AMD-DBSCAN: An Adaptive Multi-density DBSCAN

Ziqing Wang
Zhirong Ye
School of Microelectronics Science and Technology, Sun Yat-Sen University
Zhuhai, China
wangzq37@mail2.sysu.edu.cn

ABSTRACT

DBSCAN is a powerful tool in density-based clustering algorithms, but it requires two hard-to-determine hyperparameters (i.e. *Eps* and *MinPts*).To initialize these two hyperparameters, many configurations have to be conducted to achieve good clustering results, which is very time-consuming. Moreover, traditional DBSCAN can not perform Multi-density clustering. In this paper, an adaptive Multi-density DBSCAN algorithm(AMD-DBSCAN) is being proposed to solve these problems. The experiments results show that our approach speeds up the execution time of parameter adaptive procedures. Furthermore, our method relies on only one hyperparameter that is easy to determine and insensitive to the results, avoiding complicated repetitive initialization operations and allowing it to be applied to large-scale datasets where the hyperparameters are difficult to determine. In addition, our proposal is much better than existing algorithms in Multi-density datasets and also maintains its good performance in single-density scenarios.

Keywords-DBSCAN, Parameter Adaption, Multi-denstiy

1 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 more accurately in the presence of noise[5]. It is widely used in various fields such as ship detection[6], wafer classification[7], astronomy[8], robotics[9]and disease diagnosis[10]. DBSCAN clustering relies on two hyperparameters - *Eps* and *MinPts*. For a data point, *Eps* is the radius of the circle whose center is that point. All the points contained in that circle could be considered as neighbors of that point. *MinPts* is a threshold value that can be utilized to find core points during the process of DBSCAN clustering. Specifically, a data point could be marked as a core point if the number of its neighbors is more than *MinPts*. These two parameters guide the clustering process in DBSCAN and their values significantly affect the accuracy of the clustering. However, in most cases, it is very difficult to determine these two parameters because once the dataset has a dimensionality greater than three dimensions, the visualization of the data is difficult, 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], making the clustering results unsatisfactory.

Therefore, some approaches have been proposed to adapt these two parameters. However, some of them could only adapt one of the two parameters, e.g. YADING[12], which only adapts *Eps*. There are also algorithms that can adapt both parameters(i.e. *Eps* and *MinPts*) at the same time, such as PDDBSCAN[13]. But it is time-consuming because the algorithm requires to traverse all candidate combinations of parameters(i.e. *Eps* and *MinPts*). To address the drawback, inspired by PDDBSCAN, a relatively efficient self-adaptive strategy that can evaluate *Eps* and *MinPts* simultaneously is being proposed in this paper. Besides, by virtual of the binary search approach, the execution time of it is significantly reduced, compared with PDDBSCAN.

On top of that, the distribution of points in the same dataset may not be balanced. To be more specific, the density of each cluster is different. For datasets with uneven densities, it is unreasonable to use only a fixed pair of parameters for clustering. That is because giving parameters at the density of the sparse clusters would lead to the merging of similar datasets. In contrast, giving parameters at the density of the dense clusters would result in many clusters with less density being incorrectly identified as noise. Therefore, Multi-density clustering was proposed to deal with this situation. Specifically, the densest data points are clustered first, then the data points clustered are stripped, and the densest points among the remaining points are continued to be clustered until all points are clustered. This means that different combinations of parameters are used for different densities. But the parameters of each layer are difficult to determine at the same time. Therefore, an adaptive parameter approach fused with Multi-density clustering is being proposed in this paper, which can provide two parameters for each layer of data simultaneously.

In this paper, our main contribution could be summarized as 3 key points:

(1) An adaptive parameter method is being proposed to configure DBSCAN parameters (i.e. *Eps* and *MinPts*), at the same time and the experimental results show that the accuracy of clustering results of our method is high.
(2) The binary search algorithm is introduced in this paper, which accelerates parameter adaptation and reduces the computational complexity of the PDDBSCAN[13].
(3) AMD-DBSCAN integrates the Multi-density clustering method and provides *Eps* and *MinPts* parameters for clusters of different densities, which can be applied to Multi-density datasets.

2 RELATED WORK

The clustering results of DBSCAN are significantly affected by both hyperparameters of the algorithm and the distribution of datasets. As for hyperparameters, 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 dataset s' 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.

2.1 Configuration of Hyperparameters

To find appropriate hyperparameters for clustering of DBSCAN, many approaches have been proposed. These methods can be divided into two categories, i.e. adapting one of the two parameters and configuring both of them simultaneously.

2.1.1 ***Single Parameter.*** Reversing the nearest neighbor has been proposed by [14] 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, 15, 16] to adapt appropriate *Eps* by locating the flat part of the curve. However, the analysis of the k_{dis} curve proposed by [12, 16] requires another two hyperparameters and is with a relatively large complexity ($O(nlog(n))$). The OPTICS[17] algorithm is an improvement for the input parameter *Eps*. Although it also requires two input parameters, OPTICS is insensitive to the *Eps* and generally fixes *Eps* to infinity. AA-DBSCAN[18] use the approximate adaptive *Eps* for each density so that it can find the clusters in the Multi-density datasets.Overall, 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 hyperparameters at the same time.

2.1.2 ***Multiple Parameters.*** The following method is proposed to provide two hyperparameters. In GMDBSCAN [19], centers of clusters are generated by GD to adapt both two parameters. However, it does not perform as satisfactorily as AMD-DBSCAN when dealing with the Multi-density datasets.

2.2 Properties of Datasets

2.2.1 ***Multi-density Datasets.*** Many algorithms are being proposed for tackling the situation of Multi-density datasets with unevenly distributed points. For instance, VDBSCAN[20] proposed to automatically customize clustering parameters for different density regions. Furthermore, AEDBSCAN[15] improves the k_{dis} plotting process 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[20], while optimizing the clustering speed. DVBSCAN [21] can handle local density variation within a cluster, but it can not determine parameters automatically. HDBSCAN[22] is a hierarchical clustering method that allows it to perform well on multi-density datasets and it is parameter insensitive but has a long runtime.

Since AMD-DBSCAN can adapt a pair of parameters at each layer during the Multi-density DBSCAN process, our algorithm achieves good clustering results on Multi-density datasets.

2.2.2 ***Large-scale Datasets.*** Some algorithms are proposed to be applied under large-scale datasets. Some methods [23–25] use a partitioning strategy and a distributed structure that allow them to be applied to large-scale datasets. SDBSCAN algorithm [26] combines sampling techniques with DBSCAN for clustering large spatial databases. In IDBSCAN[27], 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 cannot handle Multi-density datasets.

AMD-DBSCAN is adaptive to parameters that are difficult to determine in large-scale datasets, so it can have good performance.

3 PROPOSED ALGORITHM

3.1 Overview

AMD-DBSCAN is divided into three steps: parameter adaptation, getting candidate *Eps* list, and clustering. Figure 1 gives the information of the framework proposed in this paper.

(1) **Parameter Adaptation:** 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. After that, the *Eps* and *MinPts* parameter lists are sequentially input into the DBSCAN to obtain the number of clusters. And then, by the binary search algorithm, the most suitable set of *Eps* and *MinPts* pairs for clustering is selected, where *MinPts* is assigned to the K needed for the next step.
(2) **To get Candidate Eps List:** The k_{dis} curve is drawn by using the K values obtained from the last step, where k_{dis} value is defined as the distance between a point and the K^{th} neighbor of that point, and the k_{dis} curve is a series of k_{dis} values arranged in ascending order. After that, the $K-means$ algorithm is utilized to cluster the values of k_{dis} curves to obtain N candidate *Eps*, where N refers to the number of flat parts by directly observing the k_{dis} curve.
(3) **Clustering:** The density-based clustering algorithm essentially discovers a high-density data set in a dataset, that is, a data set in which the average distance between data points is small. In our paper, there are no assumptions about the distribution of the dataset, which means that the input dataset may have arbitrary shapes and different densities. When dealing with some datasets with uneven densities, since the traditional DBSCAN has only one pair of parameters, it cannot handle such datasets. Therefore, inspired by YADING[12], Multi-density DBSCAN, well suited to handle datasets with such characteristics, is introduced in this paper, which utilizes the obtained *CandidateEpsList* in ascending order to Multi-density cluster the dataset sequentially and obtain the final clustering results.

3.2 Parameter Adaptation

There are many algorithms ([12, 16], including our algorithm) that utilize k_{dis} curves to obtain the important parameter *Eps* for clustering, where the plotting of k_{dis} curves requires manual input of the parameter K, which represents the K^{th} nearest neighbor of each point. The effect of clustering depends directly on the choice of the parameter K.

According to our experiments, shown in the latter section, a large value of K reduces the number of core points while a small value of K leads to a large number of points being marked as core

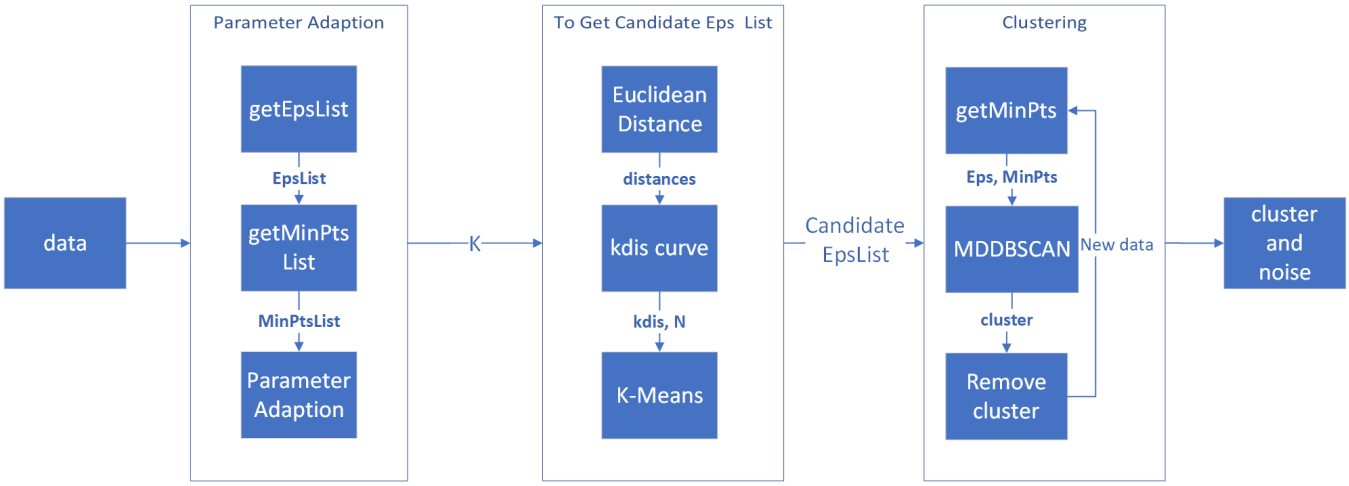

Figure 1: Framework of AMD-DBSCAN

points. Specifically, the decrease in the number of core points leads to a large number of clusters (what should be one large cluster is clustered into many small clusters), and the increase in the number of core points leads to a small number of clusters (what should be multiple clusters are clustered into one cluster). Therefore, to avoid the randomness of manually setting K during clustering, a parameter adaptation algorithm is being proposed in this paper. The specific process of parameter adaptation is as follows.

3.2.1 ***To compute Eps List.*** First of all, the distance distribution matrix $\mathrm{DIST}_{n\times n}$ of dataset D is calculated :

$$DIST_{n\times n} = \{dist(i,j), 1 \le i \le n, 1 \le j \le n\} \tag{1}$$

where $n = |D|$ represents the number of points in the data set 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 data set D.

Then, 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 $\overline{D_k}$ is obtained by averaging the data in the vector D_k, which yields a list of Eps as shown in Equation 2.

$$EpsList = \left\{\overline{D_k}, 1 \le k \le n\right\} \tag{2}$$

The whole process is encapsulated as a function $getEpsList(data)$. The input is data set D and the output is $EpsList$.

3.2.2 ***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.

$$MinPtsList = \left\{MinPts_j, 1 \le j \le n\right\} \tag{3}$$

$$MinPts_j = \frac{1}{n}\sum_{i=1}^{n} Point_i \tag{4}$$

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.

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

3.2.3 ***To get K.*** The $EpsList$ and $MinPtsList$ are chosen as the Eps candidates as well as $MinPts$ parameters, respectively. And then each pair of parameters is input into DBSCAN for clustering in turn 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 as the optimal number of clusters. And the best index is the maximum index when the number of clusters is equal to n.

Since the number of clusters decreases monotonically with *index* and to avoid traversing all combinations of parameters and reduce the time significantly, the **binary search algorithm** is utilized to find the best index. Finally, the $MinPts$ corresponding to the best index in the $MinPtsList$ is our adaptive K.

The algorithm is based on two findings that are concluded through our experiments.

(1) Since $EpsList$ is in ascending order, as the *index* increases, Eps increases. An experiment is designed in which each pair of Eps and $MinPts$ parameters input sequentially into DBSCAN to obtain the number of clusters and observe the change in the number of clusters with index. Based on our experiment, the number of clusters decreases monotonically with increasing *index*, as shown in Figure 2. As Eps increases, more points in the dataset are clustered into one class, so it 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, the point with the largest index in this stable part of the curve corresponds to the best clustering. Normalized Mutual Information (NMI)[28]

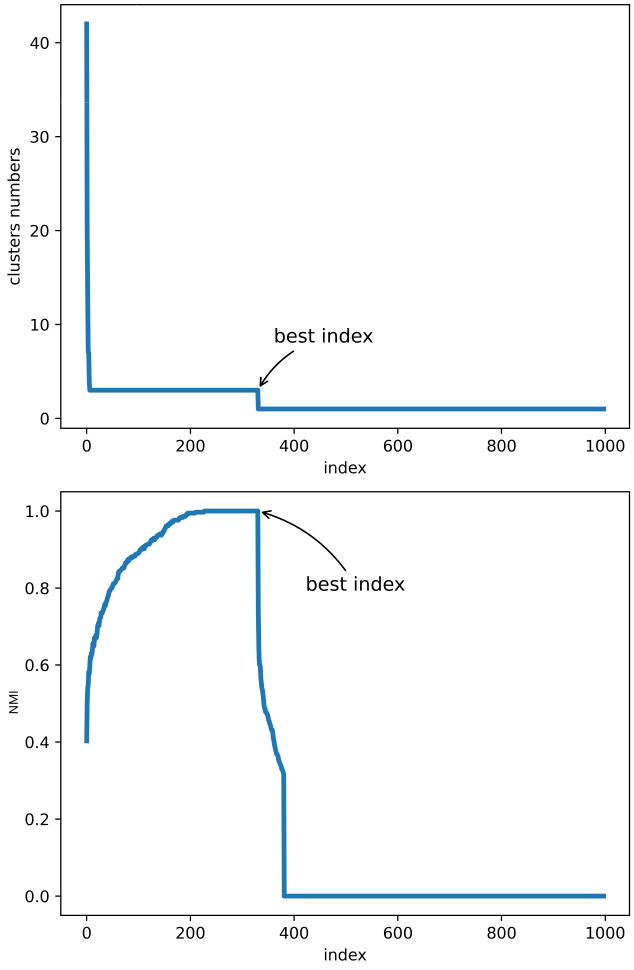

Figure 2: Variation of the clusters numbers and NMI score

is utilized to judge the clustering effect. An experiment is designed whereby the NMI of each DBSCAN result is recorded and the change in NMI with index is observed. As shown in Figure 2, our experiment shows that when the number of clusters is stable, the NMI increases until the point with the largest index is reached, which is called the best index.

The pseudo-code for getting K is shown in Algorithm 1.

3.3 To get Candidate Eps List

Because for Multi-density DBSCAN, multiple *Eps* are needed for clustering, so this part is to get the candidate *Eps* list. In the parameter adaptation above, K is obtained based on the distribution properties of the *data*, which is utilized to draw the k_{dis} curve.

As illustrated in Figure 3, 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. Inspired by YADING[12], the k_{dis} value of the flat part of the curve can be considered as a candidate *Eps* because it ensures that there are

Algorithm 1: Parameter Adaption

input : $data$
output: K

$Eps \leftarrow getEpsList(data)$
$MinPts \leftarrow getMinPtsList(data, Eps)$
$counter \leftarrow 0$
for $i \leq len(data) - 1$ **do**
  $ClusterNum[i] \leftarrow$ DBSCAN$(Eps[i], MinPts[i])$
  $ClusterNum[i+1] \leftarrow$ DBSCAN$(Eps[i+1], MinPts[i+1])$
  if $ClusterNum[i] == ClusterNum[i+1]$ **then**
    $counter \leftarrow counter + 1$
  end
  else
    $counter \leftarrow 0$
  end
  if $counter > 3$ **then**
    $n \leftarrow ClusterNum[i]$
    $left \leftarrow i$
    $right \leftarrow len(data) - 1$
    while $start \leq end$ **do**
      $mid \leftarrow (start + end)/2$
      if $DBSCAN(Eps[mid], MinPts[mid]) < n$ **then**
        $right \leftarrow mid$
      end
      else if $DBSCAN(Eps[mid], MinPts[mid]) > n$ **then**
        $Left \leftarrow mid$
      end
      else
        best index $\leftarrow mid$
        return $K \leftarrow MinPts$[bext index]
      end
    end
  end
end

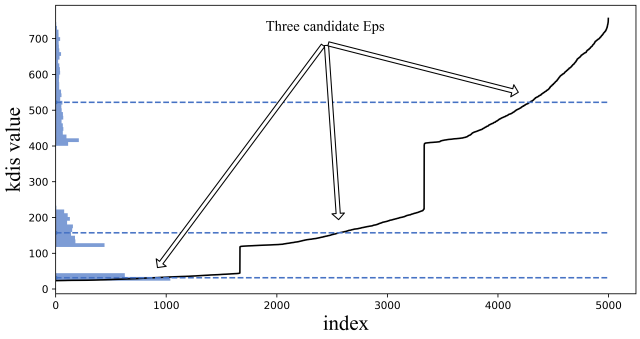

Figure 3: k_{dis} Curve

a large number of points in this range that can be clustered into one class. But unlike YADING's algorithm, which considers the inflection point is the flat part, our algorithm considers the center of the k_{dis} curve after doing $K - means$ clustering. This process will allow more similar k_{dis} values to be clustered in one class, so it is more representative of the *Eps* of this part of the points.

The $K-means$ algorithm is being proposed to find the flat parts on the k_{dis} curve and cluster those with similar k_{dis} values into one class. To reach this goal, the k_{dis} curve is drawn by using the K obtained in parameter adaptation. The $K-means$ algorithm is utilized to cluster the k_{dis} curves to obtain N candidate Eps, where N is the number of $K-means$ clustering centers determined by directly observing the number of flat parts on the k_{dis} curve. For example, as shown in the Figure 3, it can be observed that there are three flat parts, which correspond to three candidate Eps. And as shown in Figure 3 vertical axis, most of the similar k_{dis} values are clustered in the center after $K-means$ clustering. Therefore, as shown by the blue dashed line in the Figure 3, the k_{dis} value corresponding to the center of the clustering result is considered as the candidate Eps. After the above process, N candidate Eps will be obtained.

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

Algorithm 2: To Get Candidate Eps List

input : $data$
$K \leftarrow ParameterAdaption(data)$
output : $CandidateEpsList$: candidate Eps list for Multi-density DBSCAN

$distances \leftarrow sort(euclidean\_distances(data))$
$k_{dis} \leftarrow distances[K]$
$N \leftarrow$ Number of flat parts of k_{dis} curve
$CandidateEpsList \leftarrow \text{KMeans}(k_{dis}, N).cluster\_centers$
return $CandidateEpsList$

3.4 Clustering

Once obtaining the candidate Eps list, the next step is to perform Multi-density DBSCAN on the $data$. Unlike the algorithm of YADING[12], which sets all $MinPts$ to K, our algorithm calculates $MinPts$ based on the distribution properties of the $data$. $MinPts$ is calculated by the same algorithm as parameter adaptation, using the Equation 4,which utilizes a candidate Eps and $data$ to obtain $MinPts$. This process is encapsulated as the function $getMinPts(data, Eps)$.

So the whole process is that the first step is to sort the obtained candidate Eps list in ascending order, and then the getMinPts function is utilized to get the adaptive $MinPts$. Then, $data$, Eps, $MinPts$ input into DBSCAN for clustering. After that, the $data$ that has been clustered is discarded and repeats the above steps. Until all the Eps input into the model, the remaining points are the noise points. The whole clustering is finished.

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

3.5 Algorithm Complexity Analysis

Generally speaking, for a data set containing n points, the complexity of DBSCAN is $O(n^2)$.If adopting the divideand-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 $O(\log(n))$, so the complexity of the whole parameter

Algorithm 3: Multi-density DBSCAN

input : $data$
$EpsList \leftarrow getEpsCandidateList(data, K)$
output : $labels$: $labels$ are clustering results

for Eps *in* $EpsList$ **do**
  $MinPts \leftarrow getMinPts(data, Eps)$
  $cluster \leftarrow \text{DBSCAN}(data, Eps, MinPts)$
  Mark $data$ with $labels$
  Remove $cluster$ from $data$
end
Mark $data$ as $noise$
return $cluster$ and $noise$

adaptation is $O(\log(n)f(n))$. In the process of getting 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 $O(\log(n)f(n)) + O(n) + O(f(n)) = O(\log(n)f(n))$

4 EXPERIMENT

4.1 Experiment platform and dataset

The experimental platform in this paper is a personal computer with a processor AMD Ryzen 7 5800H and 16GB of RAM.

Several representative datasets are selected for our experiments from [29]. The make_blobs1 to make_blobs8 datasets are generated by scikit-learn[30]. And the nine datasets from make_blobs1 to make_blobs8, and unbalance are all Multi-density datasets, while the others are single-density datasets.

To better test the performance of our algorithm with Multi-density datasets, some artificial datasets were generated. make_blob1 is a dataset with a large difference in density of each cluster, so it serves as a base dataset. To test the robustness of the algorithm, some changes were made to the base dataset. In specific, the make_blob2 dataset is added with noise. The make_blob3 and make_blob4 datasets are modified the density of the dataset. And the make_blob5 and, make_blob6, make_blob7, and make_blob8 datasets are changed the distance between each cluster. Finally, eight different make_blob datasets were generated.

The details of the dataset are shown in Table 1.

4.2 Evaluation Metrics

There are two metrics utilized to evaluate the clustering results.

(1) A well-known evaluation metric, Normalized Mutual Information (NMI)[28] is utilized to judge the clustering results. Specifically, Mutual Information (MI) is a useful information metric in information theory, which can be viewed as the amount of information contained in one random variable about another random variable, or the reduction of uncertainty in one random variable due to the knowledge of another random variable.NMI is the scaling of MI between [0,1], which is utilized to measure the similarity of clustering results. The greater the value, the better the clustering results.

Table 1: Dataset Properties

data name	size	clusters	multi-Density
Aggregation	788	7	FALSE
Compound	399	5	FALSE
D31	3100	31	FALSE
Flame	240	2	FALSE
R15	600	15	FALSE
make_blobs1	4998	6	TRUE
make_blobs2	5048	6	TRUE
make_blobs3	4998	6	TRUE
make_blobs4	4998	6	TRUE
make_blobs5	4998	6	TRUE
make_blobs6	4998	6	TRUE
make_blobs7	4998	6	TRUE
make_blobs8	4998	6	TRUE
unbalance	6500	7	TRUE

(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.

4.3 Parameter Adaption Experiment

The experiment is designed to compare the execution time of our parameter adaptive approach with the traditional parameter adaptive approach PDDBSCAN[13] for different data sets. 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 2.

Table 2: Comparison of Execution Time

	our work		PDDBSCAN	
data name	t_min	t_average	t_min	t_average
Aggregation	0.165	0.191	0.275	0.291
Compound	0.031	0.041	0.049	0.052
D31	1.35	2.33	2.55	2.82
Flame	0.021	0.024	0.042	0.045
R15	0.076	0.087	0.225	0.234
unbalance	6.407	6.552	95.658	97.782

As can be seen from Table 2, after adopting the binary search algorithm, the execution time of our algorithm is much shorter than that of PDDBSCAN[13], which does not utilize the binary search algorithm. In particular, the speedup of AMD-DBSCAN is more effective in some datasets where the flat area of the number of clusters is particularly long. 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 15 times. So AMD-DBSCAN is well suited for this kind of Multi-density dataset with a large amount of data.

4.4 Single-Density DBSCAN Experiment

The experiment is designed to compare the single-density clustering approaches (i.e. ROCKA[16] and PDDBSCAN[13]) and our method on single-density datasets. The clustering results are illustrated in Table 3 and Figure 4.

As shown in Table 3, although all the algorithms achieve similar performances, our algorithm AMD-DBSCAN achieves the best performance on the Compound and Flame data sets. Due to the process of parameter adaption, our algorithm's execution is longer than ROCKA but still more efficient than PDDBSCAN.

Figure 4 indicates that, for the Aggregation dataset, AMD-DBSCAN considers more points as noise compared to the other algorithms but distinguishes all clusters. ROCKA fails to distinguish two clusters close to each other, while PDDBSCAN clustering works best.

As for the Compound dataset, the clustering results of the three algorithms are similar, because there are a lot of noise points in this dataset.

In terms of the D31 dataset, which consists of 31 clusters as well as many distantly distributed noise points, our algorithm performs slightly worse than the other two algorithms.

When it comes to the Flame dataset, which is comprised of only two clusters, AMD-DBSCAN performs far better than the other two algorithms, especially ROKCA. Since the points in this dataset are relatively scattered, many of them are considered as noise, leading to low accuracy.

In the case of the R15 dataset, which consists of 15 clusters, the clustering results of all three algorithms are good.

Overall, although AMD-DBSCAN is designed for Multi-density clustering, it still performs well on single-density datasets.

4.5 Multi-density DBSCAN Experiment

The experiment is designed to compare the Multi-density clustering approaches (i.e.YADING[12], AEDBSCAN[31]) and our method on Multi-density datasets. The clustering results are shown in Table 4 and Figure 5 .

As illustrated in Table 4, our algorithm AMD-DBSCAN performs well on Multi-density datasets. It shows the best performance on the majority of datasets and is much better than the other two algorithms at accuracy and NMI. But since having the parameter adaption process, our algorithm's execution is more time-consuming than YADING but still faster than AEDBSCAN.

As shown in Figure 4, different colors are represented in varied clusters. Noticed that dark blue points consider noise. For the make_blobs1 dataset, the density of clusters with a small radius is high, while the density of clusters with a large radius is low. AMD-DBSCAN achieves the best clustering performance. The other two algorithms consider the points at the edge of each cluster as noise, and AEDBSCAN even considers a sparse cluster as noise, resulting in low accuracy.

As for the make_blobs2 dataset, the addition of noise to the base dataset of make_blobs2 did not affect the clustering effect of our algorithm, proving that our algorithm is highly resistant to noise.

In terms of the make_blob3 and make_blob4 datasets, the number of points of different clusters is modified. AMD-DBSCAN is still the best, but there is a cluster with a too-small radius that is considered

Table 3: Comparison of Different Algorithm in Single-Density datasets

	our work			ROCKA			PDDBSCAN		
data name	**accuracy**	**NMI**	**t_average(s)**	**accuracy**	**NMI**	**t_average(s)**	**accuracy**	**NMI**	**t_average(s)**
Aggregation	0.944	0.931	0.141	0.948	0.940	**0.013**	**0.982**	**0.974**	0.099
Compound	**0.905**	0.874	0.055	0.902	**0.879**	**0.008**	0.900	0.871	0.027
D31	0.876	0.870	1.237	**0.894**	**0.884**	**0.140**	0.892	0.883	1.137
Flame	**0.938**	**0.748**	0.043	0.754	0.573	**0.005**	0.892	0.667	0.022
R15	0.988	0.985	0.093	0.988	0.985	**0.010**	**0.990**	**0.986**	0.062

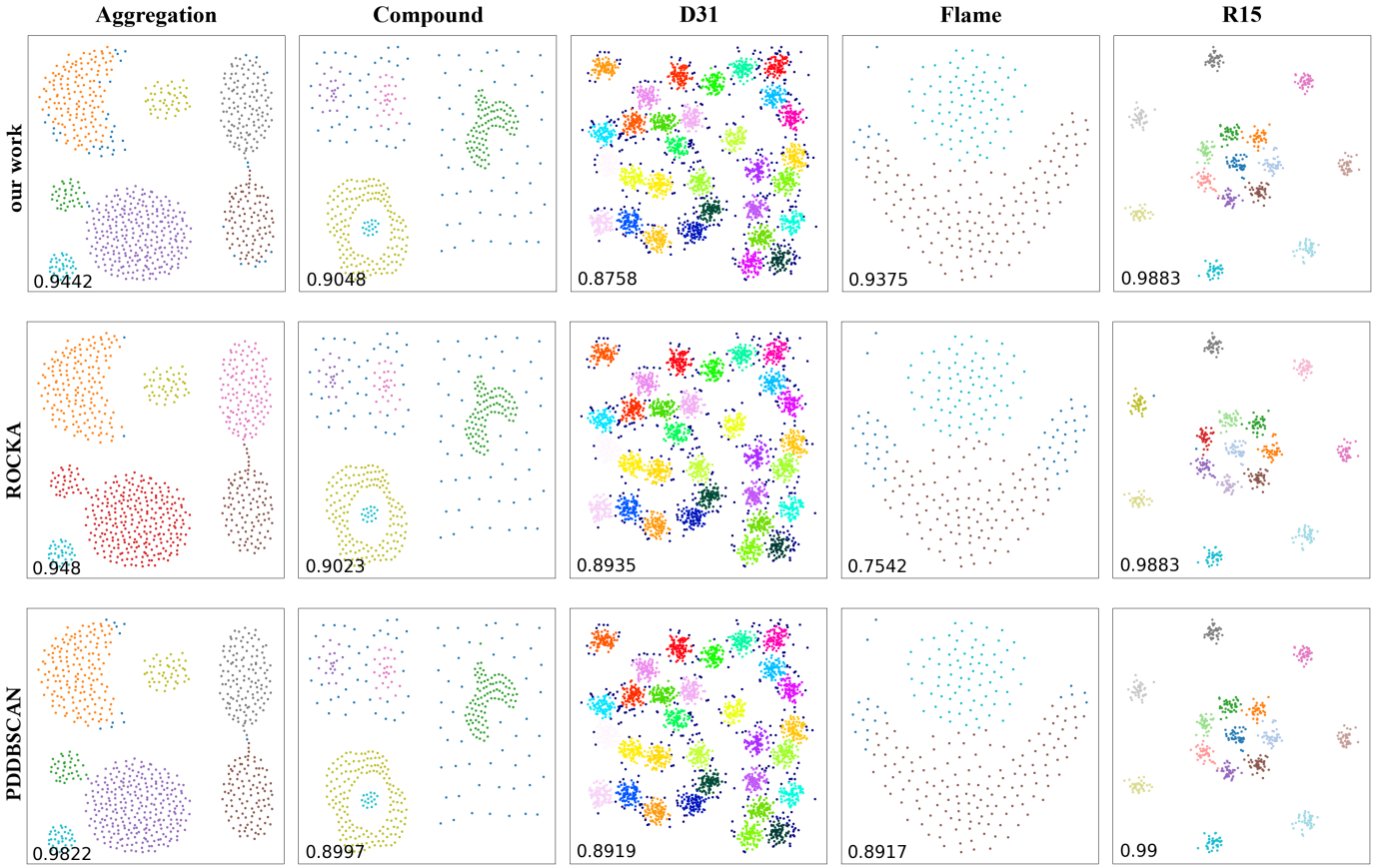

Figure 4: Clustering results in Single-Density datasets

as noise. And the other two algorithms still can't handle those sparse clusters and the points at the edge of the clusters.

When it comes to the make_blob5 and, make_blob6, make_blob7, and make_blob8 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 points, while AMD-DBSCAN can distinguish these points from the denser ones and cluster them correctly.

The unbalance dataset is a very classical 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, they cannot influence the selection of clustering parameters by the other two algorithms. Therefore, the other two algorithms can only distinguish three high-density clusters. In contrast, AMD-DBSCAN can completely distinguish eight clusters. Obviously, the candidate *Eps* obtained by our algorithm and their corresponding *MinPts* are very suitable.

4.6 Algorithm Validation

To verify the rationality of AMD-DBSCAN, three sets of comparison experiments are designed.

Table 4: Comparison of Different Algorithm in Multi-density datasets

	our work			YADING			AEDBSCAN		
data name	**accuracy**	**NMI**	**t_average(s)**	**accuracy**	**NMI**	**t_average(s)**	**accuracy**	**NMI**	**t_average(s)**
make_blobs1	**1.000**	**1.000**	3.625	0.658	0.787	**1.694**	0.666	0.907	3.693
make_blobs2	**1.000**	**0.992**	3.771	0.666	0.835	**1.335**	0.666	0.809	4.178
make_blobs3	**0.900**	0.949	3.434	0.800	**0.958**	**1.342**	0.500	0.753	4.281
make_blobs4	**0.946**	**0.956**	3.062	0.909	0.917	**0.491**	0.912	0.919	4.144
make_blobs5	**0.950**	**0.973**	4.882	0.678	0.638	**2.520**	0.800	0.884	6.468
make_blobs6	**0.950**	**0.973**	12.528	0.695	0.709	**1.298**	0.800	0.884	5.223
make_blobs7	**1.000**	**1.000**	3.929	0.600	0.685	**1.660**	0.600	0.692	4.885
make_blobs8	**1.000**	**1.000**	3.649	0.747	0.782	**1.749**	0.500	0.753	4.602
unbalance	**0.999**	**0.998**	**7.234**	0.909	0.907	7.474	0.938	0.954	12.114

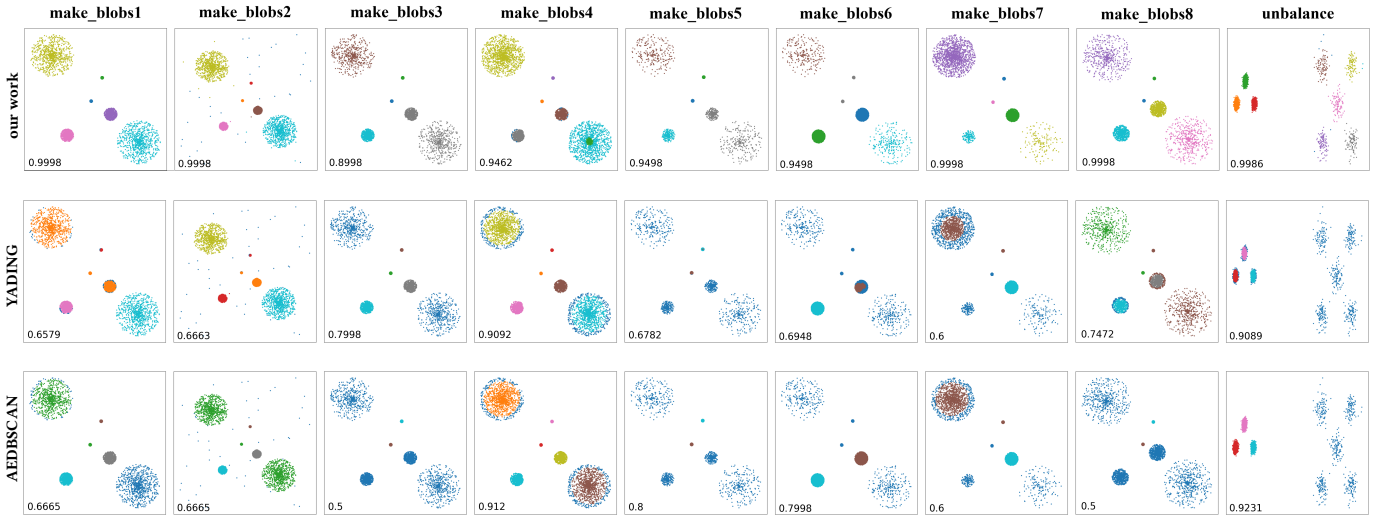

Figure 5: Clustering results in Multi-density datasets

First, test1 verifies that the K obtained by parameter adaptation is valid. In YADING and ROCKA, K is taken as 4 by default. Therefore, in the test1, K is taken to be 4, and then continue to complete our clustering process.

test2 together with test1 proves the theory presented in section 3.2, that is, a large value of K reduces the number of core points, leading to a small number of clusters. On the contrary, a small value of K reduces the number of core points, resulting in a large number of clusters.

test3 is to verify that the candidate Eps obtained by $K-means$ is valid. So replace $K-means$ in our algorithm with YADING's algorithm, and then continue to complete our clustering process.

The clustering results are shown in Table 5. From the clustering results, it can be found that K obtain by the parameter adaption is instructive for drawing k_{dis} curves because it can adapt a reasonable K according to the distribution characteristics of the data set. However, if K is constant by default 4, the clustering results are unsatisfied.

In the 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 the test1. The comparison of the number of clusters from test1 and test2 shows that when K is small, the number of clusters is large, and when K is large, the number of clusters is small, which proves our theory is correct.

The algorithm proposed in this paper for processing k_{dis} curves using k-means is effective. Compared with YADING's algorithm, the clustering results of our algorithm are much better.

5 DISCUSSION

To adapt the parameters of DBSCAN and to achieve Multi-density clustering, there are many aproaches[13, 16] that draw k_{dis} curves and use slope to find the inflection points to get the candidate Eps, and then utilize Eps to calculate $MinPts$ to get multiple parameter pairs. However, these methods tend to add new hard-to-determine hyperparameters and often result in many duplicate invalid parameter pairs. In contrast, our approach AMD-DBSCAN is a new

Table 5: Comparison of Different Algorithm

	our work			test1(k=4)			test2(k=n/2)			test3(no K-means)	
data name	**accuracy**	**NMI**	**clustes**	**accuracy**	**NMI**	**clusters**	**accuracy**	**NMI**	**clusters**	**accuracy**	**NMI**
Aggregation	0.944	0.931	7	0.675	0.728	24	0.346	0.000	1	**0.957**	**0.939**
Compound	**0.905**	**0.874**	5	0.782	0.831	5	0.396	0.082	1	0.789	0.867
D31	**0.876**	**0.870**	31	0.555	0.671	101	0.032	0.000	1	0.813	0.820
Flame	**0.938**	0.748	2	0.925	**0.808**	2	0.638	0.000	1	0.925	0.718
R15	**0.988**	**0.985**	15	0.620	0.660	26	0.105	0.104	2	0.983	0.980
make_blobs1	**1.000**	**1.000**	6	0.875	0.878	35	0.325	0.411	3	0.825	0.807
make_blobs2	**1.000**	**0.992**	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	0.580	0.690	4	**0.966**	**0.961**
make_blobs4	**0.946**	**0.955**	7	0.618	0.789	51	0.250	0.000	1	0.834	0.916
make_blobs5	**0.950**	**0.973**	5	0.603	0.777	17	0.547	0.592	4	0.899	0.888
make_blobs6	**0.950**	**0.973**	5	0.718	0.772	28	0.450	0.259	2	0.678	0.830
make_blobs7	**1.000**	**1.000**	6	0.907	0.902	19	0.630	0.688	4	0.756	0.705
make_blobs8	**1.000**	**1.000**	6	0.840	0.898	23	0.400	0.503	2	0.965	0.958
unbalance	**0.999**	**0.998**	9	0.629	0.685	92	0.323	0.322	2	0.997	0.992

exploration of the utilization of k_{dis} curves and minimizes hyperparameters.

For the utilization of k_{dis} curve, AMD-DBSCAN no longer sticks to finding the inflection points, but directly observes the distribution properties of k_{dis} values as shown in Figure 3 and clusters it using $K-means$ to obtain candidate Eps. Secondly, to reduce hyperparameters, our algorithm also utilizes parameter adaptation to obtain K and $MinPts$. Compared to other algorithms that analyze the k_{dis} curve, our algorithm obtains parameter pairs more simply and requires fewer hyperparameters that are easier to determine. In detail, the only hyperparameter in our approach can be determined by directly observing the k_{dis} curve. Moreover, a slight change in this hyperparameter does not greatly affect the final results, so our algorithm is also robust. And our algorithm outperforms all these algorithms for Multi-density clustering and is comparable to related algorithms for single-density clustering.

But limited by the method of Multi-density clustering, AMD-DBSCAN does not make good use of the multiple sets of parameter pairs obtained by parameter adaptation, and there is room for improvement in the accuracy of clustering. In the experiments, it is found that the clustering results are not good for the cluster with a small density but a large number, resulting in a circular shape of clustering results, for example, the make_blob5 dataset. In specific, when the parameter pairs with smaller Eps are clustered, the low-density points that account for most of the dataset are clustered first, while the parameter pairs with larger Eps are clustered later, eventually leading to a very clear layering. This defect is caused by the nature of the Multi-density clustering algorithm.

Due to the limitations mentioned above, a better Multi-density clustering algorithm can be searched for in subsequent studies to improve the accuracy of clustering. Since our algorithm requires only one hyperparameter and is easy to determine, and the accuracy of clustering is high, our algorithm can be used for large data clustering where it is difficult to determine hyperparameters.

6 CONCLUSION

In this paper, an adaptive Multi-density DBSCAN algorithm with high robustness and efficiency is being proposed. Compared to other approaches, AMD-DBSCAN requires only one hyperparameter that is easily determined and insensitive to the clustering results. Moreover, AMD-DBSCAN accelerates the parameter adaption process, making our parameter adaptive execution time much faster. And AMD-DBSCAN is much better than other algorithms on Multi-density datasets and has comparable performance on single-density datasets.

REFERENCES

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[2] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996.

[3] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques, Morgan Kaufmann Publishers. *San Francisco, CA*, pages 335–391, 2001.

[4] Craig Cooper, Daniel Franklin, Montserrat Ros, Farzad Safaei, and Mehran Abolhasan. A comparative survey of VANET clustering techniques. *IEEE Communications Surveys & Tutorials*, 19(1):657–681, 2016.

[5] Rakshit Arya and Geeta 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*, pages 695–702. Springer, 2014.

[6] Haitao Lang, Yuyang Xi, and Xi Zhang. Ship detection in high-resolution SAR images by clustering spatially enhanced pixel descriptor. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5407–5423, 2019.

[7] Cheng Hao Jin, Hyuk Jun Na, Minghao Piao, Gouchol Pok, and Keun Ho Ryu. A novel DBSCAN-based defect pattern detection and classification framework for wafer bin map. *IEEE Transactions on Semiconductor Manufacturing*, 32(3):286–292, 2019.

[8] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2(2):169–194, 1998.

[9] Alex Bewley, Rajiv Shekhar, Sam Leonard, Ben Upcroft, and Paul Lever. Real-time volume estimation of a dragline payload. In *2011 IEEE International Conference on Robotics and Automation*, pages 1571–1576. IEEE, 2011.

[10] ÖZGE PASİN and HANDAN ANKARALI. Usage of Kernel K Means and DBSCAN cluster algorıthms in health studies An application. 2015.

[11] Eamonn Keogh and Abdullah Mueen. Curse of Dimensionality. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 314–315. Springer US, Boston, MA, 2017.

[12] Rui Ding, Qiang Wang, Yingnong Dang, Qiang Fu, Haidong Zhang, and Dongmei Zhang. Yading: Fast clustering of large-scale time series data. *Proceedings of the VLDB Endowment*, 8(5):473–484, 2015.

[13] Xin Lu, Yu Wang, Jiao Yuan, Xun Wang, Kun Fu, and Ke 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 (MLBDBI)*, pages 249–256. IEEE, 2020.

[14] Avory Bryant and Krzysztof Cios. RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1109–1121, 2017.

[15] Manisha Naik Gaonkar and Kedar Sawant. AutoEpsDBSCAN: DBSCAN with Eps automatic for large dataset. *International Journal on Advanced Computer Theory and Engineering*, 2(2):11–16, 2013.

[16] Zhihan Li, Youjian Zhao, Rong Liu, and Dan Pei. Robust and rapid clustering of kpis for large-scale anomaly detection. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2018.

[17] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[18] Jeong-Hun Kim, Jong-Hyeok Choi, Kwan-Hee Yoo, and Aziz Nasridinov. AA-DBSCAN: An approximate adaptive DBSCAN for finding clusters with varying densities. *The Journal of Supercomputing*, 75(1):142–169, 2019.

[19] Chen Xiaoyun, Min Yufang, Zhao Yan, and Wang Ping. GMDBSCAN: Multi-density DBSCAN cluster based on grid. In *2008 IEEE International Conference on E-Business Engineering*, pages 780–783. IEEE, 2008.

[20] Peng Liu, Dong Zhou, and Naijun Wu. VDBSCAN: Varied density based spatial clustering of applications with noise. In *2007 International Conference on Service Systems and Service Management*, pages 1–4. IEEE, 2007.

[21] Anant Ram, Sunita Jalal, Anand S. Jalal, and Manoj Kumar. A density based algorithm for discovering density varied clusters in large spatial databases. *International Journal of Computer Applications*, 3(6), 2010.

[22] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer, 2013.

[23] Yaobin He, Haoyu Tan, Wuman Luo, Huajian Mao, Di Ma, Shengzhong Feng, and Jianping Fan. Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce. In *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, pages 473–480. IEEE, 2011.

[24] Yanwei Yu, Jindong Zhao, Xiaodong Wang, Qin Wang, and Yonggang Zhang. Cludoop: An efficient distributed density-based clustering for big data using hadoop. *International Journal of Distributed Sensor Networks*, 11(6):579391, 2015.

[25] Emad A. Mohammed, Behrouz H. Far, and Christopher Naugler. Applications of the MapReduce programming framework to clinical big data analysis: Current landscape and future trends. *BioData mining*, 7(1):1–23, 2014.

[26] Shuigeng Zhou, Aoying Zhou, Jing Cao, Jin Wen, Ye Fan, and Yunfa Hu. Combining sampling technique with DBSCAN algorithm for clustering large spatial databases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 169–172. Springer, 2000.

[27] Bhogeswar Borah and Dhruba K. Bhattacharyya. An improved sampling-based DBSCAN for large spatial databases. In *International Conference on Intelligent Sensing and Information Processing, 2004. Proceedings Of*, pages 92–96. IEEE, 2004.

[28] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005(09):P09008, 2005.

[29] Pasi Fränti and Sami Sieranoja. K-means properties on six clustering benchmark datasets. *Applied Intelligence*, 48(12):4743–4759, December 2018.

[30] Scikit-learn: Machine learning in Python — scikit-learn 1.0.2 documentation. https://scikit-learn.org/stable/.

[31] Vidhi Mistry, Urja Pandya, Anjana Rathwa, Himani Kachroo, and Anjali Jivani. AEDBSCAN—Adaptive Epsilon Density-Based Spatial Clustering of Applications with Noise. In *Progress in Advanced Computing and Intelligent Engineering*, pages 213–226. Springer, 2021.