information

Article

Fast Component Density Clustering in Spatial Databases: A Novel Algorithm

Bilal Bataineh

Information Systems Department, College of Computers and Information Systems, Umm Al-Qura University, Makkah P.O. Box 715, Saudi Arabia; bmbataineh@uqu.edu.sa

Abstract: Clustering analysis is a significant technique in various fields, including unsupervised machine learning, data mining, pattern recognition, and image analysis. Many clustering algorithms are currently used, but almost all of them encounter various challenges, such as low accuracy, required number of clusters, slow processing, inability to produce non-spherical shaped clusters, and unstable performance with respect to data characteristics and size. In this research, a novel clustering algorithm called the fast component density clustering in spatial databases (FCDCSD) is proposed by utilizing a density-based clustering technique to address the aforementioned existing challenges. First, from the smallest to the largest point in the spatial field, each point is labeled with a temporary value, and the adjacent values in one component are stored in a set. Then, all sets with shared values are merged and resolved to obtain a single value that is representative of the merged sets. These values represent final cluster values; that is, the temporary equivalents in the dataset are replaced to generate the final clusters. If some noise appears, then a post-process is performed, and values are assigned to the nearest cluster based on a set of rules. Various synthetic datasets were used in the experiments to evaluate the efficiency of the proposed method. Results indicate that FCDCSD is generally superior to affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, and density-based spatial clustering of applications with noise, ordering points for identifying clustering structures, and Gaussian mixture clustering methods.

Keywords: clustering; data mining; density-based clustering; unsupervised learning

check for updates

Citation: Bataineh, B. Fast Component Density Clustering in Spatial Databases: A Novel Algorithm. *Information* **2022**, *13*, 477. https://doi.org/10.3390/info13100477

Academic Editor: Annalisa Appice

Received: 30 August 2022
Accepted: 23 September 2022
Published: 2 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Clustering analysis is a technique of grouping unlabeled data into different clusters based on their similarities [1–4]. This approach is essential in the areas of unsupervised machine learning, data mining, pattern recognition, image analysis, and much more [4–7]. The unlabeled dataset is essentially divided into labeled groups regardless of the presence of similar patterns [8,9]. These labeled clusters can be used simply to process and analyze large and complex datasets in many applications, such as healthcare [10,11], renewable energy [12,13], image segmentation [14–16], data analysis [1,9], social network analysis [17,18], security [19,20], finance and business [21,22], and much more. Many studies on clustering algorithms have been conducted. The most well-known clustering types are partitioning clustering, distribution model-based clustering, hierarchical clustering, fuzzy clustering, and density-based clustering [23,24].

The abovementioned types have been effectively utilized in various fields and achieved good precision under most dataset scenarios. However, they suffer from many limitations, including the need for a prior specification of the number of clusters [25]. The performance of some algorithms with huge sizes, complex shapes, and outlier data is also slow [26]. Furthermore, many algorithms produce clusters only in spherical shapes, which indicates limited performance when wide types of datasets are handled [9,27]. These techniques cannot provide meaningful cluster descriptors because the data always contain numerous

inconsistencies, outliers, and noise; in other words, reliable performance across different data types cannot be established [3,6,28,29].

Density-based clustering is a technique for overcoming most clustering issues. It is based on the idea that the density within each cluster of adjacent points is higher than the density outside of the cluster. This technique consists of some algorithms that are applicable to various databases [30–32]. Density-based spatial clustering of applications with noise (DBSCAN) and the ordering points for identifying the clustering structure (OPTICS) are the most well-known algorithms. They detect clusters of varying shapes, sizes, and densities in spatial datasets with outliers and noise. Moreover, density-based clustering does not require information about the number of clusters [28,31,32]. However, this technique is imperfect. For example, DBSCAN cannot properly cluster datasets of varying densities, and using it is extremely costly under certain worst-case dataset scenarios. In addition, some border points from one cluster may belong to other clusters [8,31]. Meanwhile, the OPTIC algorithm produces only a cluster order, and it is inapplicable for high-dimensional data and outrageously costly [30,31].

The aim of this study is to propose a novel FCDCSD algorithm to overcome the majority of clustering challenges mentioned above and achieve much higher performance. The proposed FCDCSD algorithm employs density-based clustering and consists of two stages. First, all points are labeled with temporary values, and then adjacent label values are stored in sets. Then, all sets with shared values are merged, and the representative value is obtained and replaced with their temporary equivalents to identify the clusters. Additionally, a post-process is executed to solve the noise problem.

Experiments on several datasets were conducted to evaluate the efficacy of the proposed FCDCSD. The results indicate that FCDCSD is generally superior to the affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, and Gaussian mixture clustering methods. Moreover, FCDCSD can improve accuracy across multiple datasets, is simple to implement, has fast computer performance, and is unaffected by data properties. It is efficient in handling noise and outliers, and it can verify density datasets and create clusters with complex shapes and sizes.

The contributions of this paper are a proposed method that does not need to predetermine the number of clusters and implements large sizes, complex shapes, and external datasets. Moreover, the proposed method is stable and produces clusters of any shape, and deals with outliers, varying densities, and noise. In other words, it has reliable performance across different data types, and its principle is applicable to high-dimensional data. In addition to the above, this paper presents the most well-known methods of clustering, explains their theoretical techniques, and presents the complexity-time and visual comparison of results between several well-known, state-of-the-art methods with different types of data sets.

The rest of this paper is organized as follows. The next section presents the state-of-the-art clustering algorithms. Section 2 describes the proposed method in detail. Section 3 discusses the experiments and results. Finally, in Section 4, the conclusion is included.

State-of-the-Art Methods

Existing clustering algorithms can be categorized into partitioning clustering, distribution model-based clustering, hierarchical clustering, fuzzy clustering, and density-based clustering [33,34]. Partitioning clustering, a popular technique, divides data points into k parts (clusters) based on a particular function. K-means [35] and mean-shift clustering [36] are common partitioning clustering algorithms. They can be directly applied to clusters and are efficient and effective in handling dense datasets. However, the number of clusters must be specified manually, and initial values, noise, and outliers are difficult to process [37–39].

Distribution model-based clustering can be directly applied to clustering data points by using statistical distribution models, such as Gaussian distribution. The likelihood in which objects share the same probability distribution is used to define the clusters. Then, each cluster is represented by a unique group [40,41]. Distribution-based clustering primarily

uses the Gaussian mixture model [25]. This technique does not require a specification of prior values, is compatible with real-time data, and its metrics are simple to design and implement; however, it is complicated, computationally costly, and cannot be scaled to large datasets.

Hierarchical clustering detects how data points are similar and dissimilar to each other, and it iteratively splits or merges data points to produce a tree structure, in which the root node represents the entire set of data, and the branches form the clusters [42–44]. The algorithm for agglomerative hierarchical clustering is a common algorithm for this technique [45]. The number of clusters does not need to be defined manually, and the hierarchical clustering is easy to implement, adaptable, and scalable for scenarios involving point linkages. Agglomerative hierarchical clustering provides object order and may highlight data patterns [23,42,45]. However, after splitting or merging, modifications cannot be further performed, and grouping instructions lack interpretability.

In the fuzzy clustering technique, each data point may belong to many clusters [46]. Fuzzy c-means clustering, fuzzy k-means, and fuzzy c-ordered-means are a few examples [46,47]. For strongly correlated and overlapping datasets, fuzzy clustering is utilized [47–49]. However, the number of clusters must be specified manually; furthermore, the method is inefficient and cannot be used for large datasets, and it is ineffective for noisy and outlier data.

Density-based clustering relies on the adjacent density of points. It detects dense data space by considering the distance between neighboring points and distinguishing regions with relatively low densities [28,31,50]. Density-based clustering does not require a fixed number of clusters, and it can recognize clusters of different shapes and sizes and subsequently handle noise and outliers [31]. The most notable examples are DBSCAN [32] and the OPTICS algorithm [30].

DBSCAN, the most widely used density-based algorithm [31,39], continuously selects a point from a dataset. If a minimum number of points fall within the radius of a specified distance from the selected point, then these points are grouped. The step is repeated until all points are scanned [32]. The DBSCAN rules entail a fast operation, automatically compute the number of clusters, find clusters of any shape, and they are robust to noise and outliers [31,32,39]. However, DBSCAN cannot cluster datasets with highly diverse densities. In addition, the algorithm is not totally deterministic because it begins with a random point, in which some border points from one cluster may belong to other clusters. Additionally, DBSCAN may be time-consuming for some datasets under the worst scenario.

OPTICS was proposed to overcome one of DBSCAN's challenges, which is the inability to detect clusters among data with varying densities [30]. OPTICS utilizes the distance between neighboring points to build a reachability plot, which is used to distinguish clusters of varying densities from noise. For each point, OPTICS records the core distance and an acceptable reachability distance. OPTICS stores the output ordering in a list based on the minimum reachability distance from the nearest core point, which is the density that must be tracked for a cluster [30,39]. Furthermore, it can cluster datasets with varying densities and outliers. However, OPTICS generates only a cluster order, is incapable of processing high-dimensional data, is a slow algorithm, and is extremely computationally expensive, particularly when the search distance is huge [31,39].

A summary of the current state-of-the-art approaches demonstrates that the distribution model-based clustering, hierarchical clustering, and fuzzy clustering techniques suffer from several issues, with some of them requiring input parameters that are difficult to identify, hence the remarkable effect on the clustering outcomes. Meanwhile, all partition, distribution model-based, hierarchical, and fuzzy clustering methods or algorithms are affected by noise and outliers; they are incapable of clustering complex data structures, cannot be scaled, and are costly on large datasets. Density-based clustering algorithms perform better with noisy and outlier data; they can detect clusters with complex shapes and sizes and automatically compute the number of clusters. However, DBSCAN is unsuitable for datasets with varying densities. Furthermore, OPTICS produces only the cluster order,

fails to handle multidimensional datasets, and has a high computational cost due to the vast search distance.

2. Materials and Methods

As shown in Figure 1, when points of any dataset are represented in spatial space, different clusters and noises may be distinguished based on the densities and distances between points. Each group of points can be detected as an independent cluster if it has a density that is much higher than that of the surrounding area. Thus, if any piece of density overlaps with other close pieces with a matching density, then these two pieces belong to the same cluster. This principle is applied to the proposed method.

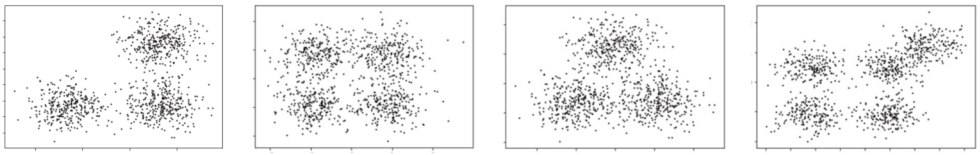


Figure 1. Dataset points in 2D spatial representation.

Particularly, in the proposed method, D is a dataset whose points are represented in a spatial plot (i.e., a two-dimensional space in this study), and P denotes a point of dataset D. The points are placed in a space plot from bottom to top and left to right from smallest to largest. The smallest point in D is plotted at the lower part of the space, whereas the largest point is positioned in the highest position. All remaining points are placed between the smallest and largest points in the spatial plot.

A set of adjacent points is considered a component. In this study, the component is a circular area that groups a set of adjacent points. All points in the component are implemented as a single unit. This component scans all data points in an orderly motion, from bottom to top and from left to right, and the points in the dataset are included in a component at least one time.

In creating and using the component in the proposed algorithm, two parameters, namely, component size and component density, are considered.

- Component size (CS) is the radius size of the component. The component is a circular region formed by a radius and a core point as a center. The radius must fit the nature of distances between data points. Large distances require a large radius value. However, a minimum value must be determined, and it should be sufficient to create a component for covering several points in each process.
- Minimum density (MD) is the threshold density value used to classify each created component as a cluster or noise. It refers to the minimum number of points that must be in the component. This parameter must be suitable for the size of the dataset, and large data require a large value.

Choosing incorrect values for the above parameters can cause problems in which many clusters will be combined, or the points are considered outliers. Therefore, it is not a good idea to estimate these parameters at random.

In this work, CS is the radius of the circular shape component, i.e., the maximum distance between two points considered neighbors, which are widely used in image processing and data analysis. Several techniques are used to estimate the optimal value for this parameter, but the k-distance graph is the most common. It is based on plotting the distances between each point and its nearest neighbor in order from largest to smallest value. Good CS values are where this plot shows an elbow. In general, small values of CS are preferred, and, as a rule, only a small part of the points should be within this distance from each other. This parameter is the main factor in the accuracy of the clustering process. On the other hand, the MD value is not necessary for the accuracy of the performance. It improves processing time and boosts performance with noisy data sets. The default value is MD = 1 for the best processing time. However, a slightly larger value is preferred for

noisy datasets and results in more significant clusters. In addition, the following set of variables are defined:

- Core point (P_core) is the selected center point for creating the current component. This point must be an "unlabeled point."
- Unlabeled points (PU) are points that do not yet have label values.
- Labeled points (PL) are points within the component, and it is given a label value by a previous process.
- Component points (PC) are the data points within the component.
- Component density (CD) represents the number of data points in the component.
- Temporary label (TL), represented by values, is assigned to the points. The initial value is 1, and it increases by 1 after each use.
- Label list (LL) stores the adjacent TLs. The stored values are used to calculate the final clusters.

2.1. Temporary Labeling Stage

In this stage, each point in the dataset is labeled by a TL value, and all associated label values in one component are stored for the next stage.

In an orderly manner in space, i.e., from bottom to top and from left to right, the smallest point is selected. If the point has a label value, then it is ignored, and the process moves to test the next point. If the point does not have a label value, then it is considered a P_core, and a new component is created on it. The details are shown in Figure 2b. Then, the CD is calculated. If the CD is higher than the MD, then all points in the component are considered to belong to the same cluster.

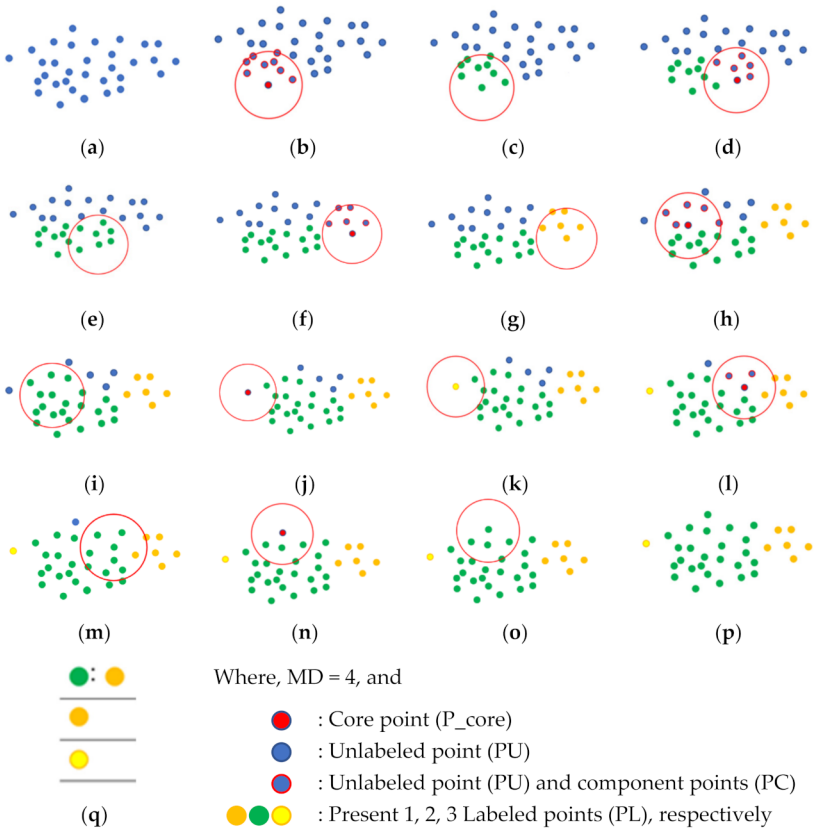


Figure 2. Example of the temporary labeling stage process: (**a**) original unlabeled data points; (**b**–**o**) seven times of the core point selection and labeling process; (**p**) output data labeled with temporary values; (**q**) resultant LL.

Therefore, the TL value is assigned to all Pus based on one of two cases. First, if all points in the new component are previously non-labeled Pus, then they will be labeled according to the TL value (Figure 2b,f), and the temporary value is increased by 1 (TL = TL + 1). However, if a PL exists in the component (Figure 2d,h,l,n), then the PU without a label value is labeled using the smallest label value (min(PL)) in the component (Figure 2e,I,m,o). The different label values in the component are stored in the LL as associated labels for subsequent use when finding clusters (Figure 2q).

When the CD is smaller than the MD, then the component has noise points (Figure 2j). Thus, only the P_core is labeled by the TL (Figure 2k), and the TL value is incremented by 1 (TL = TL + 1).

When previous rules have been applied to the selected point, the next smallest point is selected, and the rules are repeated. The previous processes are repeated until all points acquire a TL value, and the LL is constructed. Figure 2 shows a sample of the temporary labeling stage processes. Algorithm 1 presents the first stage rules.

Algorithm 1: Temporary labeling stage.
Input: Component size (CS), minimum density (MD), TL = 1
Output: temerity labeled points, labels list (LL)

```
IF P is PU:
    P_Core ← P # Create new component is over it
      IF CD > MD:
        IF PL in PC # if any labeled point in the component
          PC ← min(PL) # label all points by the minimum label
          LL(TL) ← ALL (PL) # store all labels in the list
      ELSE:
          ALL PC ← TL
          LL(TL) ← TL # store the label in the list
          TL = TL + 1
       ELSE:
           P_Core ← TL
           LL(TL) ← TL # store the label in the list
           TL ← TL + 1
```

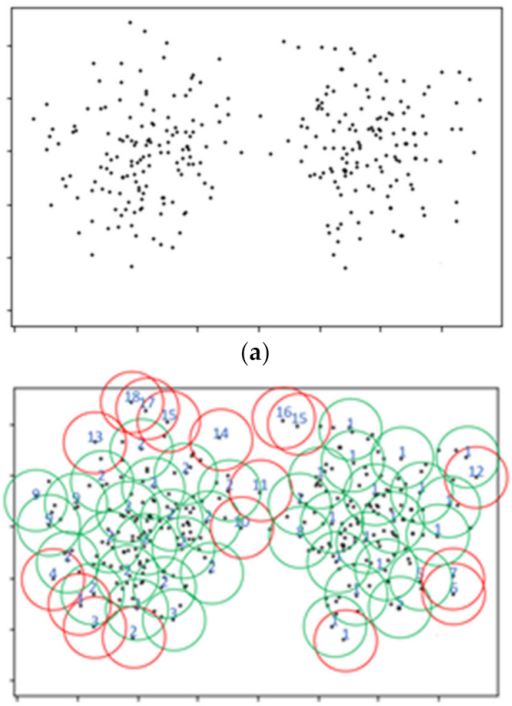
A practical example of handling a dataset consisting of 300 points and two clusters is presented in Figure 3. Figure 3a represents the unlabeled datasets. Figure 3b shows the dataset after applying the temporary labeling stage. The green parts represent components whose CDs are higher than the MD, whereas the red parts represent components whose CDs are lower than the MD. The required LL is presented in Figure 3c.

2.2. Clustering Stage

The aim of this stage is to find the actual clusters by replacing all adjacent TL values with a single representative value only. After the first stage, all points in the dataset have a temporary value and are then stored in the LL. The LL is a list storing the adjacent TLs in the dataset structure. At this stage, the LL is resolved to find the lowest label in the different sets of associated label values. Subsequently, this lowest value is used as a representative value to replace all adjacent TLs. A data structuring technique is applied to achieve the Union and Find operations.

In general, each individual set of adjacent temporary values in the LL can be assumed as a tree data structure. Each node of the tree is a TL, and the root is the minimum value of labels. The Union–Find data structure keeps track of the elements. Then, these elements are divided into several disjoint subsets by performing two operations (Union and Find). The union process merges several sets according to their common values into a single set, from which the results are represented as a tree. The Find process returns the root of the tree, and all labels in the result set become associated with the same root (i.e., the smallest value

in the set). Many Union–Find data structure methods have been previously proposed, and any of them can be used in this work.



(**a**) (**b**)

LT	Adjacent LT in the same component
1	5, 9, 13
2	3, 8, 11,12
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

(**c**)

Figure 3. Example of the temporary labeling stage: (**a**) original datasets, (**b**) processing of data points; (**c**) resultant LL.

As the LL in this work is usually simple to operate, a simple process is also proposed to replace each TL with its representative value. Each cell of the obtained LL contains either a single temporary value that is not adjacent to other values or a set of two or more values. Thus, the sets with common values are merged into one set, and the minimum value of each result set is the root, which is the representative value of all temporary values in the merged sets. In the case of some complex shapes of data, the sequential levels of connectivity between sets should be the focus. Figure 4 shows an example of these processes and outputs based on synthetic LL. Each TL value of the dataset points is replaced by its equivalent representative value. Consequently, all points can be labeled by their cluster values.

TL	Adjacent labels
1	1, 4
2	2, 5, 6
3	3, 8
4	4, 5, 7
5	5
6	6
7	7
8	8

(**a**)

TL	Merged sets
1	1, 2, 4, 5, 6, 7
2	2
3	3, 8
4	4
5	5
6	6
7	7
8	8

(**b**)

TL	Representative value
1	1
2	1
3	3
4	1
5	1
6	1
7	1
8	3

(**c**)

Figure 4. Example of the resolving step: (**a**) synthetic resultant LL, (**b**) LL after sets merged; (**c**) the temporary values and their representative value.

Finally, noise points, which are single points with a unique value, are handled. The distance between each noise point and the closest clustered point is determined. If the value is less than the radius of the initial CS, then this point belongs to the current cluster, and its value is replaced by the cluster label; otherwise, its label value is maintained. Algorithm 2 presents the steps of this second stage.

Algorithm 2: Clustering stage.
Input: Labels list (LL), temerity labeled data points
Outputs: List of representative values, clustered data

```
Sort all sets of adjacent label values in if LL incrementally
    For I in 1 to TL: # Union
        For v in LL[i]:
            IF v ! = i and length (LL[i]) > 1:
                LL[ i ] ← LL[ i ] + LL[ v ]
                LL[ v ] ← v
FOR i in 1 to TL: #Find
    Representative [ I ] ← min(LL[ I ])
FOR each TL in Data: # Replace TL by representative value
    P_Cluster ← Representative [TL] # The final cluster value for each point
FOR each Noise:
    IF distance (Noise, nearest cluster) < radius:
        Noise ← nearest cluster
```

Thus, after the second stage, all points in a dataset can be clustered. In summary, in the first stage, the TLs are assigned and stored in the LL. Then, in the second stage, the LL is resolved, and the TL values are replaced by their representative values. Finally, noise points are handled.

3. Results

Experiments were conducted, and their outcomes were analyzed to determine the effectiveness of the proposed clustering method. Aimed at evaluating the clustering performance, a comparative analysis was performed using several well-known clustering algorithms, namely, affinity propagation, agglomerative hierarchical clustering, k-means, mean-shift, spectral, DBSCAN, OPTICS, and Gaussian mixtures methods. Six synthetic sample databases were employed to compare the performance of each method.

The results of each method were matched with the ground truth labels of the synthetic databases and then evaluated using performance metric measurements, such as time complexity, Rand index (*RI*), adjusted Rand index (*ARI*), homogeneity, completeness, V-measure (normalized mutual information (*NMI*)), and Fowlkes–Mallow's score. All experiments were conducted on a computer running Windows 10 with an Intel(R) Core(TM) i5-9600KF processor at 3.70 GHz and 32 GB of RAM. No GPU was used. The clustering methods and measurements were imported using the Scikit-Learn machine learning library. Each program was executed on Spyder Python 3.

3.1. Dataset Selection

Normally, class labels are not available in clustering datasets, which is the primary reason for relying on unsupervised learning. Given the lack of class labels, the performance of clustering algorithms cannot be accurately evaluated. In the evaluation of the clustering outcomes, datasets containing ground truth class labels must be used. By comparing the results of each clustering method to the ground truth class labels, their performance can be evaluated.

Six synthetic datasets with different shapes were used in the experiments to ensure an accurate evaluation: noisy circles, noisy moons, blobs, anisotropically distributed data, blobs with varying variances, and overlapping distributed datasets. These synthetic

datasets were previously used in published research [4,25,30,39,51,52]. Each dataset contained 1500 data points. The noisy circles, noisy moons, blobs, and overlapping distributed datasets were composed of two clusters each, whereas the blobs, anisotropically distributed data, and blobs with varying variances datasets comprised three clusters each. Figure 5 shows the distribution values for each dataset.

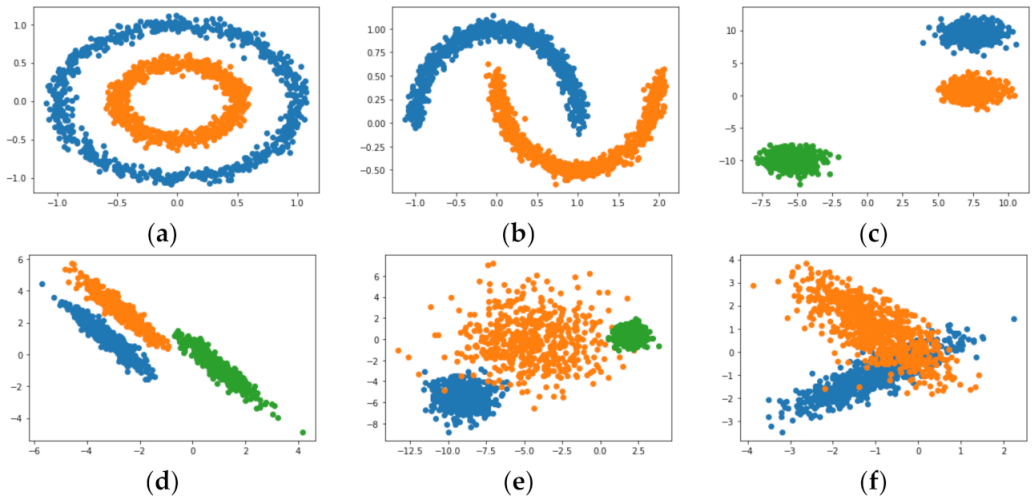


Figure 5. Used datasets: (**a**) noisy circles, (**b**) noisy moons, (**c**) blobs, (**d**) anisotropically distributed data, (**e**) blobs with varied variances, and (**f**) overlapped distributed datasets.

3.2. Estimation of Parameters

To display the significance of the CS and MD parameters and the ability of the k-distance graph to estimate the value of CS, the following experiment was introduced. First, a small data set is used to illustrate the set points; a blobs data set with 500 points was used. Figure 6 shows the resulting graph after applying the k-distance graph based on the default value of MD = 1 where k = MD. The optimal value of CS is the value of 0.7. The estimated optimal CS value is evaluated, and Figure 7 shows the clustering results for the proposed method using the optimal CS value compared to some selected CS values. The results show that the dataset correctly clustered with the optimal CS value (Figure 7a), but the clustering failed with the set CS values (0.6 and 0.8), Figure 7b,c, respectively.

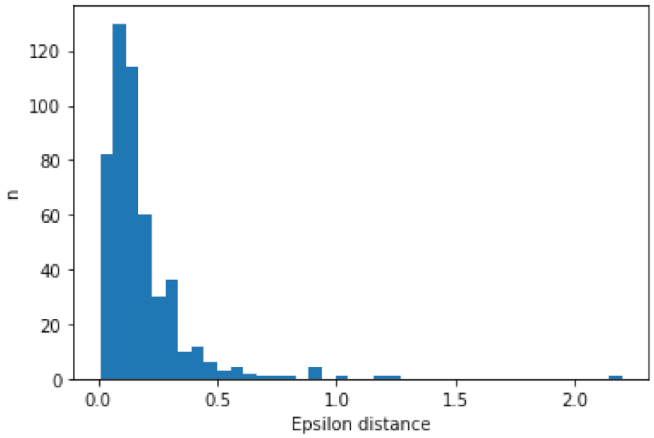


Figure 6. The k-distance graph of k = 1 distance to the nearest neighbor (elbow in value 0.7).

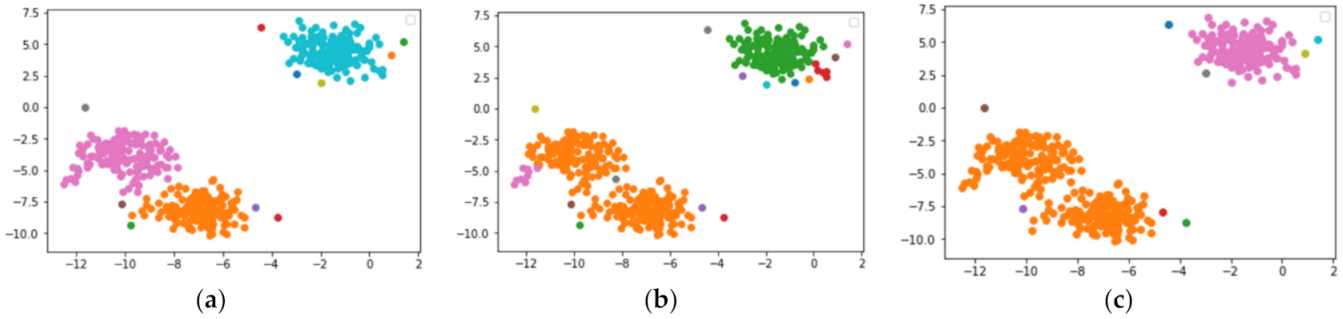


Figure 7. Visual result of proposed method with DM = 1 and (**a**) CS = 0.7, (**b**) CS = 0.6, and (**c**) CS = 0.8.

Figure 8 shows the clustering performance for the optimal CS value with different values of MD. In addition to DM = 1 displayed in Figure 7a, the results for the values MD = 2, MD = 3 and MD = 4 are shown in Figure 8a–c. The results show that values from DM = 1 to DM = 3 give good clustering performance, while clustering starts to fail from value MD = 4. Obviously, the default value is good (Figure 7a), but the result of the following value MD = 2 is better with outlier points (Figure 8a). The problem of outliers was solved with MD = 2 and MD = 3, but the clusters started merging from MD = 4.

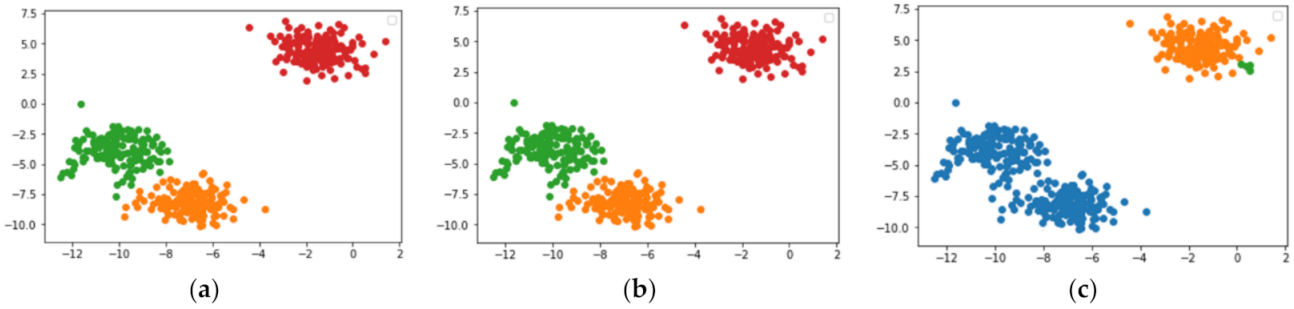


Figure 8. Visual result of proposed method with CS = 0.7 and (**a**) DM = 2, (**b**) DM = 3, and (**c**) DM = 4.

3.3. Evaluation Measurement Methods

Many metrics can be used to evaluate the performance of clustering algorithms. Here, the evaluation of clustering performance should not be directly correlated to the label value errors. Rather than adopting the absolute values of clusters, the clustered values as a proxy were used to meet the requirements of the ground truth dataset. In this manner, the limits between similar data within a set of classes or similar points belonging to the same class that differs from similar points belonging to other classes can be determined.

As explained previously, the accuracy measurement used in many studies is inefficient. In other cases, even though the performance of some clustering algorithms is excellent, their accuracy rates are low because the adopted measurements depend on the absolute label values of clusters, and the output label values do not match the values in the ground truth dataset. In this study, aimed at circumventing this issue, the approaches of time complexity, *RI*, *ARI*, homogeneity, completeness, V-measure (normalized mutual information (*NMI*)), and Fowlkes–Mallow's score were adopted. The metrics can be described as follows.

Time complexity is used to offer a standard measurement of the methods regardless of the external variables. The processing time of any clustering algorithm depends on several variables, such as the number of clusters, size of the dataset, type of hardware and software, and architecture of the program. Time complexity, which is employed to avoid the impacts of various variables, is typically measured by computing the number of elementary steps required to execute an algorithm, assuming that each step takes the same amount of time. The constant runtime is represented by O(1), O(log n), O(sqrt n), O(n), O(n log n), O(n2), O(n3), O(2n), O(10n), and O(n!) in the order from best to worst runtime.

RI measures the clustering similarity of two datasets. It counts all pairings assigned to the same or different clusters in the real clustering and disregards permutations. The scores of perfectly agreeable labels move close to 1, whereas those of weak agreeable labels move close to 0. *RI* is essentially equivalent to the percentage of decisions calculated using the following equation:

$$Percentage = \frac{TP + TN}{TP + FP + FN + TN}$$

where *TP*, *TN*, *FP*, and *FN* are the number of true positives, true negatives, false positives, and false negatives, respectively.

$$RI = \frac{a + b}{a + b + c + d} = \frac{a + b}{\binom{n}{2}}$$

In data clustering, *C* is the ground truth class assignment, *K* is the clustering, a is the number of pairs of elements that are in the same set (i.e., in both *C* and *K*), and b is the number of pairs of elements that are in different sets in *C* and in different sets in *K*.

- *a*, the number of pairs of elements in S that are in the same set in *C* and in the same set in *K*.
- *b*, the number of pairs of elements in S that are in different sets in *C* and in different sets in *K*.
- *c*, the number of pairs of elements in S that are in the same set in *C* and in different sets in *K*.
- *d*, the number of pairs of elements in S that are in different sets in *C* and in the same set in *K*.

ARI can be described from the perspective of *RI*. For instance, if the number of clusters and the number of samples have the same order of magnitude, then *RI* cannot ensure random label assignments to ensure a value close to 0. In view of counteracting this effect, the predicted *RI* $E[RI]$ of random labeling is discounted using the following definition of the modified *RI*.

$$ARI = \frac{RI - E[RE]}{MAX[RI] - E[RE]}$$

Homogeneity assesses the amount of only a sample of a single class by which the cluster contains. **Completeness** assesses how much similar samples are assigned together to the same cluster. The harmonic average between homogeneity and completeness is given by the **V-measure**, which is the *NMI*.

Information (*NMI*)

$$\text{Homogeneity} = 1 - \frac{H(C \mid K)}{H(C)}$$

$$H(C \mid K) = -\sum_{c,k} \frac{n_{c,k}}{N} \, log\left(\frac{n_{c,k}}{n_k}\right)$$

$$\text{Completeness } 1 - \frac{H(K \mid C)}{H(K)}$$

$$NMI = 2 \, \frac{H * C}{H + C}$$

where *H* represents homogeneity, *C* represents completeness, and $H(C \mid K)$ is the ratio between the number of samples labeled as *c* and the total number of samples in cluster *k*.

The **Fowlkes–Mallow's index** is defined as the geometric mean of pairwise accuracy and recall.

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}$$

In the previous measurements, 1 denotes the best performance, whereas a decreasing value implies moving towards poor performance.

3.4. Clustering Performance

Each clustering algorithm's time complexity (Big O) is listed in Table 1. The constants and scaling variables, such as the number of clusters or the size of the database, are disregarded because only the asymptote is of interest. However, the number of iterations is maintained because it is crucial to determine the processing time in certain cases. The results indicate that the FCDCSD is a relatively fast method, with time complexity of O(N), because its stages are ordered regardless of the size and shape of the data or the number of clusters. It does not require an iterative process and is stable while still being (O(N)) under the worst-case scenario. The next best algorithms are k-means clustering, OPTICS, and DBSCAN. Agglomerative hierarchical clustering, Ward's hierarchical clustering, spectral clustering, and Gaussian mixtures were performed poorly according to $O(N^3)$.

Table 1. The time complexity (Big O) for each clustering algorithm.

Method	Complexity Time
Affinity Propagation	O(N2I)
Agglomerative Hierarchical Clustering	$O(N^3)$
K-means clustering	O(IN)
Mean-shift	O(IN2)
Spectral clustering	O(N3)
Ward hierarchical clustering	O(N3)
DBSCAN	O(N2), could be O(N log N)
OPTICS	O(N log N)
Gaussian mixtures	O(N3)
Proposed	O(N)

N = number of points, I = number of iterations.

The trends are shown in Tables 2–7 and Figure 9, while Figure 10 presents a visual representation of each method and all datasets to sufficiently illustrate the results of each method. The results indicate that the performance of each method varies depending on the dataset type. Some methods perform poorly in all cases, whereas other algorithms perform differently. Overall, the FCDCSD provides the best and most reliable performance regardless of the dataset type.

Table 2. The *RI* of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods on noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

	Noisy Circles	Noisy Moons	Blobs	Aniso	Varied	Overlapped
Affinity Propagation	0.53	0.55	0.76	0.72	0.73	0.52
Agglo. Hierarchical	0.50	0.73	1.00	0.80	0.99	0.72
K-means	0.50	0.62	1.00	0.82	0.92	0.73
Mean-shift	0.50	0.71	1.00	0.76	0.93	0.74
Spectral	0.50	0.65	1.00	0.89	0.97	0.69
DBSCAN	1.00	1.00	0.98	0.99	0.96	0.62
OPTICS	0.51	0.50	0.58	0.58	0.56	0.50
Gaussian mixtures	0.50	0.75	1.00	1.00	0.99	0.82
Proposed	1.00	1.00	1.00	1.00	0.97	0.69

Table 3. The *ARI* of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods on noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

	Noisy Circles	**Noisy Moons**	**Blobs**	**Aniso**	**Varied**	**Overlapped**
Affinity Propagation	0.07	0.10	0.33	0.21	0.24	0.05
Agglo. Hierarchical	0.01	0.47	1.00	0.57	0.97	0.46
K-means	0.00	0.24	1.00	0.61	0.83	0.46
Mean-shift	0.00	0.42	1.00	0.54	0.85	0.49
Spectral	0.00	0.29	1.00	0.74	0.93	0.39
DBSCAN	1.00	1.00	0.96	0.98	0.91	0.24
OPTICS	0.01	0.01	0.02	0.01	0.01	0.00
Gaussian mixtures	0.00	0.50	1.00	1.00	0.97	0.64
Proposed	1.00	1.00	1.00	1.00	0.94	0.39

Table 4. The homogeneity of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods on noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

	Noisy Circles	**Noisy Moons**	**Blobs**	**Aniso**	**Varied**	**Overlapped**
Affinity Propagation	1.00	1.00	1.00	0.99	0.95	0.69
Agglo. Hierarchical	0.00	0.48	1.00	0.63	0.95	0.43
K-means	0.00	0.18	1.00	0.63	0.81	0.43
Mean-shift	0.01	0.36	1.00	0.52	0.83	0.45
Spectral	0.00	0.22	1.00	0.74	0.91	0.41
DBSCAN	1.00	1.00	0.98	0.99	0.94	0.46
OPTICS	0.58	0.47	0.46	0.48	0.43	0.35
Gaussian mixtures	0.00	0.40	1.00	1.00	0.94	0.55
Proposed	1.00	1.00	1.00	1.00	0.91	0.41

Table 5. The completeness of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods on noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

	Noisy Circles	**Noisy Moons**	**Blobs**	**Aniso**	**Varied**	**Overlapped**
Affinity Propagation	0.20	0.23	0.45	0.36	0.35	0.14
Agglo. Hierarchical	0.00	0.51	1.00	0.68	0.95	0.45
K-means	0.00	0.18	1.00	0.63	0.82	0.45
Mean-shift	0.00	0.37	1.00	0.89	0.84	0.48
Spectral	0.00	0.22	1.00	0.74	0.91	0.46
DBSCAN	1.00	1.00	0.91	0.93	0.85	0.17
OPTICS	0.12	0.12	0.19	0.19	0.18	0.09
Gaussian mixtures	0.00	0.40	1.00	1.00	0.94	0.55
Proposed	1.00	1.00	1.00	1.00	0.91	0.46

Table 6. The V_measure of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

	Noisy Circles	**Noisy Moons**	**Blobs**	**Aniso**	**Varied**	**Overlapped**
Affinity Propagation	0.34	0.37	0.62	0.53	0.52	0.23
Agglo. Hierarchical	0.00	0.49	1.00	0.65	0.95	0.44
K-means	0.00	0.18	1.00	0.63	0.81	0.44
Mean-shift	0.00	0.36	1.00	0.65	0.83	0.47
Spectral	0.00	0.22	1.00	0.74	0.91	0.44
DBSCAN	1.00	1.00	0.94	0.96	0.89	0.25
OPTICS	0.20	0.19	0.27	0.27	0.26	0.14
Gaussian mixtures	0.00	0.40	1.00	1.00	0.94	0.55
Proposed	1.00	1.00	1.00	1.00	0.91	0.43

Table 7. The Fowlkes–Mallows of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods on noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

	Noisy Circles	Noisy Moons	Blobs	Aniso	Varied	Overlapped
Affinity Propagation	0.26	0.31	0.52	0.40	0.43	0.22
Agglo. Hierarchical	0.51	0.75	1.00	0.73	0.98	0.74
K-means	0.50	0.62	1.00	0.74	0.88	0.74
Mean-shift	0.36	0.72	1.00	0.76	0.90	0.75
Spectral	0.50	0.65	1.00	0.83	0.96	0.72
DBSCAN	1.00	1.00	0.98	0.98	0.94	0.55
OPTICS	0.31	0.38	0.32	0.31	0.34	0.39
Gaussian mixtures	0.50	0.75	1.00	1.00	0.98	0.82
Proposed	1.00	1.00	1.00	1.00	0.96	0.71

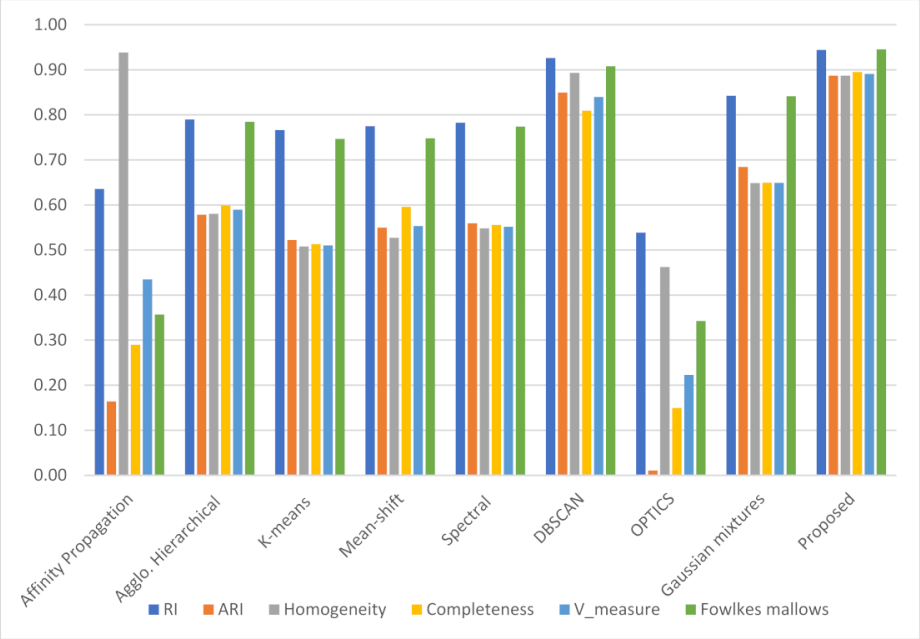


Figure 9. The average of *RI*, *ARI*, homogeneity, completeness, and Fowlkes–Mallow's score. Note: measurements of affinity propagation, agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, Gaussian mixtures, and the proposed clustering methods.

In particular, the FCDCSD and DBSCAN attained the best performance on the noisy circles and noisy moons in all metrics, whereas the remaining methods performed poorly on these two datasets. On the blobs datasets, the proposed method performed comparably with the agglomerative hierarchical, k-means, mean-shift, spectral, and Gaussian mixture methods across all metrics. On the anisotropic dataset, only the FCDCSD and the Gaussian mixture method performed well on all metrics. Notably, the DBSCAN failed to achieve the maximum performance on the blobs and anisotropic datasets despite its good performance on average.

On the varying variance and overlapping datasets, none of the methods achieved the same level of performance compared with the previously presented datasets due to the high levels of noise and the significant overlaps. In particular, on the varying variance dataset, the order of high performance (with minor variations) is agglomerative hierarchical, Gaussian mixture, the proposed method, and the spectral algorithm. The overlapping datasets particularly have overlapping data labels, and all methods presented an average performance when utilizing them. The Gaussian mixture method attained the best performance in this case.

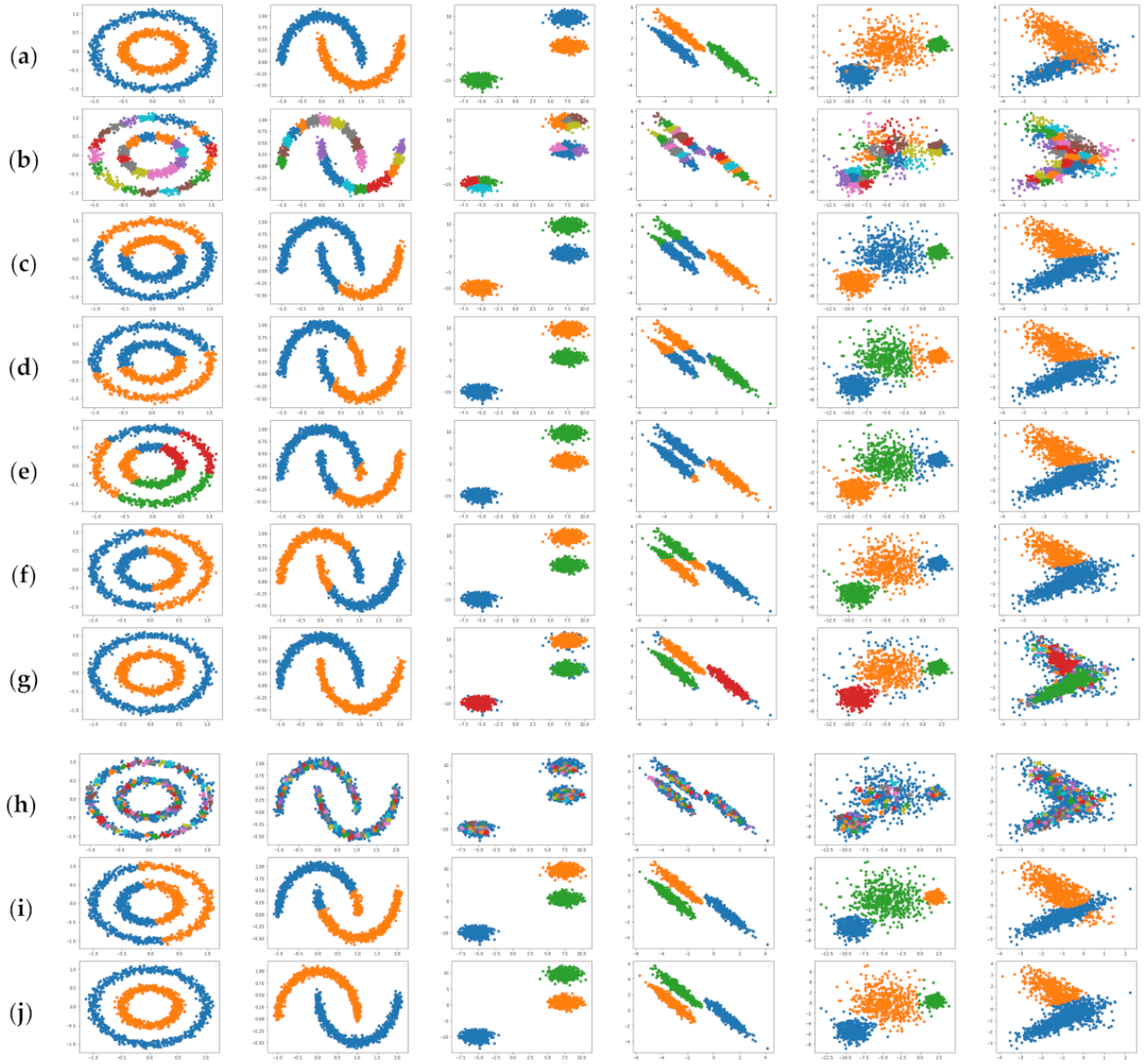


Figure 10. Visual result of each clustering method on each dataset: (**a**) ground truth, (**b**) affinity propagation, (**c**) agglomerative hierarchical, (**d**) k-means, (**e**) mean-shift, (**f**) spectral, (**g**) DBSCAN, (**h**) OPTICS, (**i**) Gaussian mixtures, and (**j**) the proposed clustering methods. Left to right: noisy circles, noisy moons, blobs, anisotropic, varying variance, and overlapping datasets.

4. Discussion

The results further indicate that the FCDCSD can efficiently handle various types of datasets. Among the evaluated algorithms, the proposed method covers a wider variety of data types. It exhibits ideal performance on a majority of data types except for the average performance on the overlapping datasets. Figure 9 shows the total average of all datasets used for the measurements. The FCDCSD performed better than all other methods, followed by DBSCAN.

The FCDCSD can achieve improved accuracy in most dataset scenarios compared with existing methods (Figure 9), and the number of previous clusters does not need to be specified. In addition, the proposed algorithm has a faster execution time (Table 1) and a simpler time complexity because of its simple structure, and its step behavior has no effect on the shape, size, and spatial distribution of the data compared with the other methods. The FCDCSD can preserve meaningful clustering results on large and complex shaped datasets (Figure 10), and it can accommodate noise points in the datasets compared with

most clustering methods. The proposed method can also handle outliers and widely dense datasets, whereas most of the methods fail with outliers or large variation densities. Finally, the FCDCSD can produce clusters in complex shapes, whereas most of the existing methods can only form clusters in elliptical or circular styles.

5. Conclusions

The FCDCSD proposed in this study attempts to solve the current drawbacks of clustering algorithms by utilizing the density-based clustering technique. FCDCSD consists of two main stages. First, all data points are labeled by temporary values, and sets of adjacent values are stored. Second, all sets with common values are merged and resolved to obtain the representative value, replacing their temporary counterparts in the dataset to generate the final clusters. In addition, a post-process is performed to address the noise issue. Experiments were conducted using metrics and synthetic datasets to evaluate the performance of the FCDCSD. The result indicates that the proposed model is superior to affinity propagation agglomerative hierarchical, k-means, mean-shift, spectral, DBSCAN, OPTICS, and Gaussian mixture clustering methods. Furthermore, the FCDCSD can enhance the accuracy of most datasets. It is also easy to develop and fast to implement, and its operations are unaffected by data size or complexity. Finally, the proposed FCDCSD can accommodate noise, effectively handle outliers and datasets with widely varied densities, and produce clusters with complex shapes.

Funding: This research was funded by DEANSHIP OF SCIENTIFIC RESEARCH AT UMM AL-QURA UNIVERSITY, grant number 22UQU4361009DSR01.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Zhao, J.; Ding, Y.; Zhai, Y.; Jiang, Y.; Zhai, Y.; Hu, M. Explore unlabeled big data learning to online failure prediction in safety-aware cloud environment. *J. Parallel Distrib. Comput.* **2021**, *153*, 53–63. [CrossRef]
2. Xu, X.; Ding, S.; Wang, Y.; Wang, L.; Jia, W. A fast density peaks clustering algorithm with sparse search. *Inf. Sci.* **2021**, *554*, 61–83. [CrossRef]
3. Rehman, A.U.; Belhaouari, S.B. Divide well to merge better: A novel clustering algorithm. *Pattern Recognit.* **2022**, *122*, 108305. [CrossRef]
4. Najim Adeen, I.M.; Abdulazeez, A.M.; Zeebaree, D.Q. Systematic review of unsupervised genomic clustering algorithms techniques for high dimensional datasets. *Technol. Rep. Kansai Univ.* **2020**, *62*, 355–374.
5. Wang, H.; Yang, Y.; Liu, B.; Fujita, H. A study of graph-based system for multi-view clustering. *Knowledge-Based Syst.* **2019**, *163*, 1009–1019. [CrossRef]
6. Zhu, X.; Zhang, S.; He, W.; Hu, R.; Lei, C.; Zhu, P. One-Step Multi-View Spectral Clustering. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 2022–2034. [CrossRef]
7. Naik, A.; Reddy, D.; Jana, P.K. A novel clustering algorithm for biological data. In Proceedings of the 2011 Second International Conference on Emerging Applications of Information Technology, Kolkata, India, 19–20 February 2011; pp. 249–252. [CrossRef]
8. Lytvynenko, V.; Lurie, I.; Krejci, J.; Voronenko, M.; Savina, N.; Taif, M.A. Two step density-based object-inductive clustering algorithm. *CEUR Workshop Proc.* **2019**, *2386*, 117–135.
9. Haoxiang, W.; Smys, S. Big data analysis and perturbation using data mining algorithm. *J. Soft Comput. Paradig. (JSCP)* **2021**, *3*, 19–28. [CrossRef]
10. Okagbue, H.I.; Oguntunde, P.E.; Adamu, P.I.; Adejumo, A.O. Unique clusters of patterns of breast cancer survivorship. *Health Technol.* **2022**, *12*, 365–384. [CrossRef]
11. Bateja, R.; Dubey, S.K.; Bhatt, A. Evaluation and Application of Clustering Algorithms in Healthcare Domain Using Cloud Services. In *Second International Conference on Sustainable Technologies for Computational Intelligence*; Springer: Singapore, 2021; pp. 249–261.
12. Hao, Y.; Dong, L.; Liao, X.; Liang, J.; Wang, L.; Wang, B. A novel clustering algorithm based on mathematical morphology for wind power generation prediction. *Renew. Energy* **2019**, *136*, 572–585. [CrossRef]
13. Cai, N.; Diao, C.; Khan, M.J. A Novel Clustering Method Based on Quasi-Consensus Motions of Dynamical Multiagent Systems. *Complexity* **2017**, *2017*, 4978613. [CrossRef]

14. Bataineh, B. A fast and memory-efficient two-pass connected-component labeling algorithm for binary images. *Turk. J. Electr. Eng. Comput. Sci.* **2019**, *27*, 1243–1259. [CrossRef]
15. Bataineh, B.; Abdullah, S.N.H.S.; Omar, K. An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows. *Pattern Recognit. Lett.* **2011**, *32*, 1805–1813. [CrossRef]
16. Bataineh, B.; Abdullah, S.N.H.S.; Omar, K. Adaptive binarization method for degraded document images based on surface contrast variation. *Pattern Anal. Appl.* **2017**, *20*, 639–652. [CrossRef]
17. Pandey, M.; Avhad, O.; Khedekar, A.; Lamkhade, A.; Vharkate, M. Social Media Community Using Optimized Clustering Algorithm. In *ICT Analysis and Applications*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 669–675.
18. Nasrazadani, M.; Fatemi, A.; Nematbakhsh, M. Sign prediction in sparse social networks using clustering and collaborative filtering. *J. Supercomput.* **2022**, *78*, 596–615. [CrossRef]
19. Appiah, S.K.; Wirekoh, K.; Aidoo, E.N.; Oduro, S.D.; Arthur, Y.D. A model-based clustering of expectation–maximization and K-means algorithms in crime hotspot analysis. *Res. Math.* **2022**, *9*, 2073662. [CrossRef]
20. Kumar, J.; Sravani, M.; Akhil, M.; Sureshkumar, P.; Yasaswi, V. Crime Rate Prediction Based on K-means Clustering and Decision Tree Algorithm. In *Computer Networks and Inventive Communication Technologies*; Springer: Singapore, 2022; pp. 451–462.
21. Castañeda, G.; Castro Peñarrieta, L. A Customized Machine Learning Algorithm for Discovering the Shapes of Recovery: Was the Global Financial Crisis Different? *J. Bus. Cycle Res.* **2022**, *18*, 69–99. [CrossRef]
22. Dai, T. Computer Management Method of Foreign Trade Business Expenses Based on Data Analysis Technology. In *2021 International Conference on Big Data Analytics for Cyber-Physical System in Smart City*; Springer: Singapore, 2021; pp. 1037–1043.
23. Alalyan, F.; Zamzami, N.; Bouguila, N. Model-based hierarchical clustering for categorical data. In Proceedings of the 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada, 12–14 June 2019; pp. 1424–1429.
24. Aljibawi, M.; Nazri, M.Z.A.; Sani, N.S. An Enhanced Mudi-Stream Algorithm for Clustering Data Stream. *J. Theor. Appl. Inf. Technol.* **2022**, *100*, 3012–3021.
25. Wang, L.; Leckie, C.; Ramamohanarao, K.; Bezdek, J. Automatically determining the number of clusters in unlabeled data sets. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 335–350. [CrossRef]
26. Ahmed, M.; Seraj, R.; Islam, S.M.S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* **2020**, *9*, 1295. [CrossRef]
27. Huang, J.; Zhu, Q.; Yang, L.; Cheng, D.; Wu, Q. QCC: A novel clustering algorithm based on Quasi-Cluster Centers. *Mach. Learn.* **2017**, *106*, 337–357. [CrossRef]
28. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *KDD* **1996**, *96*, 226–231. Available online: https://www.semanticscholar.org/paper/A-Density-Based-Algorithm-for-Discovering-Clusters-Ester-Kriegel/5c8fe9a0412a078e30eb7e5eeb0068655b673e86 (accessed on 22 September 2022).
29. Zelig, A.; Kaplan, N. KMD clustering: Robust generic clustering of biological data. *bioRxiv* **2020**. [CrossRef]
30. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. OPTICS: Ordering Points to Identify the Clustering Structure. *SIGMOD Rec. (ACM Spec. Interes. Gr. Manag. Data)* **1999**, *28*, 49–60. [CrossRef]
31. Bhattacharjee, P.; Mitra, P. A survey of density based clustering algorithms. *Front. Comput. Sci.* **2021**, *15*, 5–7. [CrossRef]
32. Hahsler, M.; Piekenbrock, M.; Doran, D. dbscan: Fast density-based clustering with R. *J. Stat. Softw.* **2019**, *91*, 1–30. [CrossRef]
33. Mittal, M.; Goyal, L.M.; Hemanth, D.J.; Sethi, J.K. Clustering approaches for high-dimensional databases: A review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*, e1300. [CrossRef]
34. Renjith, S.; Sreekumar, A.; Jathavedan, M. Performance evaluation of clustering algorithms for varying cardinality and dimensionality of data sets. *Mater. Today Proc.* **2020**, *27*, 627–633. [CrossRef]
35. Likas, A.; Vlassis, N.; Verbeek, J.J. The global k-means clustering algorithm. *Pattern Recognit.* **2003**, *36*, 451–461. [CrossRef]
36. Derpanis, K.G. Mean shift clustering. *Lect. Notes* **2005**, *32*, 1–4.
37. Kong, D.; Xie, X.; Zhang, Z. Clustering-based Partitioning for Large Web Graphs. *arXiv* **2022**, arXiv:2201.00472. [CrossRef]
38. Mustafi, D.; Mustafi, A.; Sahoo, G. A novel approach to text clustering using genetic algorithm based on the nearest neighbour heuristic. *Int. J. Comput. Appl.* **2022**, *44*, 291–303. [CrossRef]
39. Kashyap, M.; Gogoi, S.; Prasad, R.K. A Comparative Study on Partition-based Clustering Methods. *Int. J. Create. Res. Thoughts (IJCRT)* **2018**, *6*, 1457–1463. Available online: https://books.google.co.uk/books?hl=en&lr=&id=DEZ1EAAAQBAJ&oi=fnd&pg=PT7&dq=39.%09Kashyap,+M.%3B+Gogoi,+S.%3B+Prasad,+R.K.%3B+Science,+C.+A+Comparative+Study+on+Partition-based+Clustering+Methods&ots=8ud8p982IK&sig=HpmAfa5p3FrBBQf3KDYsy-hNJ7g&redir_esc=y#v=onepage&q&f=false (accessed on 22 September 2022).
40. Fraley, C.; Raftery, A.E. Model-based clustering, discriminant analysis, and density estimation. *J. Am. Stat. Assoc.* **2002**, *97*, 611–631. [CrossRef]
41. McNicholas, P.D. Model-based clustering. *J. Classif.* **2016**, *33*, 331–373. [CrossRef]
42. Johnson, S.C. Hierarchical clustering schemes. *Psychometrika* **1967**, *32*, 241–254. [CrossRef] [PubMed]
43. Nielsen, F. Hierarchical clustering. In *Introduction to HPC with MPI for Data Science*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 195–211.
44. Murtagh, F.; Contreras, P. Algorithms for hierarchical clustering: An overview. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2012**, *2*, 86–97. [CrossRef]

45. Day, W.H.E.; Edelsbrunner, H. Efficient algorithms for agglomerative hierarchical clustering methods. *J. Classif.* **1984**, *1*, 7–24. [CrossRef]
46. Askari, S. Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development. *Expert Syst. Appl.* **2021**, *165*, 113856. [CrossRef]
47. Leski, J.M. Fuzzy c-ordered-means clustering. *Fuzzy Sets Syst.* **2016**, *286*, 114–133. [CrossRef]
48. Zhang, H.; Li, H.; Chen, N.; Chen, S.; Liu, J. Novel fuzzy clustering algorithm with variable multi-pixel fitting spatial information for image segmentation. *Pattern Recognit.* **2022**, *121*, 108201. [CrossRef]
49. Baraldi, A.; Blonda, P. A survey of fuzzy clustering algorithms for pattern recognition—Part II. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1999**, *29*, 786–801. [CrossRef]
50. Chen, J.Y.; He, H.H. A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data. *Inf. Sci.* **2016**, *345*, 271–293. [CrossRef]
51. Wang, M.; Min, F.; Zhang, Z.H.; Wu, Y.X. Active learning through density clustering. *Expert Syst. Appl.* **2017**, *85*, 305–317. [CrossRef]
52. Cai, J.; Wei, H.; Yang, H.; Zhao, X. A Novel Clustering Algorithm Based on DPC and PSO. *IEEE Access* **2020**, *8*, 88200–88214. [CrossRef]