value on clustering results. (c) effect of Eps value on clustering results. (d) effect of $Minpts$ value on clustering results.

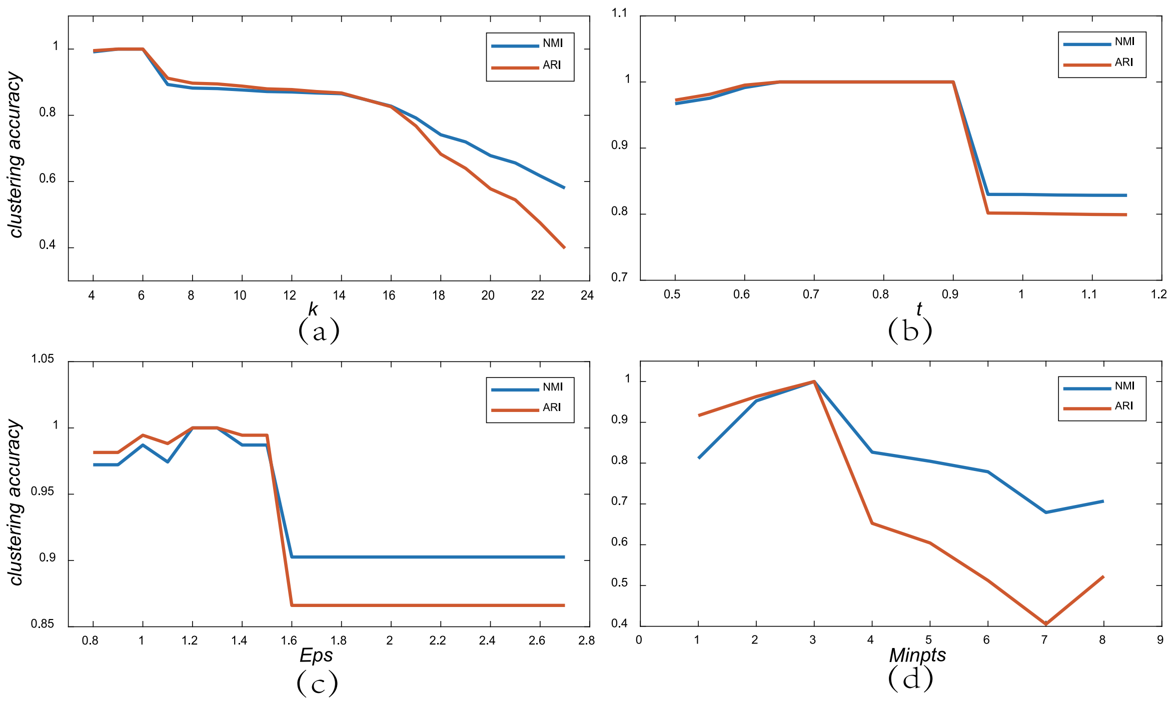


Fig. 21. For the dataset Compound: (a) effect of k value on clustering results. (b) effect of t value on clustering results. (c) effect of Eps value on clustering results. (d) effect of $Minpts$ value on clustering results.

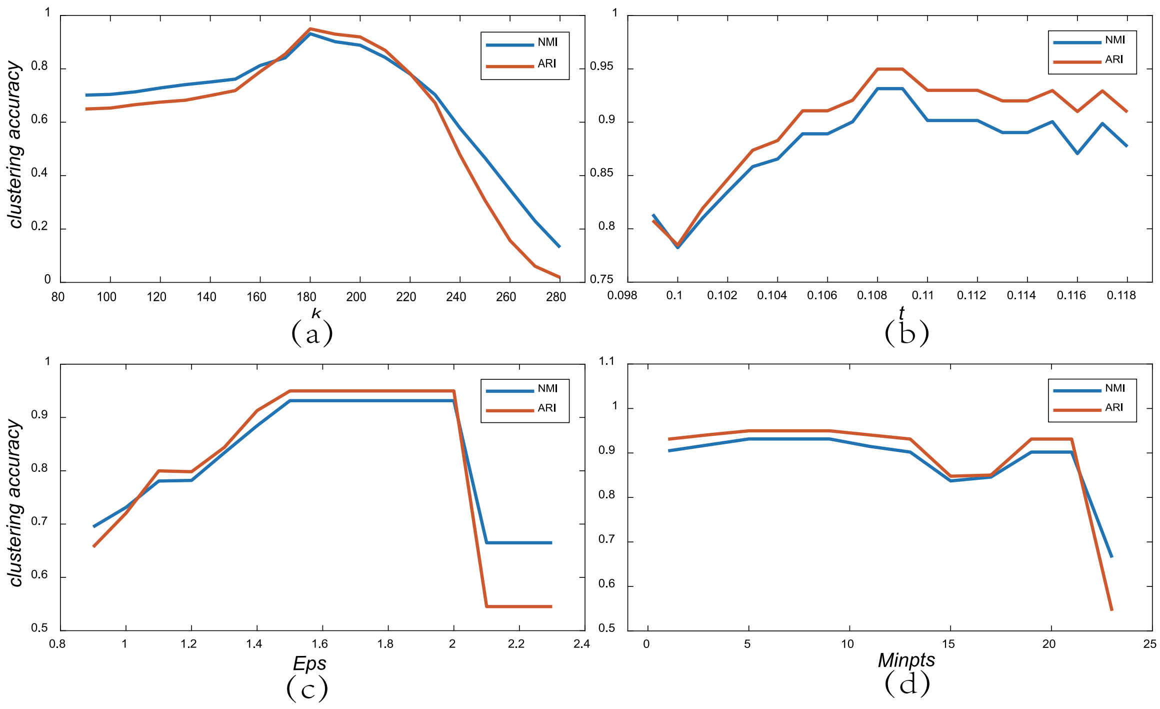


Fig. 22. For the dataset Pathbased: (a) effect of k value on clustering results. (b) effect of t value on clustering results. (c) effect of Eps value on clustering results. (d) effect of $Minpts$ value on clustering results.

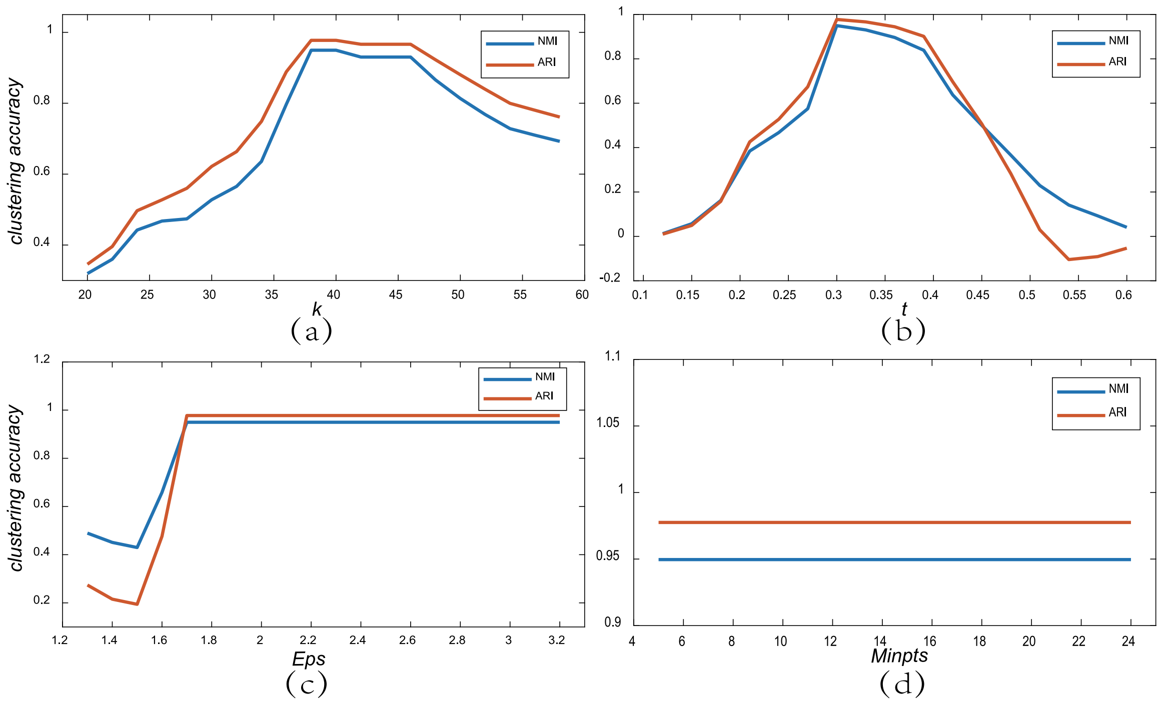


Fig. 23. For the dataset Jain: (a) effect of k value on clustering results. (b) effect of t value on clustering results. (c) effect of Eps value on clustering results. (d) effect of $Minpts$ value on clustering results.

datasets, it will face the problem that the adjustment range of parameter k is too large. Therefore, we will focus on finding the adaptive adjustment method of parameter k to simplify the execution steps of the algorithm. The other is the improvement of the SNNC method, it is worth noting that the effect of this method is not significant when the k value is too large, and will be improved in subsequent studies.

CRediT authorship contribution statement

Jiaxin Qian: Methodology, Software, Writing – original draft, Writing – review & editing. **You Zhou:** Writing – review & editing. **Xuming Han:** Supervision, Writing – review & editing. **Yizhang Wang:** Methodology, Software, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yizhang Wang reports administrative support was provided by Innovation and Entrepreneurship Program of Jiangsu Province (JSSCBS20211048). Yizhang Wang reports administrative support was provided by Postdoctoral Research Foundation of China (2022M713064). Yizhang Wang reports administrative support was provided by Lvyangjinfeng Excellent Doctoral Program of Yangzhou (YZLYJFJH2021YXBS105). Yizhang Wang reports administrative support was provided by Jiangsu Provincial Universities of Natural Science General Program (21JJB520021). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used in this paper are available on the Github repository https://github.com/mlyizhang/Clustering-Datasets.

Acknowledgments

This research is supported by the China Postdoctoral Science Foundation (Grant No. 2022M713064), Lvyangjinfeng Excellent Doctoral Program of Yangzhou (Grant No. YZLYJFJH2021YXBS105), Innovation and Entrepreneurship Program of Jiangsu Province, China (Grant No. JSSCBS20211048), and Jiangsu Provincial Universities of Natural Science General Program, China (Grant No. 21KJB520021).

References

[1] I.C. Gormley, T.B. Murphy, A.E. Raftery, Model-based clustering, Annu. Rev. Stat. Appl. 10 (2023) 573–595.
[2] B. Moseley, J.R. Wang, Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search, J. Mach. Learn. Res. 24 (1) (2023) 1–36.
[3] F. Murtagh, P. Contreras, Algorithms for hierarchical clustering: an overview, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 2 (1) (2012) 86–97.
[4] A.M. Ikotun, A.E. Ezugwu, L. Abualigah, B. Abuhaija, J. Heming, K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data, Inform. Sci. 622 (2023) 178–210.
[5] M. Ankerst, M.M. Breunig, H.P. Kriegel, J. Sander, OPTICS: Ordering points to identify the clustering structure, ACM Sigmod Rec. 28 (2) (1999) 49–60.
[6] M. Ester, H.P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, vol. 96, no. 34, 1996, pp. 226–231.
[7] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[8] E. Müller, Graph clustering with graph neural networks, J. Mach. Learn. Res. 24 (2023) 1–21.
[9] M. Meila, J. Shi, Learning segmentation by random walks, Adv. Neural Inf. Process. Syst. 13 (2000).
[10] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[11] T. Xiang, S. Gong, Spectral clustering with eigenvector selection, Pattern Recognit. 41 (3) (2008) 1012–1029.
[12] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Advances in Neural Information Processing Systems, vol. 14, 2001.
[13] S. Pei, F. Nie, R. Wang, X. Li, An efficient density-based clustering algorithm for face groping, Neurocomputing 462 (2021) 331–343.
[14] H. Mittal, A.C. Pandey, R. Pal, A. Tripathi, A new clustering method for the diagnosis of CoVID19 using medical images, Appl. Intell. 51 (2021) 2988–3011.
[15] G. Arauz-Garofalo, M. Jodar, M. Vilanova, A. de la Iglesia Rodriguez, J. Castillo, A. Soler-Ventura, R. Oliva, M. Vilaseca, M. Gay, Protamine characterization by top-down proteomics: Boosting Proteoform identification with DBSCAN, Proteomes 9 (2) (2021) 21.
[16] D. Cheng, R. Xu, B. Zhang, R. Jin, Fast density estimation for density-based clustering methods, Neurocomputing 532 (2023) 170–182.
[17] J.H. Kim, J.H. Choi, K.H. Yoo, A. Nasridinov, AA-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities, J. Supercomput. 75 (1) (2019) 142–169.
[18] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, DBSMOTE: density-based synthetic minority over-sampling technique, Appl. Intell. 36 (2012) 664–684.
[19] J. Xie, R. Wu, H. Wang, H. Chen, X. Xu, Y. Kong, W. Zhang, Prediction of cardiovascular diseases using weight learning based on density information, Neurocomputing 452 (2021) 566–575.
[20] R. Zhang, H. Peng, Y. Dou, J. Wu, Q. Sun, Y. Li, J. Zhang, P.S. Yu, Automating DBSCAN via deep reinforcement learning, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 2620–2630.
[21] Z. Wang, Z. Ye, Y. Du, Y. Mao, Y. Liu, Z. Wu, J. Wang, AMD-DBSCAN: An adaptive multi-density DBSCAN for datasets of extremely variable density, in: 2022 IEEE 9th International Conference on Data Science and Advanced Analytics, DSAA, IEEE, 2022, pp. 1–10.
[22] Y. Wang, Y. Yang, Relative density-based clustering algorithm for identifying diverse density clusters effectively, Neural Comput. Appl. 33 (2021) 10141–10157.
[23] R.J. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Trans. Knowl. Discov. Data (TKDD) 10 (1) (2015) 1–51.
[24] W. Lai, M. Zhou, F. Hu, K. Bian, Q. Song, A new DBSCAN parameters determination method based on improved MVO, IEEE Access 7 (2019) 104085–104095.
[25] A. Latifi-Pakdehi, N. Daneshpour, DBHC: A DBSCAN-based hierarchical clustering algorithm, Data Knowl. Eng. 135 (2021) 101922.
[26] A. Smiti, Z. Elouedi, Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques, in: 2012 IEEE 16th International Conference on Intelligent Engineering Systems, INES, IEEE, 2012, pp. 573–578.
[27] K.M. Kumar, A.R.M. Reddy, A fast DBSCAN clustering algorithm by accelerating neighbor searching using groups method, Pattern Recognit. 58 (2016) 39–48.
[28] N. Gholizadeh, H. Saadatfar, N. Hanafi, K-DBSCAN: An improved DBSCAN algorithm for big data, J Supercomput. 77 (2021) 6214–6235.
[29] Y. Zhu, K.M. Ting, M.J. Carman, Density-ratio based clustering for discovering clusters with varying densities, Pattern Recognit. 60 (2016) 983–997.
[30] H. Darong, W. Peng, Grid-based DBSCAN algorithm with referential parameters, Physics Procedia 24 (2012) 1166–1170.
[31] R. Scitovski, K. Sabo, DBSCAN-like clustering method for various data densities, Pattern Anal. Appl. 23 (2) (2020) 541–554.
[32] D. Birant, A. Kut, ST-DBSCAN: An algorithm for clustering spatial–temporal data, Data Knowl. Eng. 60 (1) (2007) 208–221.
[33] M. Meilă, Comparing clusterings—an information based distance, J. Multivariate Anal. 98 (5) (2007) 873–895.
[34] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1985) 193–218.
[35] A. Bryant, K. Cios, RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates, IEEE Trans. Knowl. Data Eng. 30 (6) (2017) 1109–1121.
[36] R. Scitovski, K. Sabo, F. Martínez-Álvarez, Š. Ungar, Cluster Analysis and Applications, Springer, 2021.

Jiaxin Qian: She is currently pursuing for her master degree at Yangzhou University, China. Her research interests include clustering algorithms, federated learning, and machine learning.

You Zhou: He received his bachelor and Ph.D. degree from Jilin University in 2002 and 2008, respectively. He is now a professor of the College of Computer Science and Technology at Jilin University, Changchun, China. His research interests include Machine Learning, Pattern Recognition and Bioinformatics.

Xuming Han: He received the M.S. degree from Tianjin University, China, and the Ph.D. degree from the Jilin University, China. He is currently a Professor with the College of Computer Federation. He has published more than 40 research articles. His research interests include big data analysis, machine learning, collaborative intelligence and optimization.

Yizhang Wang: He received Ph.D. degree from Jilin University in 2020. He joined Yangzhou University in 2020, as a lecturer. He has published 10+ papers in international journals such as Pattern Recognition, Information Sciences, and Knowledge-Based Systems. His research interests include density clustering and federated clustering.