

# AMD-DBSCAN: An Adaptive Multi-density DBSCAN for datasets of extremely variable density

Ziqing Wang<sup>1</sup>, Zhirong Ye<sup>1</sup>, Yuyang Du<sup>1</sup>, Yi Mao<sup>1</sup>, Yanying Liu<sup>1</sup>, Ziling Wu<sup>2</sup>, Jun Wang<sup>3</sup>

School of Microelectronics Science and Technology

<sup>1</sup>Sun Yat-sen University

{wangzq37, yezhr6, duyy27, maoy28, liuyy336}@mail2.sysu.edu.cn

<sup>2</sup>University of Nottingham

psxzw11@nottingham.ac.uk

<sup>3</sup>Zhuhai Jieli Technology Co.,Ltd

jimmy.junwang@gmail.com

**Abstract**—DBSCAN has been widely used in density-based clustering algorithms. However, with the increasing demand for Multi-density clustering, previous traditional DBSCAN can not have good clustering results on Multi-density datasets. In order to address this problem, an adaptive Multi-density DBSCAN algorithm (AMD-DBSCAN) is proposed in this paper. An improved parameter adaptation method is proposed in AMD-DBSCAN to search for multiple parameter pairs (i.e., Eps and MinPts), which are the key parameters to determine the clustering results and performance, therefore allowing the model to be applied to Multi-density datasets. Moreover, only one hyperparameter is required for AMD-DBSCAN to avoid the complicated repetitive initialization operations. Furthermore, the variance of the number of neighbors (VNN) is proposed to measure the difference in density between each cluster. The experimental results show that our AMD-DBSCAN reduces execution time by an average of 75% due to lower algorithm complexity compared with the traditional adaptive algorithm. In addition, AMD-DBSCAN improves accuracy by 24.7% on average over the state-of-the-art design on Multi-density datasets of extremely variable density, while having no performance loss in Single-density scenarios.

## I. INTRODUCTION

DBSCAN [1] is one of the most widely used density-based clustering methods in data mining [2]. Since objects within a cluster are similar and objects in different clusters are dissimilar [3], DBSCAN is able to classify the objects with the similar characteristics into one cluster [4] and to identify clusters of different shapes and sizes accurately from noise [5]. Due to its specialties, it is widely used in various fields such as ship detection [6], wafer classification [7], astronomy [8], robotics [9] and disease diagnosis [10].

However, traditional DBSCAN has two drawbacks. Firstly, DBSCAN clustering relies on two parameters (i.e., Eps and MinPts). For a point, *Eps* is the radius of the circle whose center is that point. All the points in that circle could be considered as neighbors of that point. *MinPts* is a threshold value that can be utilized to search core points during the process of DBSCAN clustering. However, in most cases, the

dataset has a dimensionality greater than three dimensions, which leads to the inability of the visualization. Therefore, it is hard to determine these two parameters, and it would take a lot of time and effort to modify the parameters artificially. And when the dimensionality of the data is high, there will be a curse of dimensionality [11], leading to a lower clustering accuracy.

Furthermore, traditional DBSCAN only offers a single parameter pair and it has a low accuracy on Multi-density datasets of extremely variable density. To be more specific, the density of each cluster varies greatly. Giving parameters at the density of the sparse clusters would lead to merging similar datasets, while giving parameters at the density of the dense clusters would result in many clusters with less density being incorrectly identified as noise. Therefore, traditional DBSCAN that uses only a fixed parameter pair could not give good clustering results on Multi-density datasets where different regions have various densities.

Some algorithms (e.g., YADING [12]) propose Multi-density DBSCAN to address this problem. Specifically, the data points with the highest density are firstly clustered, then the data points that have been clustered are saved, and then the remaining points that have not been clustered are continued to be clustered. Repeat the above process until all points are clustered. This means that different density clusters should utilize different parameter pairs. To obtain multiple parameter pairs, most of these algorithms utilize the  $k_{dis}$  curve. To determine a  $k_{dis}$  curve, the Euclidean distance matrix, a square matrix containing the euclidean distances between the elements of a set, is supposed to be obtained first and sorted in ascending order. The  $k_{dis}$  curve is comprised of the values of the  $k^{th}$  column of this sorted distance matrix. And  $k$  means  $k^{th}$  neighbor of a point and different  $k$  values determine different curves, resulting in different parameter pairs. However, these methods mostly use a fixed  $k$ , which is not well based on the distribution properties of the dataset. In addition, to search for multiple candidate parameter pairs, they introduce additional hyperparameters that are required to be set manually.

To enable DBSCAN to be applied to Multi-density datasets of extremely variable density, AMD-DBSCAN is proposed.

This work was supported by Key-Area R&D Program of Guangdong (2019B010135002), Innovative & Enterprising Team of Zhuhai (2019ZHCDGY07), and University-enterprise project (K20-76220-002).

<sup>3</sup> Contact Author

To the best of our knowledge, the proposed approach in this paper is a novel exploration of the  $k_{dis}$  value, which is the  $k^{th}$  nearest neighbor of a point, to search for multiple parameter pairs matching the distribution of the dataset.

The contributions of this paper are summarized as follows:

- 1) An improved parameter adaptation method is proposed to locate the adaptive  $k$  based on the distribution of the datasets and the binary search algorithm is utilized to speed up this adaptive process.
- 2) A new method for utilizing the  $k_{dis}$  value is proposed to search for multiple candidate  $Eps$ .
- 3) An adaptive Multi-density clustering algorithm is proposed to provide matching parameter pairs (i.e.,  $Eps$  and  $MinPts$ ) for each density cluster.

## II. RELATED WORK

The clustering results of DBSCAN are significantly affected by two parameters (i.e.,  $Eps$  and  $MinPts$ ) and the distribution of datasets. As for parameters, larger  $MinPts$  and smaller  $Eps$  lead to incomplete clustering (many core points are considered as noise). In contrast, small  $MinPts$  and large  $Eps$  result in over-clustering (two or more clusters of points are clustered into a single cluster). In the case of datasets' distribution, only one pair of  $Eps$  and  $MinPts$  is unlikely to cope with the Multi-density datasets whose points are not evenly distributed. In this section, previous work on these two aspects is discussed and compared with AMD-DBSCAN.

### A. Configuration of parameters

To obtain appropriate parameters for DBSCAN, many approaches have been proposed. These methods can be divided into two categories, adapting one of the two parameters (i.e.,  $Eps$  and  $MinPts$ ) and configuring both of them.

1) **Single Parameter:** Reversing the nearest neighbor has been proposed by method [13] to estimate the density around a point for adapting  $MinPts$  automatically. Besides, the  $k_{dis}$  curve, which is comprised of the distance of points of the dataset and their  $k$ -nearest neighbors in ascending order, is leveraged by [12], [14], [15] to adapt appropriate  $Eps$  by locating the flat part of the curve. However, the analysis of the  $k_{dis}$  curve proposed by [12], [15] requires another two parameters and is with a relatively large complexity ( $O(nlog(n))$ ). ISB-DBSCAN [16] algorithm utilizes an input parameter  $k$  as the number of the nearest neighbor to reduce the input parameters of DBSCAN. AA-DBSCAN [17] algorithm utilizes the approximate adaptive  $Eps$  for each density. Hence, it can find the clusters in the Multi-density datasets.

However, all the approaches above can only adapt one parameter automatically. The advantage of the AMD-DBSCAN over the above algorithms is that our method can adapt both parameters at the same time.

2) **Multiple Parameters:** The following method is proposed to provide two parameters. In GMDBSCAN [18], centers of clusters are generated by GD to adapt both two parameters. However, it has a lower accuracy compared to AMD-DBSCAN when dealing with the Multi-density datasets.

### B. Properties of Datasets

1) **Multi-density Datasets:** Many algorithms are proposed for tackling the situation of Multi-density datasets with unevenly distributed points. For instance, VDBSCAN [19] proposed to automatically customize clustering parameters for different density regions. Furthermore, AEDBSCAN [14] improves the  $k_{dis}$  curve by using the second-order difference method to improve the accuracy of the Multi-density clustering. Also, YADING [12] estimates the density by determining the  $Eps$  based on VDBSCAN [19], and it optimizes the clustering speed. DVBSAN [20] can deal with local density variation within a cluster, but it cannot determine parameters automatically. HDBSCAN [21] is a hierarchical clustering method that allows it to perform well on Multi-density datasets. However, it has a long execution time.

Since AMD-DBSCAN can adapt a parameter pair for each layer during the Multi-density DBSCAN process, it can achieve good clustering results on Multi-density datasets.

2) **Large-scale Datasets:** Some algorithms are proposed to be applied under large-scale datasets. Some methods [22]–[24] use a partitioning strategy and a distributed structure that allow them to be applied to large-scale datasets. SDBSCAN algorithm [25] combines sampling techniques with DBSCAN for clustering large spatial databases. In IDBSCAN [26] method, the greater I/O cost and memory requirements involved in clustering are addressed by using marked boundary objects to directly scale the computation without the need for actual dataset selection.

However, all of these methods above have a low accuracy on Multi-density datasets. AMD-DBSCAN can have good performance in this scenario because it can adapt two parameters according to the distribution of large-scale datasets.

## III. PROPOSED ALGORITHM

### A. Overview

In this section, the details of the three steps of AMD-DBSCAN are analyzed and the complexity of the overall algorithm is given at the end. Figure 1 gives the information of the framework proposed in this paper.

- 1) **Parameter Adaptation of  $k$ :** To achieve Multi-density clustering on the datasets of extremely variable density, it is first necessary to obtain  $k$  required to determine the  $k_{dis}$  value. An improved parameter adaptation method is proposed to locate  $k$ . The spatial distribution properties of the dataset itself are utilized to generate a list of candidate  $Eps$  and  $MinPts$  parameters, which are required by DBSCAN. The  $Eps$  and  $MinPts$  parameter lists are sequentially input into the DBSCAN to obtain the number of clusters. After that, by using the binary search algorithm,  $Eps$  and  $MinPts$  pairs that are the most match the distribution of the dataset for clustering is selected, where  $MinPts$  is used as  $k$  required for the next step.
- 2) **To obtain Candidate Eps List:** The  $k_{dis}$  value, which is defined as  $k^{th}$  neighbor of a point, is determined by

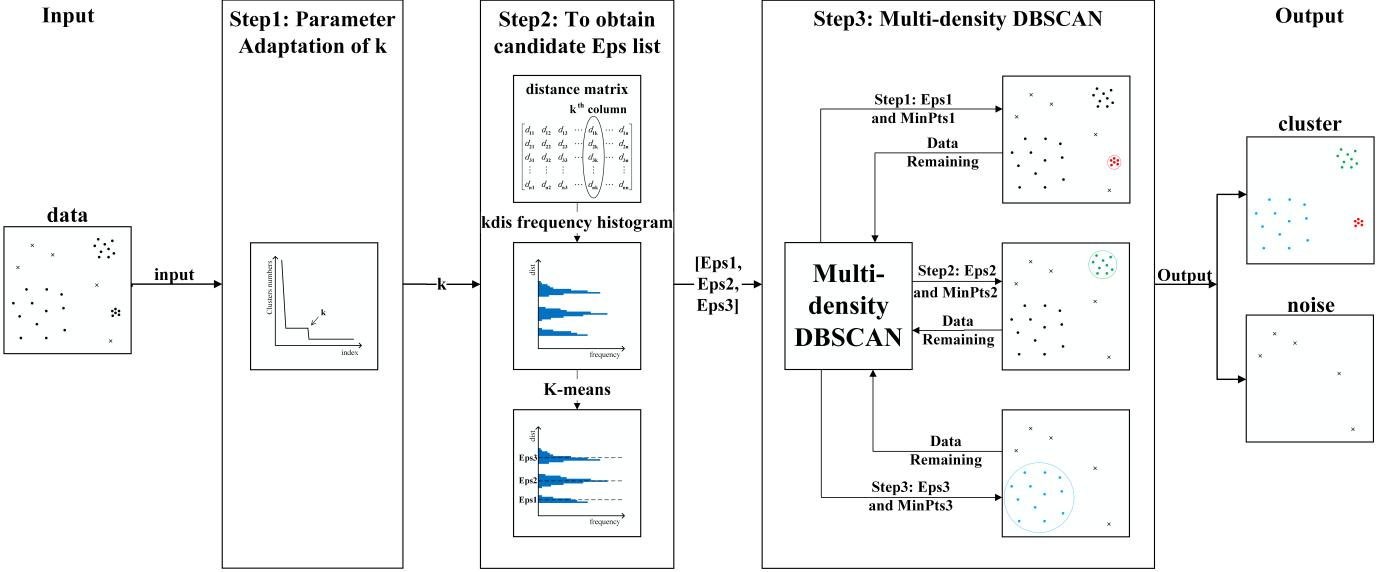


Fig. 1. Framework of AMD-DBSCAN

$k$  derived from the last step. And the  $k_{dis}$  frequency histogram can be obtained by a series of  $k_{dis}$  values arranged in ascending order. The K-means algorithm is utilized to cluster the  $k_{dis}$  values to obtain  $N$  candidate  $Eps$ , where  $N$  refers to the number of peaks by directly observing the  $k_{dis}$  frequency histogram.

- 3) **Multi-density Clustering:** Multi-density DBSCAN can perform well on Multi-density datasets and it is introduced in this study. Each  $Eps$  in the obtained *CandidateEpsList* is sorted in ascending order and used to generate the corresponding *MinPts*. The parameters of each layer are input into Multi-density clustering respectively to obtain the final results.

#### B. Parameter Adaptation of $k$

There are many algorithms [12], [15] that utilize  $k_{dis}$  value, which represents the  $k^{th}$  nearest neighbor of each point, to obtain the important parameter  $Eps$  for clustering. They require manual input of the parameter  $k$ . And our model proposes a novel exploration of  $k_{dis}$  value using the  $k_{dis}$  frequency histogram. The clustering results depend directly on the selection of the parameter  $k$ .

According to our experiments shown in the latter section, a small  $k$  leads to a large number of clusters (what should be one large cluster is clustered into many small clusters), while a large  $k$  results in a small number of clusters (what should be multiple clusters are clustered into one cluster). Therefore,  $k$  affects the clustering results significantly.

To avoid the randomness of manually setting  $k$  during clustering, a parameter adaptation algorithm is proposed in this study to search for  $k$ . This approach adapts a  $k$  that responds to the distribution properties of the dataset. Our subsequent experiments demonstrate that this adaptive  $k$  is an important guide for determining the  $k_{dis}$  value, making parameter pairs

can have good performance for Multi-density clustering. The specific process of parameter adaptation of  $k$  is as follows.

- 1) **To compute Eps List:** First of all, the Euclidean distance matrix  $DIST_{n \times n}$  of the dataset  $D$  is calculated:

$$DIST_{n \times n} = \{dist(i, j), 1 \leq i \leq n, 1 \leq j \leq n\} \quad (1)$$

where  $n = |D|$  represents the number of points in the dataset  $D$ ,  $DIST_{n \times n}$  is a symmetric matrix with  $n$  rows and  $n$  columns, and each element represents the Euclidean distance from  $i^{th}$  point to  $j^{th}$  point in the dataset  $D$ .

By arranging each row in  $DIST_{n \times n}$  in ascending order, a  $SORTED\_DIST_{n \times n}$  is obtained. The distance data of the  $k^{th}$  column in  $SORTED\_DIST_{n \times n}$  is noted as vector  $D_k$ . And  $D_k$  is obtained by averaging the data in the vector  $D_k$ , which yields a list of  $Eps$  as shown in Equation 2.

$$EpsList = \{\overline{D}_k, 1 \leq k \leq n\} \quad (2)$$

The whole process is encapsulated as a function *obtainEpsList(data)* which will be used in the pseudo-code. The input is the dataset  $D$  and the output is *EpsList*.

- 2) **To compute MinPts List:** For a given *EpsList*, each  $Eps$  value in it is utilized to calculate a *MinPts*, which is the number of neighbors corresponding to each  $Eps$ . And *MinPtsList* consists of multiple *MinPts*, as shown in Equation 3 and Equation 4.

$$MinPtsList = \{MinPts_j, 1 \leq j \leq n\} \quad (3)$$

$$MinPts_j = \frac{1}{n} \sum_{i=1}^n Point_i \quad (4)$$

where  $Point_i$  is the number of neighbors of the  $i^{th}$  point within the range of  $Eps_j$ , and  $n$  represents the number of points in the dataset  $D$ .

The whole procedure above is encapsulated as a function  $obtainMinPtsList(data, EpsList)$ . The input are the dataset  $D$  and  $EpsList$  and the output is  $MinPtsList$ .

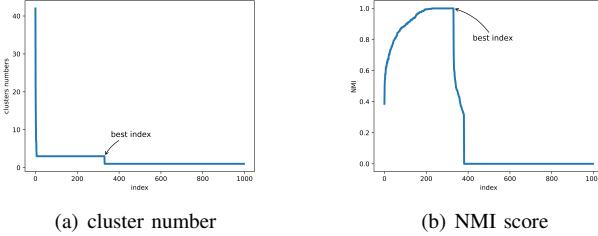


Fig. 2. **Variation of the clusters numbers and NMI score.** (a) Best index is the maximum index when the number of clusters is stable for the first time. (b) The index of the highest NMI score is exactly equal to the best index.

**3) To locate the adaptive  $k$ :** The algorithm is based on two conclusions obtained by analyzing the image of the number of clusters and the image of the effect of clustering.

1)  $EpsList$  is in ascending order, which means that as the  $index$  increases,  $Eps$  increases. An experiment is designed in which each pair of  $Eps$  and  $MinPts$  parameters are input sequentially into DBSCAN to obtain the number of clusters. By observing the change in the number of clusters with index, our experimental results show that the number of clusters decreases monotonically with increasing  $index$ , as shown in Figure 2(a). That is because as  $Eps$  increases, more points in the dataset are clustered into one class, and thus the number of clusters is monotonically decreasing.

2) When the number of clusters is stable for the first time, which means the number of clusters is the same three times in a row shown in the Figure 2(a), the point with the largest index in this stable part of the curve corresponds to the best clustering. Normalized Mutual Information (NMI) [27] is utilized to judge the clustering effect. An experiment is designed whereby the NMI of each DBSCAN result is recorded. By analyzing the change in NMI with index, our experimental results shown in Figure 2(b) indicate that the NMI increases until reaches the largest index when the number of clusters is stable, which is noted as the best index.

Based on the above conclusions, an improved parameter adaptation method is proposed. The specific implementation is shown below. The  $EpsList$  and  $MinPtsList$  are chosen as the  $Eps$  candidates and  $MinPts$  parameters, respectively. And then each parameter pair is input into DBSCAN for clustering to obtain the number of clusters. As illustrated in Algorithm 1, when the number of clusters is the same three times in a row for the first time, the clustering result is considered to be stable. The number of clusters  $n$  is considered to represent the number of true clusters of the dataset. And the best index where the NMI value is the greatest is the largest index when the number of clusters is equal to  $n$ .

According to the first conclusion, the number of clusters decreases monotonically with  $index$ . To avoid traversing all

combinations of parameters and reduce the time significantly, the binary search algorithm is utilized to locate the best index. The search array is from the first index equal to  $n$  to the last index. The search process starts from the middle index. If the number of clusters corresponding to the middle index is exactly  $n$ , the search starts from the right half of the array. If it is less than  $n$ , the search starts from the left half of the array. Repeat this process until the largest index that the number of clusters is equal to  $n$  is located, that is, the best index. Finally, the  $MinPts$  corresponding to the best index in the  $MinPtsList$  is our adaptive  $k$ .

The pseudo-code of locating  $k$  is shown in Algorithm 1.

---

#### Algorithm 1: Parameter Adaptation of $k$

---

```

input : data
output: k
1  $Eps \leftarrow obtainEpsList(data)$ 
2  $MinPts \leftarrow obtainMinPtsList(data, Eps)$ 
3  $counter \leftarrow 0$ 
4 for  $i \leq len(data) - 1$  do
5    $ClusterNum[i] \leftarrow DBSCAN(Eps[i], MinPts[i])$ 
6   if  $ClusterNum[i] == ClusterNum[i + 1]$  then
7     |  $counter \leftarrow counter + 1$ 
8   end
9   else
10    |  $counter \leftarrow 0$ 
11  end
12 if  $counter > 3$  then
13   |  $n \leftarrow ClusterNum[i]$ 
14   |  $left \leftarrow i$ 
15   |  $right \leftarrow len(data) - 1$ 
16   | while  $start \leq end$  do
17     | |  $mid \leftarrow (start + end)/2$ 
18     | | if  $DBSCAN(Eps[mid], MinPts[mid]) < n$ 
19       | | | then
20         | | | |  $right \leftarrow mid$ 
21       | | | end
22     | | else if
23       | | |  $DBSCAN(Eps[mid], MinPts[mid]) > n$  then
24         | | | |  $Left \leftarrow mid$ 
25       | | | end
26     | | else
27       | | |  $best\ index \leftarrow mid$ 
28       | | |  $k \leftarrow MinPts[best\ index]$ 
29     | | end
30   | end
31 end

```

---

#### C. To obtain Candidate $Eps$ List

To have a good performance on Multi-density datasets, multiple  $Eps$  are required for Multi-density clustering. This part is to obtain the candidate  $Eps$  list. In the step above, an adaptive  $k$  is obtained based on the distribution property of the dataset  $D$ , which is utilized to determine the  $k_{dis}$  value.

To better locate candidate  $Eps$ , several algorithms (e.g., YADING [12]) have been proposed. These algorithms look for the inflection point (i.e., the part where the second-order differential is zero) of the  $k_{dis}$  curve. In YADING [12] method,

the flat part of the  $k_{dis}$  curve indicates that the density of points is consistent, while the steep part indicates that the density of points is significantly different. However, our experimental results show that the candidate  $Eps$  list derived from such methods does not give good clustering results. Furthermore, these methods introduce additional hyperparameters.

Inspired by YADING [12], our AMD-DBSCAN proposes a new method of searching for candidate  $Eps$  using the  $k_{dis}$  frequency histogram, which is a frequency histogram of the  $k_{dis}$  values. The peak of the  $k_{dis}$  frequency histogram corresponds to the flat part of the  $k_{dis}$  curve. Therefore, the  $k_{dis}$  value of the peak of the histogram can be considered as a candidate  $Eps$  because it ensures that there are a large number of points in this range that can be clustered into one class. To automatically distinguish between different peaks in the frequency histogram, the K-means algorithm is utilized to divide the  $k_{dis}$  frequency histogram into different parts. And the clustering center of each part is considered as the candidate  $Eps$ . This process will allow more similar  $k_{dis}$  values to be clustered in one class, and it is more representative of the  $Eps$  of this part of the points.

The specific steps are as follows. The  $k_{dis}$  value can be determined by using the adaptive  $k$  derived from the last step and the  $k_{dis}$  frequency histogram can be obtained by a series of  $k_{dis}$  values.  $N$  is considered as the number of clustering centers of the  $k_{dis}$  values which is determined by observing the number of peaks of the  $k_{dis}$  frequency histogram. And this  $N$  is exactly equal to the number of clustering centers required for the K-means algorithm (i.e.,  $K$ ). Therefore, it is straightforward to utilize the K-means algorithm because  $K$  required by the K-means algorithm can be easily located. Moreover, our experiments in the latter section show that it is also effective because it has higher clustering accuracy. For example, as shown in Figure 1, it can be observed that there are three peaks in the frequency histogram. Therefore,  $N$  is equal to 3, which means that there are three candidate  $Eps$ . The K-means algorithm with  $K$  equal to 3 can be used to cluster the  $k_{dis}$  values. As shown in Figure 1, the  $k_{dis}$  values corresponding to the center of the clustering result are considered as the candidate  $Eps$ . After the above process,  $N$  candidate  $Eps$  can be obtained, which means that most of the similar  $k_{dis}$  values are clustered around these  $N$  candidate  $Eps$ . These candidate  $Eps$  can be used for Multi-density clustering.

The pseudo-code for obtaining candidate  $Eps$  list is shown in Algorithm 2.

#### D. Multi-density Clustering

Once obtaining the candidate  $Eps$  list, the next step is to perform Multi-density DBSCAN on the dataset  $D$ . Unlike the algorithm of YADING [12], which sets all  $MinPts$  to  $k$  by default, our algorithm calculates  $MinPts$  based on the distribution property of the dataset.  $MinPts$  is calculated by the same algorithm as parameter adaptation, using the Equation 4, which utilizes a candidate  $Eps$  and  $data$  to

---

#### Algorithm 2: To obtain Candidate Eps List

---

```

input : data
          $k \leftarrow ParameterAdaptation(data)$ 
output: CandidateEpsList: candidate Eps list for
         Multi-density DBSCAN
1 distances  $\leftarrow sort(euclidean\_distances(data))$ 
2  $k_{dis} \leftarrow distances[k]$ 
3  $N \leftarrow$  Number of peak of the  $k_{dis}$  frequency histogram
4 CandidateEpsList  $\leftarrow$  K-means( $k_{dis}, K=N$ ).centers
5 return CandidateEpsList
```

---

obtain  $MinPts$ . This process is encapsulated as the function *obtainMinPts*(*data*, *Eps*).

The specific process is as follows. The first step is to sort the obtained candidate  $Eps$  list in ascending order, and then the *obtainMinPts* function is utilized to obtain the adaptive  $MinPts$ . Then, *data*, *Eps*, *MinPts* are input into DBSCAN for clustering. After that, the *data* that has been clustered is saved and is not clustered in the next loop. Repeat the above steps until all the *Eps* are input into the model, and the remaining data is the noise. The whole clustering is finished.

The pseudo-code for Multi-density DBSCAN is shown in Algorithm 3.

---

#### Algorithm 3: Multi-density DBSCAN

---

```

input : data
         EpsList  $\leftarrow obtainEpsCandidateList(data, k)$ 
output: cluster and noise
1 for Eps in EpsList do
2    $MinPts \leftarrow obtainMinPts(data, Eps)$ 
3   cluster  $\leftarrow DBSCAN(data, Eps, MinPts)$ 
4   Mark data with labels
5   Remove cluster from data
6 end
7 Mark data as noise
8 return cluster and noise
```

---

#### E. Algorithm Complexity Analysis

For a data set containing  $n$  points, the complexity of DBSCAN is  $O(n^2)$ . By adopting the divide-and-conquer strategy, the complexity of DBSCAN is  $O(n \log(n))$ . For uniformity, the complexity of DBSCAN is denoted as  $O(f(n))$ . In the process of parameter adaptation, by using the binary search algorithm, the complexity is  $O(\log(n))$ , and thus the complexity of the whole parameter adaptation is  $O(\log(n)f(n))$ . In the process of locating the candidate  $Eps$ , the complexity of K-means algorithm is  $O(n)$ . The complexity of Multi-density DBSCAN in the final clustering process is also  $O(f(n))$ . In summary, the time complexity of AMD-DBSCAN is  $O(\log(n)f(n)) + O(n) + O(f(n)) = O(\log(n)f(n))$ .

## IV. EXPERIMENT AND ANALYSIS

In the experimental section, the datasets and evaluation metrics are introduced, then the AMD-DBSCAN algorithm is applied to Single-density and Multi-density datasets respectively to compare with different algorithms, and finally an ablation study is done to justify each step of our algorithm.

### A. Experiment platform and dataset

The experimental platform in this paper is an AMD Ryzen 7 5800H processor with 16GB of RAM. To verify the performance of our algorithm, some classical algorithms are reproduced and compared with our algorithm in the same experimental platform.

In order to measure the difference in density between different clusters in a dataset, a new metric, the variance of the number of neighbors (VNN), is proposed. Equation 2 is utilized to obtain an *EpsList*. The first element of the *EpsList* is taken as *Eps1*, which is the average distance between all points and their nearest neighbors. And *Eps1* is utilized as the search radius to search the number of neighbors within *Eps1* of each data point. Then the variance of the number of neighbors (VNN) is shown in Equation 5.

$$VNN = Var(neighbors_i), 1 \leq i \leq n \quad (5)$$

where  $n$  represents the number of points of the dataset, and  $neighbors_i$  represents the number of neighbors within  $Eps1$  of  $i^{th}$  point.

The value of VNN is a metric that reflects the difference in density of the dataset. For Single-density datasets, the number of neighbors for each point within a given search radius is similar, resulting in a smaller value of VNN. In contrast, for Multi-density datasets with large density differences, the number of neighbors is large for dense clusters and small for sparse clusters, leading to a large value of VNN. Therefore, the larger the value of VNN is, the greater the density difference of datasets is. In this paper, datasets with VNN less than 10 are considered as Single-density datasets, while datasets with VNN greater than 10 are considered as Multi-density datasets, especially datasets with VNN greater than 100 are considered as extreme Multi-density datasets.

Several datasets are selected for our experiments from UCI [28]. And because the Multi-density datasets of extremely variable density are hard to obtain, some datasets are generated to better test the performance of our algorithm. The *make\_blobs1* to *make\_blobs8* datasets are generated by scikit-learn [29]. *make\_blobs1* serves as a base dataset where each cluster has the same amount of points and varies greatly in density. To test the robustness of the algorithm, some changes are made to the base dataset.

The details of the datasets are shown in Table I.

### B. Evaluation Metrics

There are two metrics utilized to evaluate the clustering results.

- 1) Normalized Mutual Information (NMI) [27] is utilized to evaluate the clustering results. NMI is between [0,1] and is used to measure the similarity of the clustering results, the larger the value, the better the clustering results.
- 2) The accuracy of clustering is utilized as the second metric. Since all datasets have labels, the accuracy is the percentage of correct labels after clustering.

TABLE I  
DATASET PROPERTIES

dataset	size	clusters	VNN	Multi-density
Aggregation	788	7	0.49	FALSE
Compound	399	5	5.66	FALSE
D31	3100	31	1.76	FALSE
Flame	240	2	1.12	FALSE
R15	600	15	1.62	FALSE
make_blobs1	4998	6	1726	TRUE
make_blobs2	5048	6	3743	TRUE
make_blobs3	4998	6	2953	TRUE
make_blobs4	4998	6	645	TRUE
make_blobs5	4998	6	4467	TRUE
make_blobs6	4998	6	127	TRUE
make_blobs7	4998	6	4100	TRUE
make_blobs8	4998	6	5600	TRUE
unbalance	6500	7	17	TRUE

### C. Parameter Adaptation

The experiment is designed to compare the execution time of our parameter adaptation approach with the traditional parameter adaptation approach PDDBSCAN [30] on different datasets. Two metrics are utilized, one is the shortest execution time and the other is the average execution time of 100 rounds of experiments.

The detailed results of the experiments are illustrated in Table II. As can be seen from Table II, after adopting the binary search algorithm, the speed of our algorithm is on average 4 times faster than PDDBSCAN [30], which does not use the binary search algorithm. In particular, the speed of AMD-DBSCAN is much faster in some datasets where the same number of clusters occurs many times because the binary search algorithm can speed up the search process significantly. For example, in the *unbalance* dataset, most of the points in this dataset are clustered in three classes. After using our algorithm, the execution time is reduced by nearly 93%. Therefore, AMD-DBSCAN can speed up the execution time of the parameter adaptive process significantly on this kind of Multi-density dataset with a large number of points.

TABLE II  
COMPARISONS OF EXECUTION TIME

dataset	PDDBSCAN		AMD-DBSCAN	
	t_min	t_average	t_min	t_average
Aggregation	0.275	0.291	0.165	0.191
Compound	0.049	0.052	0.031	0.041
D31	2.550	2.820	1.350	2.330
Flame	0.042	0.045	0.021	0.024
R15	0.225	0.234	0.076	0.087
unbalance	95.658	97.782	6.407	6.552

### D. Single-density Datasets

To compare the performance on Single-density datasets (i.e., datasets with VNN less than 10) for our proposed AMD-DBSCAN, some Single-density clustering approaches

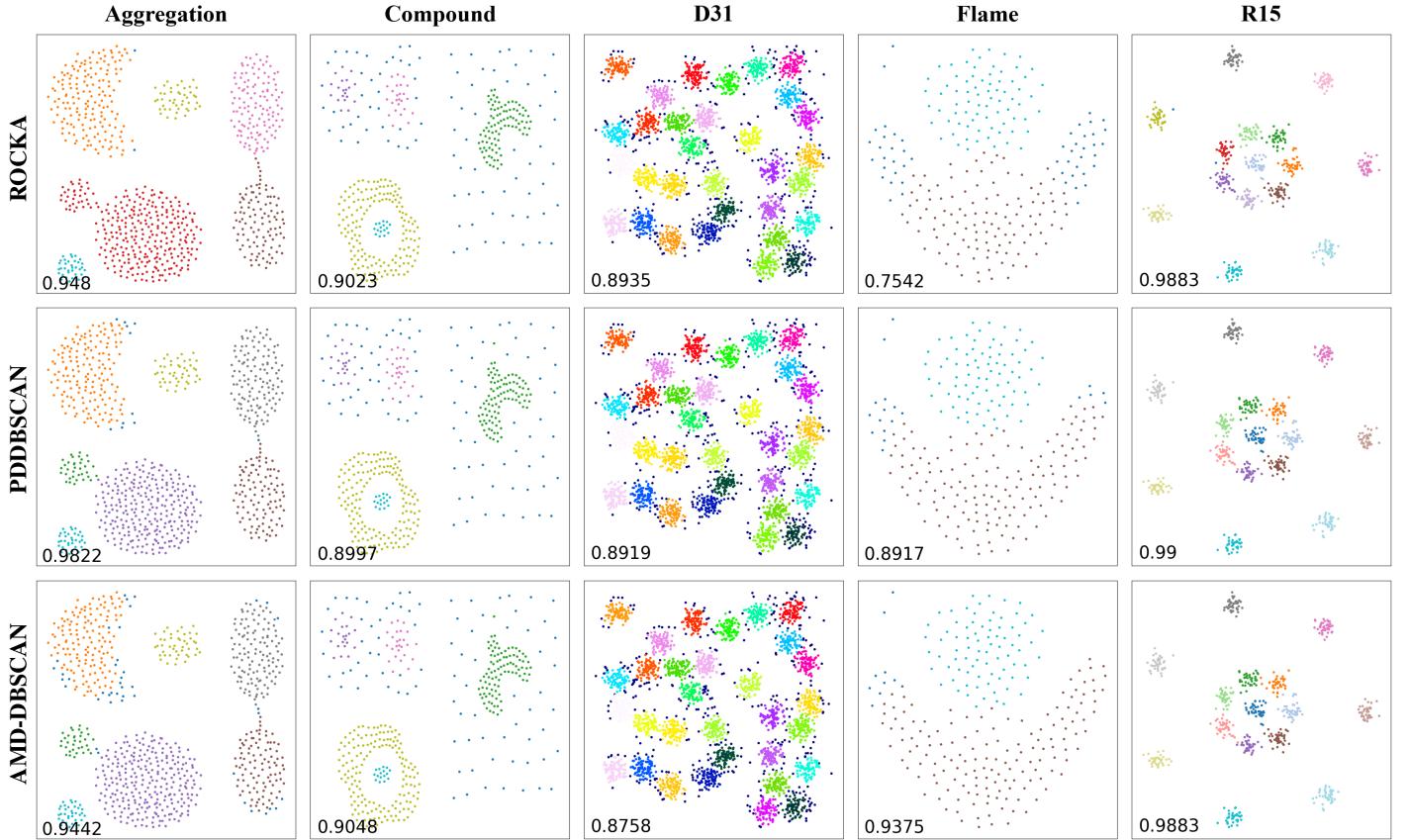


Fig. 3. Clustering results on Single-density datasets. The number in the lower left corner represents the accuracy of clustering.

(i.e., ROCKA [15] and PDDBSCAN [30]) are selected. The experimental results are illustrated in Table III, Figure 4 and the visualization effect is shown in Figure 3.

TABLE III  
COMPARISONS OF DIFFERENT ALGORITHM ON SINGLE-DENSITY DATASETS

	ROCKA			PDDBSCAN			AMD-DBSCAN		
dataset	accuracy	NMI	t(s)	accuracy	NMI	t(s)	accuracy	NMI	t(s)
Aggregation	0.948	0.94	<b>0.01</b>	<b>0.982</b>	<b>0.97</b>	0.10	0.944	0.93	0.14
Compound	0.902	<b>0.88</b>	<b>0.01</b>	0.900	0.87	0.02	<b>0.905</b>	0.874	0.05
D31	<b>0.894</b>	<b>0.88</b>	<b>0.14</b>	0.892	0.88	1.13	0.876	0.87	1.24
Flame	0.754	0.57	<b>0.01</b>	0.892	0.66	0.02	<b>0.938</b>	<b>0.75</b>	0.04
R15	0.988	0.99	<b>0.01</b>	<b>0.990</b>	<b>0.99</b>	0.06	0.988	0.98	0.09

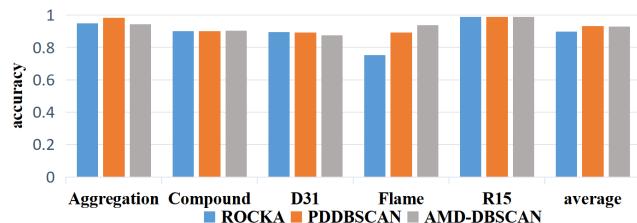


Fig. 4. Comparisons of accuracy on Single-Density datasets

As shown in Figure 4, compared to the other two algorithms,

AMD-DBSCAN has an average accuracy of 1.6% higher and achieves the best performance on the *Compound* and *Flame* datasets. Due to the process of parameter adaptation, our algorithm's execution time is longer than ROCKA [15] but still shorter than PDDBSCAN [30].

Figure 3 indicates that, for the *Aggregation* dataset, AMD-DBSCAN considers more points as noise compared to the other algorithms but distinguishes all clusters. However, ROCKA [15] fails to distinguish two adjacent clusters.

As for the *Compound* dataset, the *D31* dataset, and the *R15* dataset, the three algorithms have similar accuracy, which shows that they have good performance for datasets with lower VNN (i.e., datasets with uniform density distribution).

In the case of the *Flame* dataset, which is comprised of only two clusters, the accuracy of AMD-DBSCAN is much higher than that of the other two algorithms, especially 18.3% higher than that of ROCKA [15]. Because the points in this dataset are relatively scattered, many of them are considered as noise, which leads to low accuracy.

Therefore, AMD-DBSCAN has a good performance on Single-density datasets.

#### E. Multi-density Datasets

In order to evaluate the performance on extreme Multi-density datasets (i.e., datasets with VNN larger than 100) for our proposed Multi-density

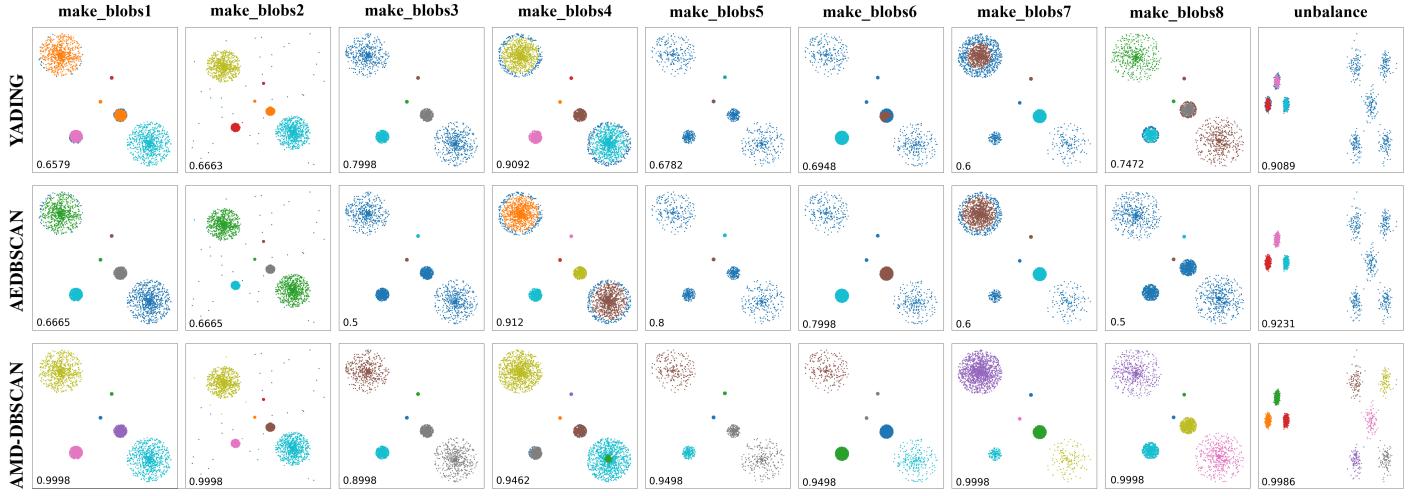


Fig. 5. Clustering results on Multi-density datasets. The number in the lower left corner represents the accuracy of clustering.

clustering approaches (i.e., YADING [12], AEDBSCAN [14]) are selected for comparison experiments. The experimental results are shown in Table IV, Figure 6 and the visualization effect is shown in Figure 5.

TABLE IV  
COMPARISONS OF DIFFERENT ALGORITHM ON MULTI-DENSITY DATASETS

dataset	YADING			AEDBSCAN			AMD-DBSCAN		
	accuracy	NMI	t(s)	accuracy	NMI	t(s)	accuracy	NMI	t(s)
make_blobs1	0.658	0.79	<b>1.69</b>	0.666	0.90	3.69	<b>1.00</b>	<b>1.00</b>	3.62
make_blobs2	0.666	0.84	<b>1.33</b>	0.666	0.80	4.17	<b>1.00</b>	<b>0.99</b>	3.77
make_blobs3	0.800	<b>0.96</b>	<b>1.34</b>	0.500	0.75	4.28	<b>0.90</b>	0.95	3.43
make_blobs4	0.909	0.92	<b>0.49</b>	0.912	0.91	4.14	<b>0.94</b>	<b>0.95</b>	3.06
make_blobs5	0.678	0.64	<b>2.52</b>	0.800	0.88	6.46	<b>0.95</b>	<b>0.93</b>	4.88
make_blobs6	0.695	0.71	<b>1.29</b>	0.800	0.88	5.22	<b>0.95</b>	<b>0.97</b>	12.52
make_blobs7	0.600	0.69	<b>1.66</b>	0.600	0.69	4.88	<b>1.00</b>	<b>1.00</b>	3.92
make_blobs8	0.747	0.78	<b>1.74</b>	0.500	0.75	4.602	<b>1.00</b>	<b>1.00</b>	3.64
unbalance	0.909	0.91	7.47	0.938	0.95	12.11	<b>0.99</b>	<b>0.99</b>	<b>7.23</b>

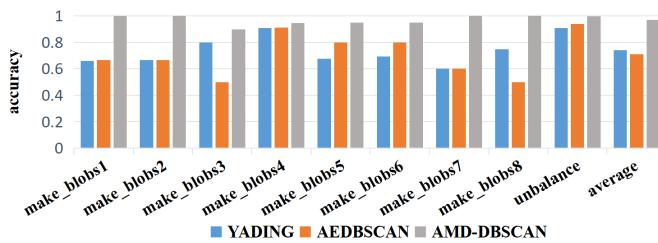


Fig. 6. Comparisons of accuracy on Multi-density datasets

As illustrated in Figure 6, AMD-DBSCAN has higher accuracy and NMI on Multi-density datasets of extremely variable density than the other two methods. Experimental results show that our AMD-DBSCAN has the best performance on the majority of datasets. Since having the parameter adaptation process, our algorithm's execution time is longer than YADING [12] but still shorter than AEDBSCAN [14].

As shown in Figure 5, different colors are represented in different clusters. For the *make\_blobs1* dataset, clusters with small radius have high density while clusters with large radius have low density. AMD-DBSCAN achieves the best clustering performance, while the other two algorithms consider the points at the edge of each cluster as noise. AEDBSCAN [14] even considers a sparse cluster as noise, resulting in low accuracy.

As for the *make\_blobs2* dataset, it is proven that our AMD-DBSCAN is highly resistant to noise by the evaluation of adding noise to the base dataset.

In terms of the *make\_blobs3* and *make\_blobs4* datasets, the number of points of different clusters is modified. AMD-DBSCAN has the highest accuracy. However, the other two algorithms are not able to distinguish those sparse clusters and the points at the edge of the clusters.

In the case of from the *make\_blobs5* to *make\_blobs8* datasets, the distribution between each cluster is changed and the whole dataset is more sparsely distributed. For these sparse points, the other two algorithms consider them as noise, while AMD-DBSCAN can distinguish these points from the denser ones and cluster them correctly.

The *unbalance* dataset is a Multi-density dataset in which the vast majority of points are in the three high-density clusters on the left, while the five on the right are low-density clusters. Since the number of these low-density clusters is small, these clusters are considered as noise. The experimental results show that the other two algorithms can only distinguish three high-density clusters. In contrast, AMD-DBSCAN can completely distinguish eight clusters with high accuracy.

To sum up, AMD-DBSCAN improves accuracy by an average of 24.7% over the other two algorithms on Multi-density datasets of extremely variable density, which proves that the candidate *Eps* obtained by our algorithm and their corresponding *MinPts* are adapted to the distribution of the dataset.

TABLE V  
ABLATION STUDY

AMD-DBSCAN				Test1( $k=4$ )			Test2( $k=n/2$ )			Test3(no K-means)	
dataset	accuracy	NMI	clusters	accuracy	NMI	clusters	accuracy	NMI	clusters	accuracy	NMI
Aggregation	0.944	0.931	7	0.675	0.728	24	0.346	0.000	1	<b>0.957</b>	<b>0.939</b>
Compound	<b>0.905</b>	<b>0.874</b>	5	0.782	0.831	5	0.396	0.082	1	0.789	0.867
D31	<b>0.876</b>	<b>0.870</b>	31	0.555	0.671	101	0.032	0.000	1	0.813	0.820
Flame	<b>0.938</b>	0.748	2	0.925	<b>0.808</b>	2	0.638	0.000	1	0.925	0.718
R15	<b>0.988</b>	<b>0.985</b>	15	0.620	0.660	26	0.105	0.104	2	0.983	0.980
make_blobs1	<b>1.000</b>	<b>1.000</b>	6	0.875	0.878	35	0.325	0.411	3	0.825	0.807
make_blobs2	<b>1.000</b>	<b>0.992</b>	6	0.904	0.919	10	0.165	0.114	3	0.851	0.834
make_blobs3	0.900	0.949	5	0.834	0.871	24	<b>0.580</b>	0.690	4	<b>0.966</b>	<b>0.961</b>
make_blobs4	<b>0.946</b>	<b>0.955</b>	7	0.618	0.789	51	0.250	0.000	1	0.834	0.916
make_blobs5	<b>0.950</b>	<b>0.973</b>	5	0.603	0.777	17	0.547	0.592	4	0.899	0.888
make_blobs6	<b>0.950</b>	<b>0.973</b>	5	0.718	0.772	28	0.450	0.259	2	0.678	0.830
make_blobs7	<b>1.000</b>	<b>1.000</b>	6	0.907	0.902	19	0.630	0.688	4	0.756	0.705
make_blobs8	<b>1.000</b>	<b>1.000</b>	6	0.840	0.898	23	0.400	0.503	2	0.965	0.958
unbalance	<b>0.999</b>	<b>0.998</b>	9	0.629	0.685	92	0.323	0.322	2	0.997	0.992

### F. Ablation Study

In order to verify the rationality of each step of AMD-DBSCAN, three sets of ablation study are designed.

First, Test1 verifies that the adaptive  $k$  derived from parameter adaptation is valid. In YADING [12] and ROCKA [15],  $k$  is taken as 4 by default. In Test1,  $k$  is taken to be 4, and then continue to complete our clustering process. The experimental results shown in Table V indicate that  $k$  obtained by the parameter adaptation is instructive for determining  $k_{dis}$  value because it can locate an adaptive  $k$  according to the distribution characteristics of the dataset. However, if  $k$  is constant by default 4, the accuracy of DBSCAN decreases by an average of 20.8%.

Test2 together with Test1 proves the theory presented in section 3.2, that is, a large  $k$  leads to a small number of clusters, while a small  $k$  results in a large number of clusters. In Test2,  $n$  represents the number of points in the dataset, and  $k$  is taken to be  $n/2$ , which is a very large value compared to that in Test1. The comparisons of the number of clusters from Test1 and Test2 show that when  $k$  is small, the number of clusters is large, and when  $k$  is large, the number of clusters is small, which proves that an appropriate  $k$  is important because it affects the clustering results significantly.

Test3 is to verify that the candidate  $Eps$  obtained by the K-means algorithm is valid. The K-means algorithm in AMD-DBSCAN is replaced by YADING's algorithm [12], and then continue to complete our clustering process. Test3 proves that the algorithm proposed in this study for processing the  $k_{dis}$  frequency histogram using the K-means algorithm is valid. This is because  $K$  can be easily determined by directly observing the  $k_{dis}$  frequency histogram. And K-means is effective at clustering one-dimensional data like  $k_{dis}$  values [31]. Experimental results show that the clustering accuracy of our algorithm is on average 8.3% better compared to YADING's algorithm [12] without using the K-means algorithm.

In summary, the ablation study shows that each step of AMD-DBSCAN is necessary because it guides the selection of better parameter pairs thus improving clustering effect.

### V. DISCUSSION

The current AMD-DBSCAN implementation is single-threaded. From the discussion of the AMD-DBSCAN algorithm, it is clear that the AMD-DBSCAN implementation can be easily parallelized because it can perform adaptive operations in multiple partitions at the same time. In specific, some partitioning strategies and distributed structures (e.g., MapReduce [24]) can be utilized in our model. Hence, our model can be applied to cluster in each small partition and finally merge the clustering results. In this way, our model can divide the large-scale dataset into different small partitions, use a parameter adaptation process to determine the parameter pairs on each small partition, and use parallelization to speed up this process. Therefore, our model can be applied to large-scale datasets. As a result, the performance of AMD-DBSCAN can be significantly improved by parallelization.

### VI. CONCLUSION

In this paper, an adaptive Multi-density DBSCAN algorithm (AMD-DBSCAN) with high robustness and efficiency is proposed. First, an improved parameter adaptation method is proposed to locate  $k$ . The binary search algorithm is utilized to speed up the adaptive process and the experimental results show that the speed of our algorithm is on average 4 times faster than the traditional methods. Second, instead of calculating the inflection points of the  $k_{dis}$  curves, multiple candidate  $Eps$  can be obtained by using the  $k_{dis}$  frequency histogram and the K-means algorithm. And compared to other approaches, AMD-DBSCAN requires only one hyperparameter that is easily determined. In addition, AMD-DBSCAN improves the algorithm of Multi-density clustering so that there is an adaptive parameter pair for each density cluster. Furthermore, the variance of the number of neighbors (VNN) is proposed to measure the difference in density. The experimental results show that the clustering accuracy of AMD-DBSCAN is 24.7% higher than other algorithms on average on Multi-density datasets of extremely variable density and is 1.6% higher on average on Single-density datasets.

## REFERENCES

- [1] 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.
- [2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, pp. 37–37, 1996.
- [3] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques," Morgan Kaufmann Publishers, San Francisco, CA, pp. 335–391, 2001.
- [4] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 657–681, 2016.
- [5] R. Arya and G. Sikka, "An optimized approach for density based spatial clustering application with noise," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I*. Springer, 2014, pp. 695–702.
- [6] H. Lang, Y. Xi, and X. Zhang, "Ship detection in high-resolution SAR images by clustering spatially enhanced pixel descriptor," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5407–5423, 2019.
- [7] C. H. Jin, H. J. Na, M. Piao, G. Pok, and K. H. Ryu, "A novel DBSCAN-based defect pattern detection and classification framework for wafer bin map," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 3, pp. 286–292, 2019.
- [8] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbcscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.
- [9] A. Bewley, R. Shekhar, S. Leonard, B. Upcroft, and P. Lever, "Real-time volume estimation of a dragline payload," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1571–1576.
- [10] Ö. PASİN and H. ANKARALI, "Usage of Kernel K Means and DBSCAN cluster algorithms in health studies An application," 2015.
- [11] E. Keogh and A. Mueen, "Curse of Dimensionality," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 314–315.
- [12] R. Ding, Q. Wang, Y. Dang, Q. Fu, H. Zhang, and D. Zhang, "Yading: Fast clustering of large-scale time series data," *Proceedings of the VLDB Endowment*, vol. 8, no. 5, pp. 473–484, 2015.
- [13] A. Bryant and K. Cios, "RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 6, pp. 1109–1121, 2017.
- [14] V. Mistry, U. Pandya, A. Rathwa, H. Kachroo, and A. Jivani, "AEDBSCAN—Adaptive Epsilon Density-Based Spatial Clustering of Applications with Noise," in *Progress in Advanced Computing and Intelligent Engineering*. Springer, 2021, pp. 213–226.
- [15] Z. Li, Y. Zhao, R. Liu, and D. Pei, "Robust and rapid clustering of kpis for large-scale anomaly detection," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–10.
- [16] Y. Lv, T. Ma, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "An efficient and scalable density-based clustering algorithm for datasets with complex structures," *Neurocomputing*, vol. 171, pp. 9–22, 2016.
- [17] J.-H. Kim, J.-H. Choi, K.-H. Yoo, and A. Nasridinov, "AA-DBSCAN: An approximate adaptive DBSCAN for finding clusters with varying densities," *The Journal of Supercomputing*, vol. 75, no. 1, pp. 142–169, 2019.
- [18] C. Xiaoyun, M. Yufang, Z. Yan, and W. Ping, "GMDBSCAN: Multi-density DBSCAN cluster based on grid," in *2008 IEEE International Conference on E-Business Engineering*. IEEE, 2008, pp. 780–783.
- [19] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: Varied density based spatial clustering of applications with noise," in *2007 International Conference on Service Systems and Service Management*. IEEE, 2007, pp. 1–4.
- [20] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, "A density based algorithm for discovering density varied clusters in large spatial databases," *International Journal of Computer Applications*, vol. 3, no. 6, 2010.
- [21] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2013, pp. 160–172.
- [22] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan, "Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce," in *2011 IEEE 17th International Conference on Parallel and Distributed Systems*. IEEE, 2011, pp. 473–480.
- [23] Y. Yu, J. Zhao, X. Wang, Q. Wang, and Y. Zhang, "Cludoop: An efficient distributed density-based clustering for big data using hadoop," *International Journal of Distributed Sensor Networks*, vol. 11, no. 6, p. 579391, 2015.
- [24] E. A. Mohammed, B. H. Far, and C. Naugler, "Applications of the MapReduce programming framework to clinical big data analysis: Current landscape and future trends," *BioData mining*, vol. 7, no. 1, pp. 1–23, 2014.
- [25] S. Zhou, A. Zhou, J. Cao, J. Wen, Y. Fan, and Y. Hu, "Combining sampling technique with DBSCAN algorithm for clustering large spatial databases," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2000, pp. 169–172.
- [26] B. Borah and D. K. Bhattacharyya, "An improved sampling-based DBSCAN for large spatial databases," in *International Conference on Intelligent Sensing and Information Processing, 2004. Proceedings Of*. IEEE, 2004, pp. 92–96.
- [27] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of statistical mechanics: Theory and experiment*, vol. 2005, no. 09, p. P09008, 2005.
- [28] P. Fräntti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, no. 12, pp. 4743–4759, Dec. 2018.
- [29] "Scikit-learn: Machine learning in Python — scikit-learn 1.0.2 documentation," <https://scikit-learn.org/stable/>.
- [30] X. Lu, Y. Wang, J. Yuan, X. Wang, K. Fu, and K. Yang, "A Parallel Adaptive DBSCAN Algorithm Based on k-Dimensional Tree Partition," in *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDI)*. IEEE, 2020, pp. 249–256.
- [31] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Icml*, vol. 1, 2001, pp. 577–584.