IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 2, 2019, accepted December 11, 2019, date of publication December 16, 2019,
date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960159

A Multi-Density Clustering Algorithm Based on Similarity for Dataset With Density Variation

XINGXING ZHOU[1,2], HAIPING ZHANG[2], GENLIN JI[1,2], AND GUOAN TANG[2]
[1]School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China
[2]School of Geographic Science, Nanjing Normal University, Nanjing 210023, China

Corresponding author: Genlin Ji (glji@njnu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 41971343.

ABSTRACT Clustering has been widely used in the fields of knowledge discovery, pattern recognition and artificial intelligence. However, discovering clusters in spatial databases is still a challenging task, especially when the shape, size, and density of clusters vary a lot. Existing algorithms have sensitive parameters, clusters must be separated far enough from each other and rich prior knowledge about datasets is required. In this paper, we propose algorithm DENSS, which performs clustering on the basis of the similarity of neighbour distribution and the number of shared neighbors for two objects. Algorithm DENSS can mine clusters that differ in densities, and within a cluster the local densities are reasonably homogeneous. Adjacent objects are separated into different clusters by significant change in densities. To verify the effectiveness of the algorithm DENSS, synthetic and real-world datasets are used for testing, and it has been compared with seven clustering algorithms. Experimental results show that the proposed algorithm has a relatively high efficiency, robustness and effectiveness, and is remarkably superior to the seven algorithms. This algorithm is universal and can rapidly and efficiently identify the clusters of different densities, shapes and sizes even in the presence of noise and outliers for any object feature types.

INDEX TERMS Similarity measurement, multi-density clustering, arbitrary shaped clustering, varied density.

I. INTRODUCTION

Clustering aims to divide objects in the datasets with different properties into their corresponding categories and has been widely used in knowledge discovery [1], pattern recognition [2], [3] and artificial intelligence [4], [5]. However, clustering still faces challenges that must be solved urgently. For example, sizes and densities of the clusters are different or the shapes of clusters are irregular. Moreover, the clustering results are unsatisfactory when no obvious division amongst different clusters exists, when the datasets contain many noise points (noise) or when users have insufficient prior knowledge of the datasets. Thus, researchers have proposed various solutions to address these challenges. Amongst the most promising algorithms is density-based clustering, which can identify clusters of any shape and is extremely robust against noise. In addition, users are not required to know the number of clusters beforehand.

The associate editor coordinating the review of this manuscript and approving it for publication was Junchi Yan.

Density-based algorithms can be divided into two categories. The first category is the clustering algorithms represented by DBSCAN [6]. However, there are two evident shortcomings in DBSCAN. Firstly, its performance depends on two parameters, namely, eps and minpts. However, without sufficient prior knowledge, determining these two parameters is difficult. Secondly, when a dataset contains multiple clusters with different densities, identifying the clusters through a set of thresholds is challenging. Figure 1(a) presents two clusters with different densities. The density of the clusters in

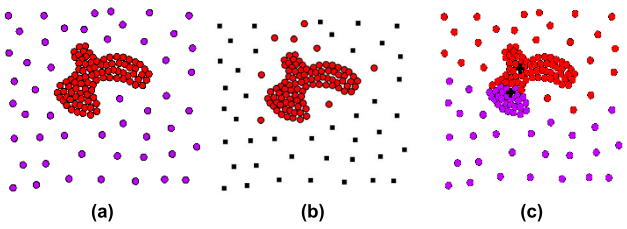


FIGURE 1. Results of the traditional DBSCAN and DPC algorithm. (a) Origin dataset. (b) Result of DBSCAN. (c) Result of DPC.

the middle is high, whilst those of the surrounding clusters are low. Figure 1(b) indicates that DBSCAN can only identify the cluster with high density (the cluster consists of red circular dots in the middle) because of the different densities of the two clusters. Meanwhile, the surrounding circular cluster is regarded as noises because of their low density (the surrounding black squares). The two clusters can only be identified as one cluster by lowering the density threshold. The problem arises because such algorithms have a fixed density threshold value [7], [8]. Therefore, only clusters that are larger than the density threshold can be identified, and clusters with a smaller density than the density threshold are treated as noise. The second category refers to the clustering algorithm represented by density peak clustering (DPC) [7]. The algorithm posits that a cluster centre should have a local density higher than those of its neighbours and a relatively larger distance from any point with a higher local density. Thus, the centre points must be selected first, and each remaining point must be assigned subsequently to the nearest cluster with a higher local density. However, DPC has two apparent drawbacks. On the one hand, in the clustering process of two or more clusters with different densities but close proximities, the calculation results of local densities may have deviations, thereby resulting in the failure of final clustering. On the other hand, when multiple extreme points exist in a cluster, a cluster may be forcibly divided into several parts. Figure 1(c) shows that DPC selects both centres (black crosses) in the clusters with a higher density, resulting in serious errors in the clustering results.

To solve the above problems, we propose a new clustering algorithm that identifies similar objects as a cluster based on the similarity between objects.

This similarity mainly includes two aspects: the similarity of neighbor distribution and the similarity of shared neighbors for the two objects. Based on this, objects with similar neighbor distribution and enough shared neighbors can be recognized as a cluster, to meet the requirements of multi-density clustering. So, in a cluster the local densities are reasonably homogeneous. The main contributions of this research presented are as follows:

(1) A new multi-density clustering algorithm is proposed. This algorithm can identify clusters of any size, any shape and any density, and it is robust against noise. In addition, even clusters without distinct division zones can be identified perfectly. In this algorithm, the parameters are simple to set, and insensitive, and not require users to have a rich prior knowledge.

(2) The measurement of the similarity between two objects is well defined, consisting of the similarities of divergence and the similarity of shared neighbors of the two objects. The new measurement can help an object determine similar neighbors rapidly and efficiently.

(3) The proposed algorithm is verified through synthetic and real-world datasets, and evaluated using three evaluation indexes, including Fowlkes–Mallows index (FMI) [9], adjusted Rand index (ARI) [10], and adjusted Mutual information (AMI) [10]. The results of DENSS are more accurate than those of other clustering algorithms such as DPC and DBSCAN.

II. RELATED WORKS

Clustering algorithms can be divided into four major categories, namely, partitioning-, hierarchy-, density- and grid-based algorithms. However, only density-based algorithms can recognise clusters with any shapes and densities. Thus, this work mainly discusses and analyses existing density-based algorithms.

DBSCAN is a classic density-based clustering algorithm. However it cannot handle the density variation in data. To address the difficulty of setting parameters, GMDBSCAN [11] divides the data space into grids, identifies a local minpts for each grid, and performs clustering by applying corresponding minpts to each grid. In addition, to handle density differences, the concept of homogenous core point is proposed in DDSC [12], that is, the density difference of all neighbors of this point is less than the specified threshold. Based on this concept, DDSC selects a random homogenous core point to begin clustering, and continuously adds other homogenous core points that the density can reach until no more point can be reached. Subsequently, a cluster is formed. However, as the algorithm uses a global radius parameter (ε) similar to DBSCAN when calculating the density of each point, no neighbour may exist within radius ε for points in clusters with low density. Therefore, when the densities of different clusters in datasets vary, clusters with lower density cannot be identified effectively. CDBSCAN [13] partitions the dataset into neighbors with a minimum number of data points and then builds local clusters. However, this algorithm tends to generate excessive singleton clusters. SSDBSCAN describes the use of labeled points to help the algorithm detect suitable density parameters to extract density-based clusters. However, SSDBSCAN [14] decreases the clustering performance when applied on a large dataset that contains severely overlapping samples from different clusters. The limitation of CDBSCAN and SSDBSCAN is their need for abundant prior knowledge.

Another important clustering algorithm is DPC in which the density of the cluster's centre point is greater than the density of the non-central points, and the centre points are far from one another. Thus, the cluster centre can be identified first, allowing the assignment of a point to a cluster according to the distance of other points from different cluster centres [15]. Many improved clustering algorithms, such as multiple DPC (MDPC) [16], PPC [17], FDPCluster [18] and DPCG [19], have been developed on the basis of DPC.

However, these algorithms tend to select points with a large density as the initial cluster centres which may result in the incorrect assignment of points with lower density to other clusters or may be treated directly as noise. The shared nearest neighbours (SNN) algorithm uses the number of shared neighbours of the two objects as a criterion for judging whether the two objects are similar or not [20]. Although this

algorithm addresses the issue of multiple-density clusters in a dataset to some extent, identifying two clusters without significant separation zones may not be accurate because the k-nearest neighbors based on the distance of an object is not necessarily on the same level of the object [21]. Even with the integrated merits of the SNN and DPC algorithms, SNN-DPC [22] may not accurately identify two clusters without evident separation zones. Furthermore, the execution process requires users to specify the number of clusters or even the centre points.

III. ALGORITHM DESIGN

In this section, algorithm DENSS is proposed which is based on the similarity of divergence and shared neighbors. The algorithm can rapidly and efficiently identify clusters of different densities, shapes and sizes even in the presence of noise and outliers. And within the same cluster the local densities are reasonably homogeneous. The algorithm can be mainly divided into two stages. 1) Neighbor correction: get true similar neighbors for each object. 2) Clustering is performed on the basis of the similarity of divergence and shared neighbors amongst objects.

A. NEIGHBOR CORRECTION

Formally, let $\mathrm{P} = \{p_1, p_2, \ldots, p_n\}$ be a point dataset of n points with m-dimensional features. The i-th object can be represented as $p_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$. For the purpose of proper visualization, 2D space datasets will be employed to demonstrate the algorithm. To facilitate presentation the subsequent clustering steps, the following definitions are given:

Definition 1 (K-Nearest-Neighbor Neighborhood (KNN Neighborhood)): given a set with n points P, for point $p_i (p_i \in \mathrm{P})$, its KNN neighborhood set is $\mathrm{KNN}(p_i)$. If $\forall p_x \in \mathrm{KNN}(p_i)$, $\forall p_y \notin \mathrm{KNN}(p_i)$ and $(p_x \in \mathrm{P}, p_y \in \mathrm{P})$, then $\mathrm{Distance}(p_x, p_i) \leq \mathrm{Distance}(p_y, p_i)$, where Distance() is the Euclidean distance between two points, and $|\mathrm{KNN}(p_i)| = k$. Note, that the proposed algorithm can work with any distance function.

Definition 2 (Noise Point): For cluster $\mathrm{C} = \{p_1, p_2, \cdots, p_j\}$, if the number of points in C is less than k ($|C| < k$), then all points in C are considered noise. For conciseness, noise point is abbreviated as noise.

Definition 3 (Shared-Nearest-Neighbors Neighborhood (SNN Neighborhood)): The shared neighbors of two points p_i and p_j in set P are the intersections of their neighbors, which are represented by $\mathrm{SNN}(p_i, p_j) = \{p_x | p_x \in \mathrm{KNN}(p_i) \text{ and } p_x \in \mathrm{KNN}(p_j)\}$, and are symmetric, that is, $\mathrm{SNN}(p_i, p_j) = \mathrm{SNN}(p_j, p_i)$. In other words, $|\mathrm{SNN}(p_i, p_j)|$ represents the number of shared neighbors of p_i and p_j.

If two points in the SNN algorithm share a certain number of neighbours, then they may be similar and belong to the same cluster. The similarity of the SNN neighbourhood can be expressed by the number of shared neighbours. Figure 2 shows two clusters with different densities when $k = 8$; the k nearest neighbors of p_A, p_B, p_C, p_D and p_E are marked by arrows. As p_A and p_B are near to each other,

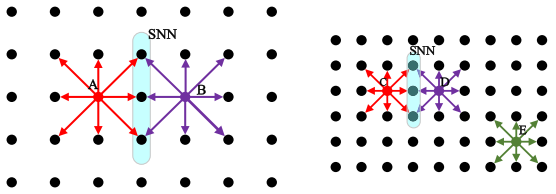


FIGURE 2. SNN neighborhood in different clusters.

they share neighbors, and $|\mathrm{SNN}(p_A, p_B)| = 3$. Similarly, p_C and p_D are close to each other, and $|\mathrm{SNN}(p_C, p_D)| = 3$. As the distance from p_C and p_D to p_A and p_B are relatively far, they have no shared neighbor, that is, $|\mathrm{SNN}(p_A, p_C)| = |\mathrm{SNN}(p_A, p_D)| = |\mathrm{SNN}(p_B, p_C)| = |\mathrm{SNN}(p_B, p_D)| = 0$. Furthermore, although p_E, p_C and p_D belong to the same cluster, p_C and p_D are far from p_E. Thus they do not share any neighbor, that is, $|\mathrm{SNN}(p_C, p_E)| = |\mathrm{SNN}(p_D, p_E)| = 0$. To sum up, two objects in a cluster that are near each other normally share a certain number of neighbours. In contrast, two objects from a same cluster that are far apart may not share any neighbour, and two objects in two different clusters do not share any neighbour. Furthermore, the number of neighbours shared by adjacent objects in two clusters of different densities is similar. Thus, algorithm SNN can identify clusters with different sizes, shapes and densities in a dataset.

However, algorithm SNN also has some defects. 1) Noise points may be incorrectly divided into nearby clusters because the algorithm SNN is sensitive to noise. In Figure 3, for point p_E in cluster 2, relevant arrows point to its 8 nearest neighbors, and the eight nearest neighbors of noise point p_F are denoted by blue arrows. Clearly, their neighbours are completely the same ($|\mathrm{SNN}(p_E, p_F)| = 8$). The similarity is even greater than that between p_A and p_B, which both belong to cluster 1, $|\mathrm{SNN}(p_A, p_B)| = 3$. In this case, noise point p_F may be classified into cluster 2. 2) When two clusters of different densities are near each other, or no distinct separation zone exists between the two clusters, the border points may be incorrectly clustered.

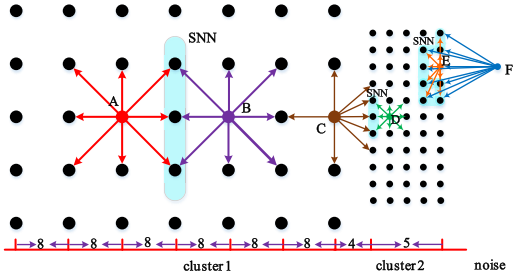


FIGURE 3. Some problems about algorithm SNN.

In Figure 3, p_C evidently belongs to cluster 1, but p_C seems to have more shared neighbors with points in cluster 2, (i.e. $|\mathrm{SNN}(p_C, p_D)| = 4$), which is far greater than $|\mathrm{SNN}(p_C, p_B)| = 1$. Thus, algorithm SNN is likely to classify the border points between the two clusters incorrectly and, in some cases, may even mistakenly integrate the two clusters into one.

To solve the aforementioned problems, we propose the concept of divergence, which is defined as follows:

Definition 4 (Divergence of Point): The divergence of point p_i is represented by the average distance between any two points in the set which composed of p_i and its neighbors,

$$p_i.div = \frac{(\sum_{j=1}^{k} distance\,(p, q_j)}{(k+1)\,k}, \quad p, q \in \{p_i\} \cup \{\mathrm{KNN}\,(p_i)\,.$$

Definition 5 (Divergence Similarity between two points (DS)): it can be represented by the ratio of the smaller divergence value to the larger divergence value.

$$DS(p_i, p_j) = \frac{\min(p_i.div, p_j.div)}{\max(p_i.div, p_j.div)}$$

The range of $DS(p_i, p_j)$ is $(0, 1]$, and the closer the value to 1 is, the greater the similarity of the divergence between the two points will be, and vice versa.

Definition 6 (K-Nearest-Neighbor Based on Divergence (KNND)): in set P with n points, for $p_i(p_i \in \mathrm{P})$, obtain the K nearest neighbors that are most similar to its divergence, which can be represented by KNND (p_i). And it must meet the following requirement: if $\forall p_x \in \mathrm{KNND}\,(p_i)$ and $\forall p_y \notin \mathrm{KNND}\,(p_i)$, then $DS\,(p_x, p_i) \geq DS\,(p_y, p_i)$.

We can obtain the divergences of p_f and p_e based on the above definition (Figure 3). Evidently, $p_f.div$ is larger than $p_e.div$, that is, their DS is very small. According to the rule of DS, p_f cannot be merged into cluster 2. Furthermore, no point of divergence similar to p_f exists, and p_f is successfully recognised as noise. To sum up, noise and non-noise points cannot be combined in the same cluster because of the great difference in their divergence values. Therefore, the concept of divergence can effectively and accurately identify noise, thereby greatly improving the robustness of the algorithm.

Definition 7 (Cluster Divergence): The divergence value of cluster $C = \{p_1, p_2, \cdots, p_{\mathrm{m}}\}$ is the average divergence value of all points in the cluster, which can be represented by $C.div = (\sum_{i=1}^{m} p_i.div)/\mathrm{m}$.

Definition 8 (Similarity between Point and Cluter (SPC)): To obtain the similarity between point p and cluster $C = \{p_1, p_2, \cdots, p_{\mathrm{m}}$, we define a function as follows:

$$\mathrm{SPC}\,(p, C) = \frac{\sqrt{\min\,(p.div, C.div)}}{\sqrt{\max(p.div, C.div)}} * \frac{\sqrt{\min(dis(p, C), C.div)}}{\sqrt{\max(p.div, C.div)}}$$

where $dis(p, C)$ refers to the smallest distance between p and the points in cluster C. The product is taken here to ensure that SPC (p, C) will achieve a higher value only if the values of $p.div$ and $dis(p, C)$ are similar to those of $C.div$.

Taking the right half of the baseline dataset (Compound) as an example (Figure 4), when k = 10, the KNN neighborhood of p_{25} are $\{p_{96}, p_{91}, p_{97}, p_{90}, p_{98}, p_{92}, p_{122}, p_{26}, p_{93}, p_{121}\}$. Among the 10 nearest neighbors, only one neighbor (p_{26}) is correct, and the divergence of p_{25} is $p_{25}.div = 1.84$. The neighbors of p_{98} are $\{p_{100}, p_{99}, p_{97}, p_{95}, p_{101}, p_{94}, p_{71}, p_{96}, p_{72}, p_{103}\}$, and $p_{98}.div = 0.93$. The neighbors of p_{30} are $\{p_{29}, p_{36}, p_{31}, p_{27}, p_{37}, p_{28}, p_{33}, p_{26}, p_{35}, p_{87}\}$, and $p_{30}.div = 3.411$. According to the similarity of shared neighbors and divergence, p_{25} can only be recognized as a noise point. However, p_{25} and p_{30} should belong to the same cluster. In this dataset, there are many points similar to p_{25}, such as $\{p_{40}, p_{45}, p_{46}, \cdots\}$. Therefore, finding the correct neighbors of a point is essential to the algorithm. Algorithm 1 is designed by this work to determine the true similar neighbours of an object.

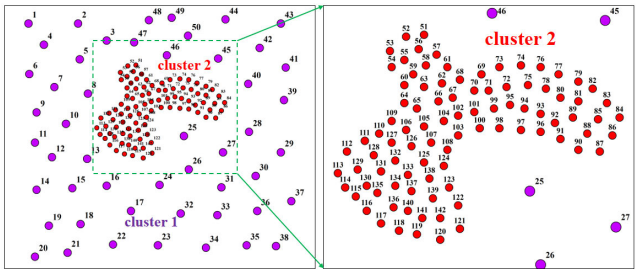


FIGURE 4. The KNN neighborhood of p_{25}.

Algorithm 1 Neighbor_Correct

Input: dataset P, number of neighbors: k, the thread of divergence similarity between two points : θ
Output: points with really similar neighbors
Process:

```
1.  get KNN neighborhood for each point
2.  get divergence for each point based on KNN
    neighborhood
3.  for each point p_i in dataset P
4.    similarNeighborNum = diversitySimilarNeighbor
    (p_i, θ);//get the number of neighbors whose
    divergence is similar to p_i.
5.    if similarNeighborNum < k
6.      muti_Neighbors = p_i.getMultiNeighbor();//Get
    more neighbors (5*k )
7.      sort muti_Neighbors in ascending order of
    divergency
8.      candidateNeighborClusters ← ∅;
9.      while muti_Neighbors != ∅
10       create a new neighborCluster C ←∅;
11       C.add(corePoint = muti_Neighbors.pop());
12.       while ((neighbors ← getSimilarNeighbor
    (corePoint, θ))!= ∅)//get the
    points whose diversity is similar to corePoint
13         C.add(neighbors);
14         muti_Neighbors = muti_Neighbors  neighbors;
15         corePoint = C.getNextPoint();
16       end while
17       candidateNeighborClusters.add(C);
18     end while
19     p_i.trueNeighbors = getTrueNeighbor(p_i,
    candidateNeighborClusters);// get the most
    similar neighbor cluster by definition 8
20     update the divergence of p_i
21   end if
22 end for
```

In algorithm 1, the k-nearest points are obtained and calculated the divergence for each point. Then check the neighbors for each point, whether there are neighbors whose divergence is greatly different from that of the point. To get the similar neighbors, we need to search for more neighbors, and sort the neighbors by their divergences. The neighbor point with the smallest divergence is selected as the starting point to clustering, and θ is used as the divergence similarity threshold. We can get two or more clusters after all neighbors have been visited. Then, we can get the most similar neighbor cluster by definition 8, and update the divergence of point by the similar neighbors. Algorithm 1 is shown as neighbors correction.

According to Algorithm 1, when the DS threshold (θ) is set to 0.4, then obtain the 50 neighbors for p_{25}, as indicated by the red dots in Figure 5(a). Subsequently, these expanded neighbours are clustered, and they can be easily divided into clusters 1: $\{p_{51}, p_{52}, p_{54}, \cdots\}$ and 2: $\{p_{26}, p_{27}, p_{24}, \cdots\}$. Finally, the cluster that is the true neighbour is determined by comparing the DS between p_{25} and these two clusters. Apparently, the similarity between p_{25} and cluster 2 is higher than that between p_{25} and cluster 1. The K nearest points of p_{25} are selected as true neighbours in cluster 2, that is, $\{p_{26}, p_{27}, p_{24}, p_{31}, p_{28}, p_{32}, p_{30}, p_{40}, p_{45}, p_{33}\}$.

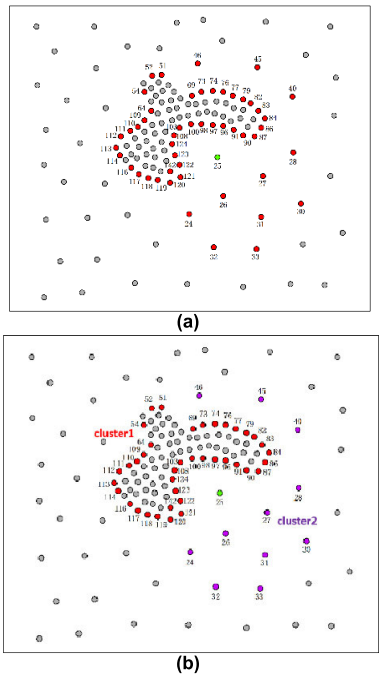


FIGURE 5. Get the true neighbors of p_{25}. (a) 50 neighbors of p_{25}. (b) Result of clustering neighbors of p_{25}.

B. ALGORITHM DESCRIPTION

Through the corrections of neighbour and divergence, the points in the same cluster should be similar in the two aspects, that is, the neighbours of the two points and their divergence are similar, as shown in Definition 9.

Definition 9 (Similarity Between Two Points): p_A and p_B are two points in dataset P. We define a function to determine the similarity between p_A and p_B as follows:

$$\text{Similarity}\,(p_A, p_B) = \begin{cases} 1, & \dfrac{\min\,(p_A.div, p_B.div)}{\max\,(p_A.div, p_B.div)} > \theta \text{ and } \dfrac{|SNN\,(p_A, p_B)\,|}{K} > \varepsilon \\ 0, & otherwise \end{cases}$$

where the theoretical range of θ is (0,1]. In practice, the range is normally [0.6,0.85] because if it is extremely small, the dissimilar points will be identified in the same category. However, when the threshold is too big, the points in the same category will be divided into multiple clusters. The theoretical value range of ε is [0, 1]. However, the practical value is [0.3, 0.5]. In Figure 2, p_A and p_B are two relatively near points in the same cluster, and their $|\text{SNN}(p_A, p_B)| = 3$, that is, the similarity of their neighbors is 3/8. In this paper, we take $\varepsilon = 0.4$, according to experience.

In algorithm 2, the k-nearest points are obtained for each point and calculated the divergence by definition 4. Neighbor correction is performed for each point by Algorithm 1. Then sort all points in ascending order of divergence. The clustering process starts from the point p with the smallest divergence. Add p into cluster C, and retrieve all similar neighbors of p. Then, each unvisited point in cluster C, is marked as visited, and its neighbours are located and added to cluster C in the same way until each point in cluster C is visited.

Algorithm 2 DENSS

Input: point set: P, number of neighbors: k, divergence similarity threshold between points:θ, neighbors similarity of two points: ε.
Output: Clusters $= \{C_1, C_2, \cdots, C_m\}$ (m is the number of clusters)
Process:
1. Clusters $= \emptyset$;
2. get KNN neighborhood and divergence for each point in P;
3. Neighbor_Correct(P, k, θ);
4. P.sortByDIV(ascending);//sort all points in ascending order of divergence
5. for each unvisited p with smallest divergence in P
6. C $= \emptyset$;
7. C.add(p);//add p into cluster C
8. mark p as visited;
9. C.add(p. getNeighbors(θ, ε));//get points similar to p and add them to cluster C;
10 for each unvisited p' in C
11 C.add(p'.getNeighbors(θ, ε));//get points similar to p' and add them to cluster C;
12 mark p' as visited;
13 end for
14 Clusters.add(C);
15 end for

Then a complete cluster is formed. The operation is repeated until all points in set P are visited and all clusters are obtained. The detailed algorithm DENSS process is shown in algorithm 2.

Execution time complexity of the algorithm is mainly determined by searching for KNN neighborhood and sort all points in ascending order of divergence. In the process of finding neighbors for each point, since index cannot be created for high-dimensional data, the time complexity of this stage is $O(n^2)$. The time complexity in sorting is $O(nlog_2 n)$ by utilized the heap sorting strategy. To sum up, time complexity of algorithm DENSS is $O(n^2)$.

IV. EXPERIMENTS AND ANALYSIS

To verify the effectiveness of DENSS, we verify its performance using classic benchmark synthetic and real-world datasets for the purpose of proper visualization. Given that this paper focuses on multi-density clustering, two synthetic datasets that contain multiple clusters of different densities were designed to fully demonstrate the performance of DENSS. However, their evaluation indexes are not as simple as the calculation of the number of errors or the accuracy and recall rate in the classification algorithms because clustering algorithms are unsupervised algorithms. The evaluation criteria of classification focus on the accuracy of the object that belongs to a category label, while clustering obtains one or more clusters without labels. The latter focuses more on the similarity of internal data and the divergence of clusters. To obtain clustering results, focus should be given to the relationships amongst data instead of the relationship between data and category. Therefore, in this section, three validation metrics, namely, the FMI, the ARI and AMI are applied to evaluate the performance of the algorithms. The upper limit for these three indexes is 1, and the higher the value is, the better the clustering results are.

A. EXPERIMENTS

We select some benchmark datasets that are widely used to test the performance of various clustering algorithms. Two types of datasets are mainly used. Firstly, the four shape datasets with natural shape clusters include Aggregation [23], Flame [24], Pathbased [25] and R15 [26]. Secondly, the five datasets with uneven density-pattern clusters include Jain [27], Compound [28], two-circle three-circle and iris dataset [29]. The clusters in all these datasets have different shapes, densities and scales. These datasets can simulate different scenarios through which the performances of different clustering algorithms can be compared. Next, we present the clustering results of these datasets in the tests. In the figures, the dots of different colours represent the classification into different clusters, and the black squares represent noise.

As shown in Figure 6 most algorithms, including DENSS, SNN-DPC, DPC, DBSCAN and OPTICS [30], can accurately identify the clusters in Aggregation. However, error occurs in the identification process of MDPC and K-Means. MDPC tends to identify a single cluster as two or identify two close clusters as one incorrectly. K-Means experiences the same problem by dividing the lower left circular cluster into three parts.

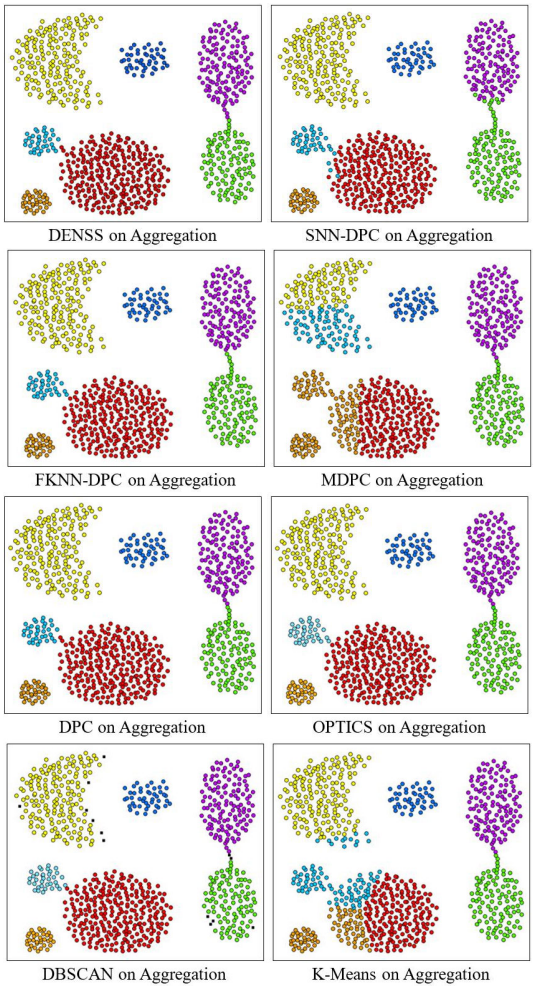


FIGURE 6. The clustering results on aggregation.

Figure 7 shows that Pathbased consists of a semi-ring cluster and two nearly circular clusters. As shown in the figure, only DENSS and SNN-DPC can guarantee the correct identification of the three clusters. For other algorithms, the results can be divided into two kinds. The first type includes those obtained by DPC, MDPC and K-Means where the left and right sides of the semi-ring cluster were incorrectly assigned to the two clusters in the middle, thereby reducing the semi-ring cluster to only its top part. FKNN-DPC has a similar problem, only that it identified the semi-ring and left clusters as a whole. The second type of results comes from OPTICS and DBSCAN which can correctly identify the two clusters in the semi-ring cluster. However, OPTICS divided the semi-ring into several small clusters. Owing to the low density, the entire semi-ring cluster was recognized by DBSCAN as noise.

Figure 8 shows the results on Flame by all algorithms. DENSS, SNN-DPC, DPC and DBSCAN can identify the

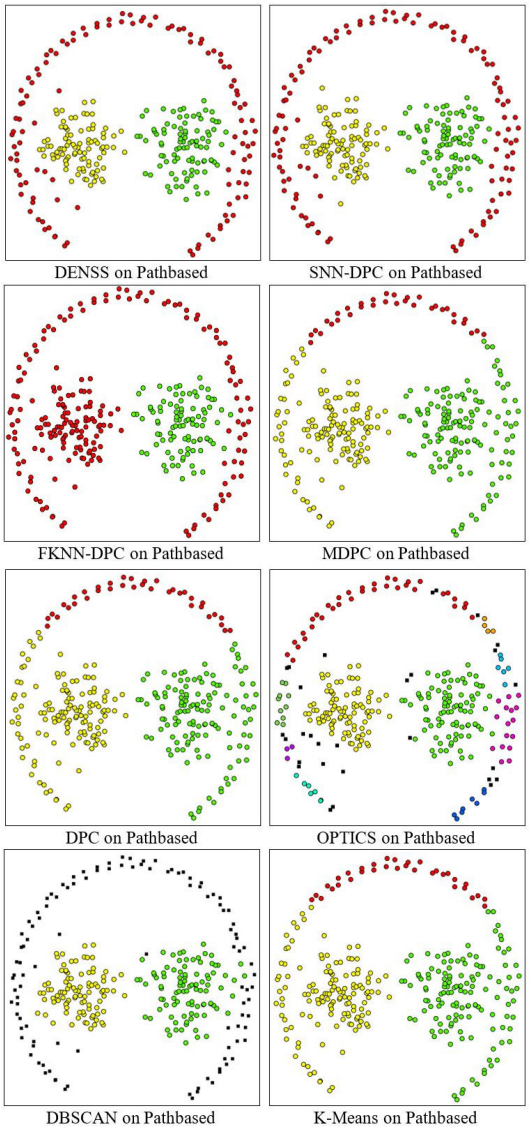


FIGURE 7. The clustering results on pathbased.

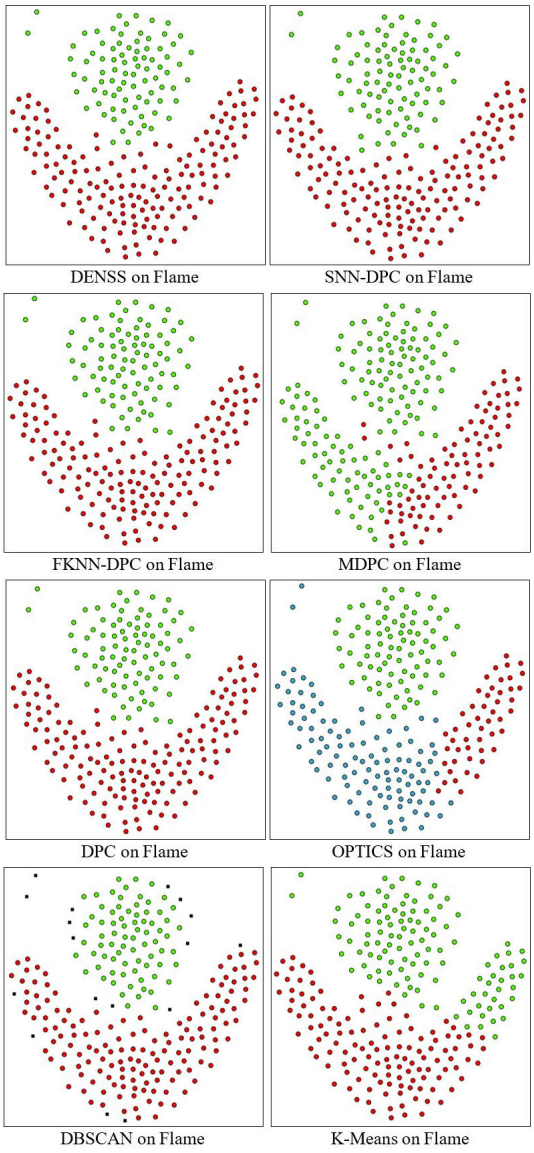


FIGURE 8. The clustering results on Flame.

clusters correctly. OPTICS divided the lower clusters into two, and the result was undesirable. K-Means assigned the left end of the lower cluster to the upper cluster. Similarly, MDPC assigned the right end of the lower cluster to the upper cluster, thereby presenting a dataset that appears to have been cut diagonally, resulting in serious errors.

Figure 9 shows the results of R15, which is the simplest data for all algorithms because of the nearly evenly distributed points in each cluster. Despite some small defects, most algorithms managed to identify them correctly. Only MDPC identified 8 clusters in the middle as one.

Figure 10 shows that in the Compound dataset, only DENSS successfully identified six clusters accurately. The other algorithms could only identify two dense clusters in the lower left corner or the two closed sparse circular clusters correctly. SNN-DPC, FKNN-DPC and MDPC relatively identified the three clusters on the left correctly, but because the two clusters were near each other, their densities were very different, and no obvious separation zone existed, thereby resulting in serious errors in the identification results of all three algorithms. Although OPTICS and DBSCAN identified the two clusters in the upper left correctly, they identified the two clusters in the lower left as one. DPC and K-Means have similar problems, that is, they both recognises the set of points on the lower left as two or more clusters, but the shape of the clusters are incorrect. Furthermore, the four algorithms failed to identify the two clusters on the right with very different densities correctly.

Figure 11 shows that Jain consists of two interwoven crescent-shaped clusters whose densities are different but close. DENSS, SNN-DPC and MDPC can flexibly handle this type of dataset, thereby completely distinguishing between these two clusters. The results of the other algorithms can be

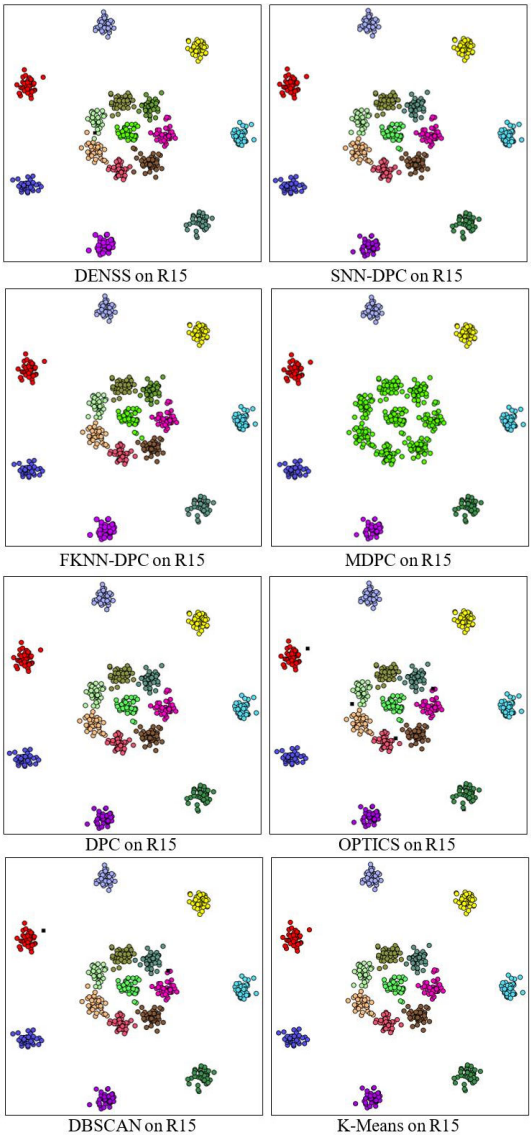


FIGURE 9. The clustering results on R15.

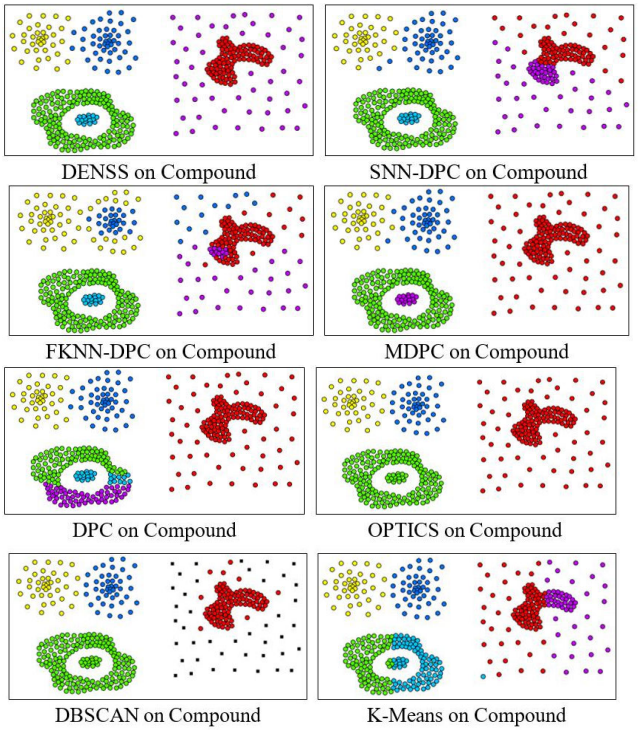


FIGURE 10. The clustering results on compound.

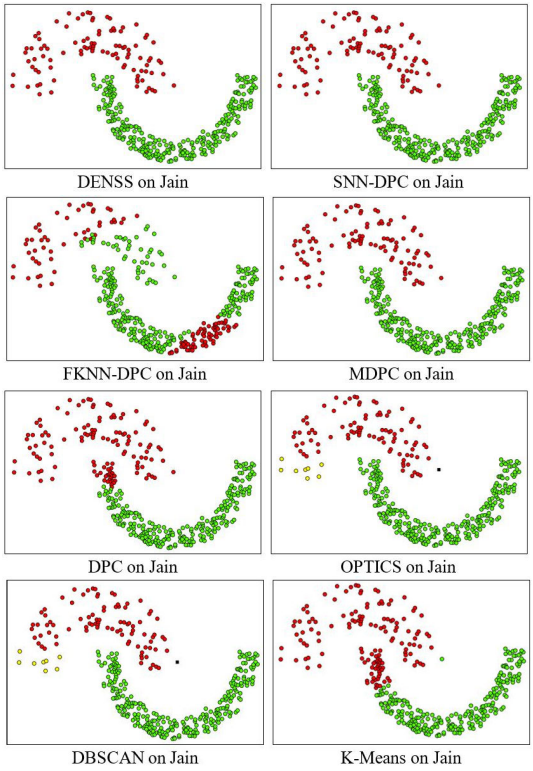


FIGURE 11. The clustering results on Jain.

divided into three kinds. The first is the results by DBSCAN and OPTICS which can accurately identify the lower cluster. However, both algorithms identified the left part of the upper cluster as a new cluster incorrectly. This error is closely related to the failure of traditional density-based clustering algorithms in dealing with variable density clustering. Although OPTICS optimised this problem, it cannot avoid the problem completely. The second results come from DPC and K-Means, which incorrectly assigned the left end of the lower cluster to the higher cluster. The third results come from FKNN-DPC, which not only incorrectly assigned a part of the upper cluster to the lower cluster, but also assigned a part of the lower cluster to the upper cluster.

In Figure 12, Two_Circle consists of a circle and two semi-rings different densities. Only DENSS can cluster this type of dataset accurately. The results of the other algorithms can be divided into three kinds. The first results come from SNN-DPC, which successfully divided the dataset into three clusters. Although the shape of each cluster has errors,

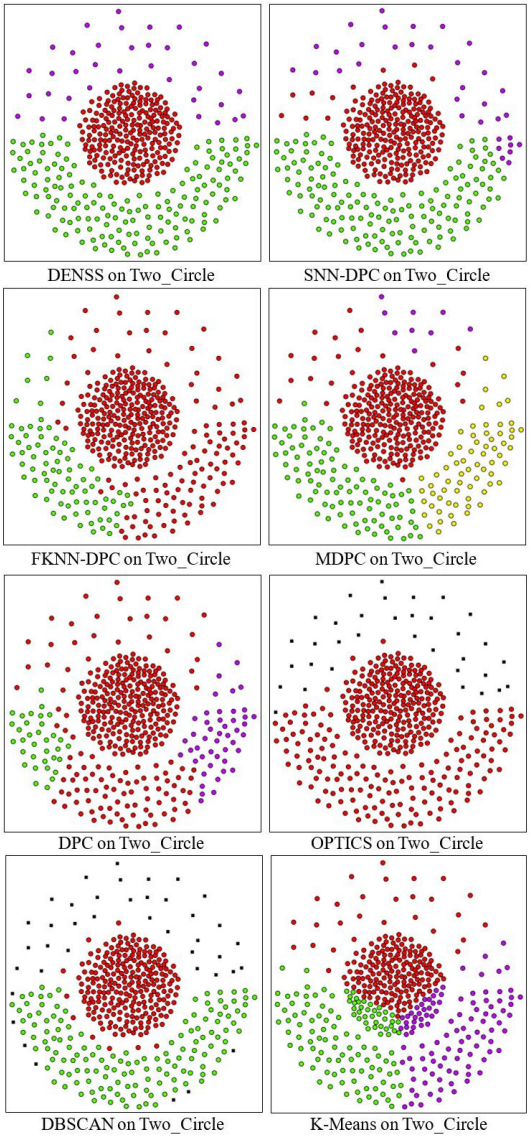


FIGURE 12. The clustering results on two_circle.

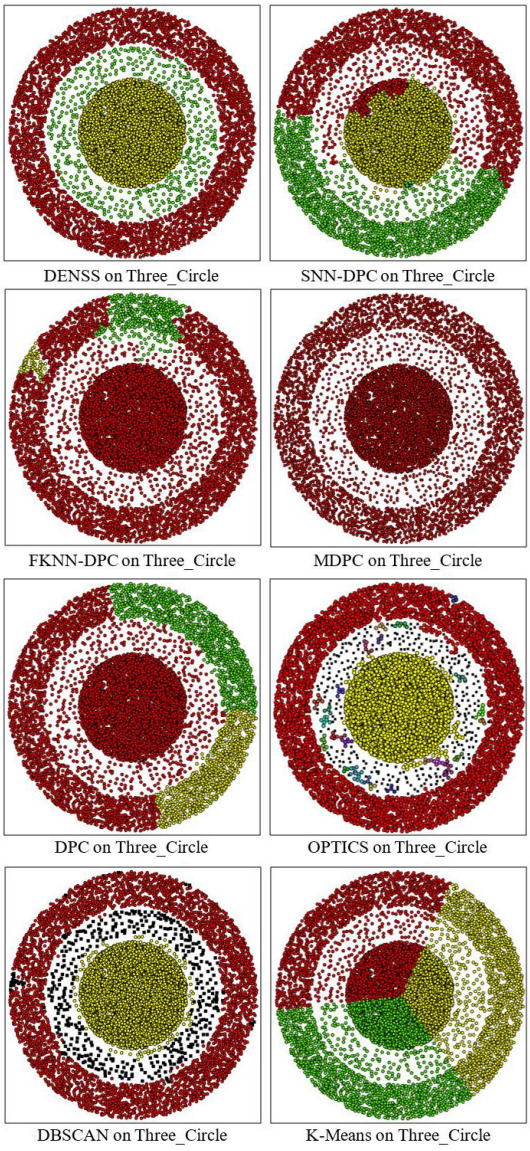


FIGURE 13. The clustering results on three_circle.

they are basically correct. The error is caused by the difference in the densities of the three clusters and their proximity. The second results were obtained by FKNN-DPC, MDPC, DPC and K-Means, which assigned some points in clusters of lower density incorrectly to clusters with higher density, thereby causing serious errors in the clustering results. The third results come from DBSCAN and OPTICS, which both identified the clusters with high densities, but recognised clusters with low density as noise correctly.

In Figure 13, Three_Circle consists of a circle and two rings which contain 7,850 points in total. The positions of the points in each cluster are randomly generated. This dataset contains the largest amount of data in all synthetic datasets. Their densities vary; the density of the middle circle is the largest, followed by the densities of the outermost and inner rings. Evidently, only DENSS can cluster such datasets accurately, and the boundaries of each cluster can be recognised precisely as well. The other algorithms produced serious errors, possibly because of the following reasons. Firstly, the densities of the three clusters are quite different, thereby causing certain errors in the process of searching for the cluster centre by SNN-DPC, FKNN-DPC, MDPC and DPC. OPTICS and DBSCAN can only identify two clusters with high density, and the circular cluster with the lowest density in the middle was simply treated as noise. K-Means can identify clusters in convex shapes better, but the three interwoven clusters posed a great challenge. Secondly, these three clusters are near one another, thereby making the clustering process quite difficult. For example, SNN-DPC identified the inner and outer rings as one, and MDPC even recognised the three clusters as a whole. Thirdly, the data are large, and the positions of the points in the cluster

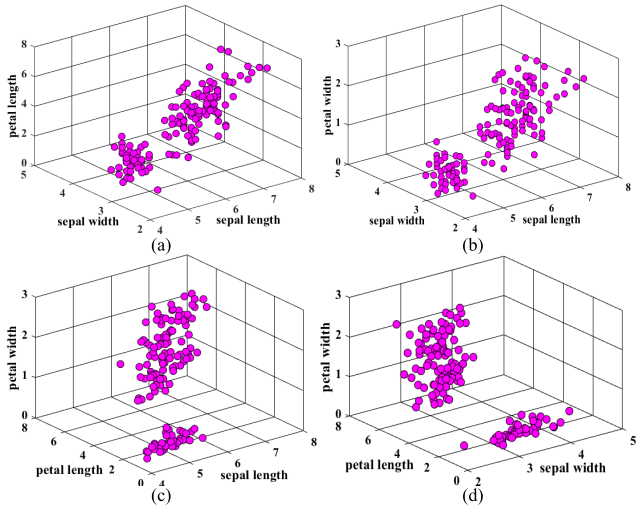


FIGURE 14. Visualization for the iris dataset with three random features. (a) Sub dataset with sepal length, sepal width and petal length. (b) Sub dataset with sepal length, sepal width and petal width. (c) Sub dataset with sepal length, petal length and petal width. (d) Sub dataset with sepal width, petal length and petal width.

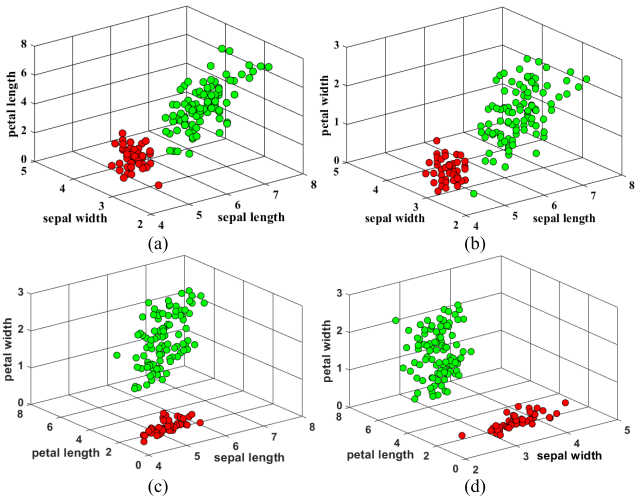


FIGURE 15. Clustering results of the iris dataset with three random features. (a) Sub dataset with sepal length, sepal width and petal length. (b) Sub dataset with sepal length, sepal width and petal width. (c) Sub dataset with sepal length, petal length and petal width. (d) Sub dataset with sepal width, petal length and petal width.

are randomly generated. Thus, the density may be locally higher or lower in the same cluster, which is in line with the distribution of points in real world. In recognising the inner ring cluster, OPTICS divided the cluster into several parts because of the uneven distribution of the points in the cluster.

In order to evaluate the usability and effectiveness of algorithm DENSS, we use the iris dataset (with four features) from the real-world dataset UCI [29]. In order to facilitate visualization, we take three features from the original four features randomly and visualize the four 3D sub datasets respectively, as shown in Figure 14. Obviously, according to the distribution of the 3D points in the four figures, they can be divided into 2 clusters. In order to reduce the effect of the difference between the density of the central part and that of the edge part of the clusters on the result, we set the number of neighbor k = 15, the divergence similarity threshold between points $\theta = 0.75$, and the neighbors similarity of two points: $\varepsilon = 0.4$. From the results in Figure 15, it can be seen that the two clusters in the four different datasets are successfully identified.

TABLE 1. Parameters of different clustering algorithms on different synthetic datasets.

	Compound	Jain	aggregation	Pathbased
DENSS	10, 0.73,0.4	10,0.8,0.4	10, 0.73,0.4	10, 0.68, 0.4
SNN-DPC	4,6	12, 2	15, 7	9, 3
FKNN-DPC	11, 6	10, 2	20, 7	9, 3
MDPC	0.05,1	0.1, 0.5	0.05, 1.5	0.08, 0.6
DPC	4,6	0.9, 2	3.4, 7	3.8, 3
OPTICS	1.5, 6	1.3, 8	1.8, 10	0.9, 4
DBSCAN	0.5, 6	0.8, 2	0.4,6	0.8, 10
K-Means	6	2	7	3
	Flame	R15	Two_Circle	Three_Circle
DENSS	10, 0.8, 0.4	10,0.5, 0.4	10, 0.75, 0.4	10, 0.68, 0.4
SNN-DPC	5, 2	10, 15	3, 3	6, 3
FKNN-DPC	6, 2	27, 15	5, 3	8, 3
MDPC	0.1, 0.65	0.06, 0.5	0.06, 0.5	0.06, 0.5
DPC	2.8, 2	0.6, 15	2.8, 3	2.6, 3
OPTICS	1, 6	0.5, 11	5.5, 6	1.5, 6
DBSCAN	0.9, 8	0.4, 12	2.6, 9	1.7,10
K-Means	2	15	3	3

B. PERFORMANCE ANALYSIS

Table 1 shows the parameters set by each algorithm for different datasets to obtain the best results. Figure 16 presents the FMI ARI and AMI scores of the clustering results by the eight selected datasets. Although SNN-DPC, FKNN-DPC and traditional DPC can spot the cluster centre via the decision diagram, to improve the accuracy of the results and accelerate the process of clustering, the correct number of clusters is directly specified. Similarly, the initial centre for K-means clustering is also nominated. Thus, the result of each algorithm is optimal. DENSS can not only obtain highly accurate results in the datasets with different shapes, but also handles datasets with different densities very well.

To evaluate the efficiency of clustering algorithms, the time consumption of different datasets in the same environment by different clustering algorithms are compared. We performed the experiments on a computer with an Intel Core i5-6500 3.20 GHz CPU and 8.0 GB of RAM running java1.8 (for DENSS and DBSCAN), MATLAB R2018a (for SNN-DPC, FKNN-DPC, and DPC) and Python 3.6.2 (for the other algorithms).

To make the results less sensitive to unexpected events, for each algorithm and dataset, we use the best parameters and repeated the execution 10 times, and then calculated the average execution time. Table 2 shows the average execution time required by each algorithm on each dataset. The number

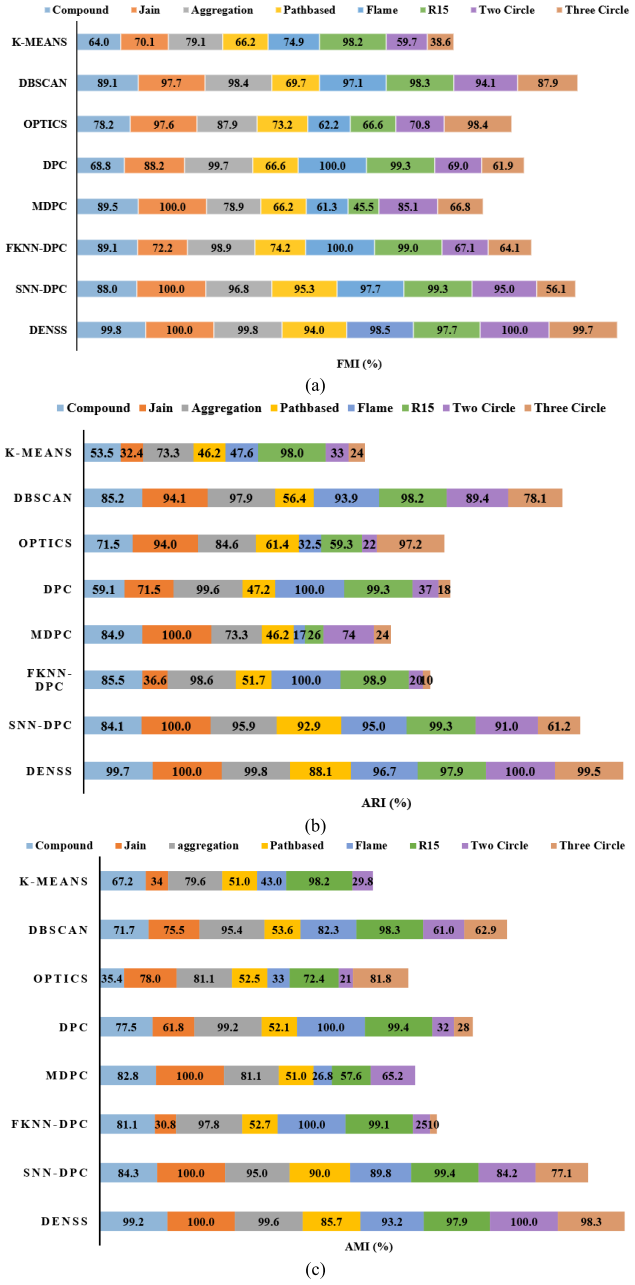


FIGURE 16. Performance of different algorithm on different datasets. (a) FMI score (b) ARI score (c) AMI score.

in parentheses below the dataset indicates the number of points in the dataset. When the data is small, DENSS, FKNN-DPC, DBSCAN and DPC consumed very little time, almost within 100 ms, and DPC required the shortest time, whereas DENSS took slightly longer than DPC. The DBSCAN algorithm does not need to judge the different cluster densities and thus consumed little time. However, this also leads to low accuracy. SNN-DPC takes more running time than the algorithms, the longest time reaches 618ms. The execution time by MDPC is less than those of the other algorithms, and the longest time takes around 10 s. On large data, DENSS takes the shortest amount of execution time, the execution time of other algorithms is even 10–100 times more than that of DENSS. Obviously, DENSS shows a better execution efficiency regardless of the size of the datasets, especially when the data is large.

TABLE 2. Execution time of clustering algorithms (Unit: millisecond).

Algorithm	DENSS	SNN-DPC	FKNN-DPC	MDPC	DPC	DBDCAN
Aggregation (788)	**65**	618	117	8330	81	66
Flame (240)	35	84	34	9856	**24**	41
Pathbased (300)	44	114	33	1954	**19**	46
R15 (600)	84	381	81	4769	**63**	86
Jain (373)	46	168	54	9692	**24**	55
Compound (399)	76	185	63	935	**26**	84
Two-circle (423)	72	206	62	2689	**30**	63
Three-Circle (7850)	**786**	12853 9	8205	1964 89	837 8	1596

In addition, the parameter settings of an excellent algorithm should have two characteristics. 1) Less prior knowledge as possible must be required from users. 2) The parameters should be insensitive to the results.

Three parameters are mainly used in this work, namely, the number of nearest k points, the DS between two points θ and the neighbour similarity between two points ε. No modifications were made for the values of k and ε during the clustering of the eight benchmark datasets, that is, k = 10 and $\varepsilon = 0.4$. The different shapes, sizes and densities of the clusters in the eight datasets are good indication of the robustness of these three parameters. The above parameters can also be adjusted according to different applications. We mainly analyse the robustness of the parameter θ below.

In DENSS, the most important parameter is the measurement threshold θ of DS, whose range interval is (0,1]. Given that the objects in the same cluster also have some differences, the value is normally [0.5, 0.83] to ensure that objects in the same cluster are similar to one another. To better verify that DENSS is insensitive to the parameters, we selected one form each of the natural shape and uneven density-pattern datasets for testing. Figure 17 shows the FMI, ARI and AMI results of the different datasets under various threshold θ values. Evidently, all the datasets are insensitive to θ. Specifically, for Compound, R15, Two-Circle and Three-Circle, when θ is within [0.6, 0.83], almost all the FMI, ARI and AMI values are greater than 0.9. Within half of the interval, the three evaluation values are greater than 0.95. For Jain and Aggregation, θ within [0.3, 0.5] and [0.73, 0.83] can obtain 100% accuracy. Given that the difference between the densities of each cluster in the two datasets, Pathbased and frame, is very small, the threshold should be more carefully set than in the other datasets. However, the threshold range for obtaining excellent results is relatively large. When the threshold θ of Pathbased is between [0.67, 0.75], the average evaluation

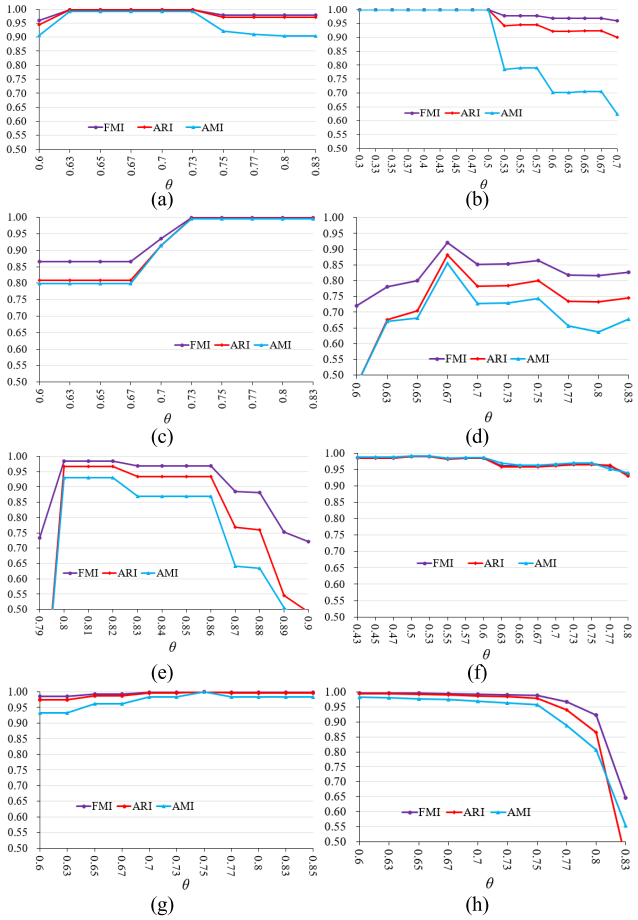


FIGURE 17. Results on different datasets with different θ. (a) Compound (b) Jain(c) Aggregation (d) Pathbased (e) Frame (f) R15 (g) Two_Circle (h) Three_Circle.

value can exceed 80%. When the threshold θ is between [0.8, 0.86] in Frame, the average evaluation value can exceed 90%.

V. CONCLUSION

This study proposes algorithm DENSS, which is a clustering algorithm based on similarity DENSS can perform clustering by judging whether the points are similar or not, thereby avoiding the problem of traditional algorithms in handling multi-density clusters. Similarity mainly includes two aspects, namely, the DS of two points and those of their neighbours. Divergence can detect the sparse degree of surrounding neighbours, whereas the similarity amongst neighbours can determine proximity. Two points are likely to belong to the same cluster when both similarities meet the requirements. The test results of eight classic synthetic datasets and one real-world dataset reveal that DENSS is a rapid, effective and self-adaptive multi-density clustering algorithm that can remarkably handle datasets of any shape, density and scale. Furthermore, DENSS is quite robust against noise.

Our work in the future will focus on the following two aspects. On the one hand, the similarity threshold might vary when clustering different datasets. Therefore, we will work on further improving the algorithm to generate the optimal threshold for different datasets smartly. On the other hand, we will try to apply DENSS to the production environment, in solving practical problems and in improving the production efficiency in relevant fields.

REFERENCES

[1] G. A. Elliott, K. C. Yang, and J. H. Anderson, "Supporting real-time computer vision workloads using OpenVX on multicore plus GPU platforms," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2015, pp. 273–284.

[2] X. X. Zhou, H. P. Zhang, G. L. Ji, and G. A. Tang, "A method to mine movement patterns between zones: A case study of subway commuters in Shanghai," *IEEE Access*, vol. 7, pp. 67795–67806, 2019.

[3] H. P. Zhang, X. X. Zhou, X. Gu, L. Zhou, G. L. Ji, and G. A. Tang, "Method for the analysis and visualization of similar flow hotspot patterns between different regional groups," *ISPRS Int. J. Geo-Inf.* vol. 7, no. 8, p. 328, 2018.

[4] X. L. Dong and T. Rekatsinas, "Data integration and machine learning: A natural synergy," *Proc. VLDB Endowment*, vol. 11, pp. 2094–2097, 2018.

[5] C. A. Escobar and R. Morales-Menendez, "Machine learning and pattern recognition techniques for information extraction to improve production control and design decisions," in *Advances in Data Mining. Applications and Theoretical Aspects*, vol. 10357. 2017, pp. 286–300.

[6] M. Ester, H. P. Kriegel, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.

[7] Y. Zhu, K. M. Ting, and M. Angelova, "A distance scaling method to improve density-based clustering," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*, 2018, pp. 389–400.

[8] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discovery Data*. vol. 10, no. 1, 2015, Art. no. 5.

[9] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *J. Amer. Statist. Assoc.*, vol. 78, no. 383, pp. 553–569, 1983.

[10] T. H. T. Nguyen and V. N. Huynh, "A K-means-like algorithm for clustering categorical data using an information theoretic-based dissimilarity measure," in *Foundations of Information and Knowledge Systems*, vol. 9616. 2016, pp. 115–130.

[11] X. Y. Chen, Y. F. Min, Y. Zhao, and P. Wang, "GMDBSCAN: Multi-density DBSCAN cluster based on grid," in *Proc. IEEE Int. Conf. e-Bus. Eng. (ICEBE)*, Oct. 2008, pp. 780–783.

[12] B. Borah and D. Bhattacharyya, "DDSC: A density differentiated spatial clustering technique," *J. Comput.*, vol. 3, no. 2, pp. 72–79, 2008.

[13] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "Density-based semi-supervised clustering," *Data Mining Knowl. Discovery*, vol. 21, pp. 345–370, Nov. 2010.

[14] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 842–847.

[15] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[16] B. Cai, G. Huang, X. Yong, H. Jing, G. L. Huang, D. Ke, and X. Zhou, "Clustering of multiple density peaks," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*, 2018, pp. 413–425.

[17] L. Ni, W. Luo, W. Zhu, and W. Liu, "Clustering by finding prominent peaks in density space," *Eng. Appl. Artif. Intell.*, vol. 85, pp. 727–739, Oct. 2019.

[18] Z. L. Yan, W. J. Luo, C. Y. Bu, and L. Ni, "Clustering spatial data by the neighbors intersection and the density difference," in *Proc. 3rd IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol. (BDCAT)*, 2016, pp. 217–226.

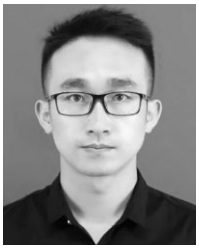
[19] X. Xu, S. Ding, M. Du, and Y. Xue, "DPCG: An efficient density peaks clustering algorithm based on grid," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 5, pp. 743–754, 2018.

[20] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Trans. Comput.*, vol. C-22, no. 11, pp. 1025–1034, Nov. 1973.

[21] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. SIAM Int. Conf. Data Mining*, 2003, pp. 47–58.

[22] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Inf. Sci.*, vol. 450, pp. 200–226, Jun. 2018.
[23] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discovery From Data*, vol. 1, no. 1, p. 4, Mar. 2007.
[24] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data," *BMC Bioinform.*, vol. 8, no. 1, p. 3, 2007.
[25] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, 2008.
[26] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1273–1280, Sep. 2002.
[27] A. K. Jain and M. H. C. Law, "Data clustering: A user's dilemma," in *Pattern Recognition And Machine Intelligence*, vol. 3776. 2005, pp. 1–10.
[28] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. C-100, no. 1, pp. 68–86, Jan. 1971.
[29] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 1998, vol. 55. [Online]. Available: http://www.ics.uci.edu/?mlearn/mlrepository.html
[30] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM Sigmod. Rec.*, vol. 28, no. 2, pp. 49–60, 1999.

XINGXING ZHOU born in 1991. He is currently pursuing the Ph.D. degree. His main research interests include spatio-temporal trajectory analysis and data mining.



HAIPING ZHANG born in 1989. He is currently pursuing the Ph.D. degree. His main research interests include spatio-temporal analysis and modeling.

GENLIN JI born in 1964. He received the Ph.D. degree. He is a professor and a Ph.D. supervisor. His main research interests include data mining, and big data analysis and its application.

GUOAN TANG born in 1961. He received the Ph.D. degree. He is a professor and a Ph.D. supervisor. His main research interest includes digital terrain analysis.