IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 9, 2019, accepted June 26, 2019, date of publication July 3, 2019, date of current version July 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2926579

FREDPC: A Feasible Residual Error-Based Density Peak Clustering Algorithm With the Fragment Merging Strategy

MILAN D. PARMAR[1,2,3], WEI PANG[4,5], DEHAO HAO[3], JIANHUA JIANG[3], WANG LIUPU[1,2], LIMIN WANG[3], AND YOU ZHOU[1,2]

[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Jilin University, Changchun 130012, China
[3]School of Management Science and Information Engineering, Jilin University of Finance and Economics, Changchun 130117, China
[4]Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, U.K.
[5]Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing, Xi'an University of Technology, Xi'an 710048, China

Corresponding author: You Zhou (zyou@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772227 and Grant 61572227, in part by the Science and Technology Development Foundation of Jilin Province under Grant 20180201045GX, in part by the Science Foundation of Education Department of Guangdong Province under Grant 2017KQNCX251 and Grant 2018XJCQSQ026, in part by the Social Science Foundation of Education Department of Jilin Province under Grant JJKH20181315SK, and in part by the Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing, Xi'an University of Technology, under Contract SKL2017CP01.

ABSTRACT The most common issues for many clustering algorithms include the slow convergence, requirement for pre-specification of a number of parameters, and the lack of robustness when dealing with anomalies. Recently, the density peak clustering (DPC) algorithm was proposed to discover the centers of clusters by finding the density peaks in a dataset based on their local densities. The DPC needs neither an iterative process nor a large number of parameters, and it supports a heuristic approach, known as the decision graph, to manually select cluster centroids. However, the selection of the key parameters of the DPC was not systematically investigated. In this paper, we propose the feasible residual error-based density peak clustering algorithm with the fragment merging strategy, where the local density within the neighborhood region is measured through the residual error computation and the resulting residual errors are then used to generate residual fragments for cluster formation. The model parameters are then able to be calculated from the equations with statistical theoretical justification. We also develop a semi-automatic cluster identification method to eliminate the iterative process of manual centroid selection. The robustness and effectiveness of the proposed algorithm compared to the DPC and other clustering algorithms are demonstrated through experiments on standard benchmark datasets. The proposed method named feasible residual error-based density peak clustering (FREDPC) algorithm with the fragment merging strategy only needs to perform in one single step without any iteration and thus it is fast and has a great potential to be applied on a wide range of applications.

INDEX TERMS Clustering, density peak clustering, anomaly detection, residual error, residual fragment.

I. INTRODUCTION

Data Clustering, as an unsupervised learning technique, plays an important role in data mining. Specifically, it aims to organize finite unlabeled data points into disjoint groups on the basis of their intrinsic similarity. Over the last three decades, several strategies have been proposed for clustering, however, they may differ significantly in their definition of cluster and how efficiently clusters are identified. Hence, the clustering result of different clustering algorithms may vary even on the same dataset. Clustering techniques has been widely applied in various fields, such as image understanding [1], [2], pattern recognition in general [3]–[7], health care [8], [9], bioinformatics [10], [11], risk analysis [12], [13], cyber security [14], social networks [15], and astronomy [16].

Also, some emerging fields, such as Virtual Reality [17], big data [18], and Internet of Things (IoT) [19], [20] can benefit from clustering analysis. Clustering methods can be

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

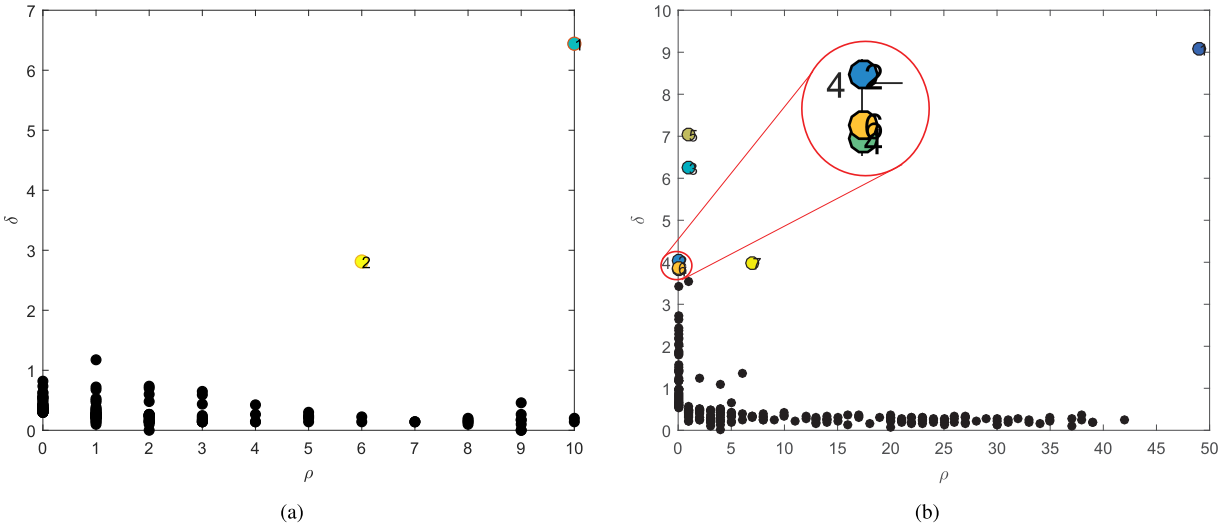


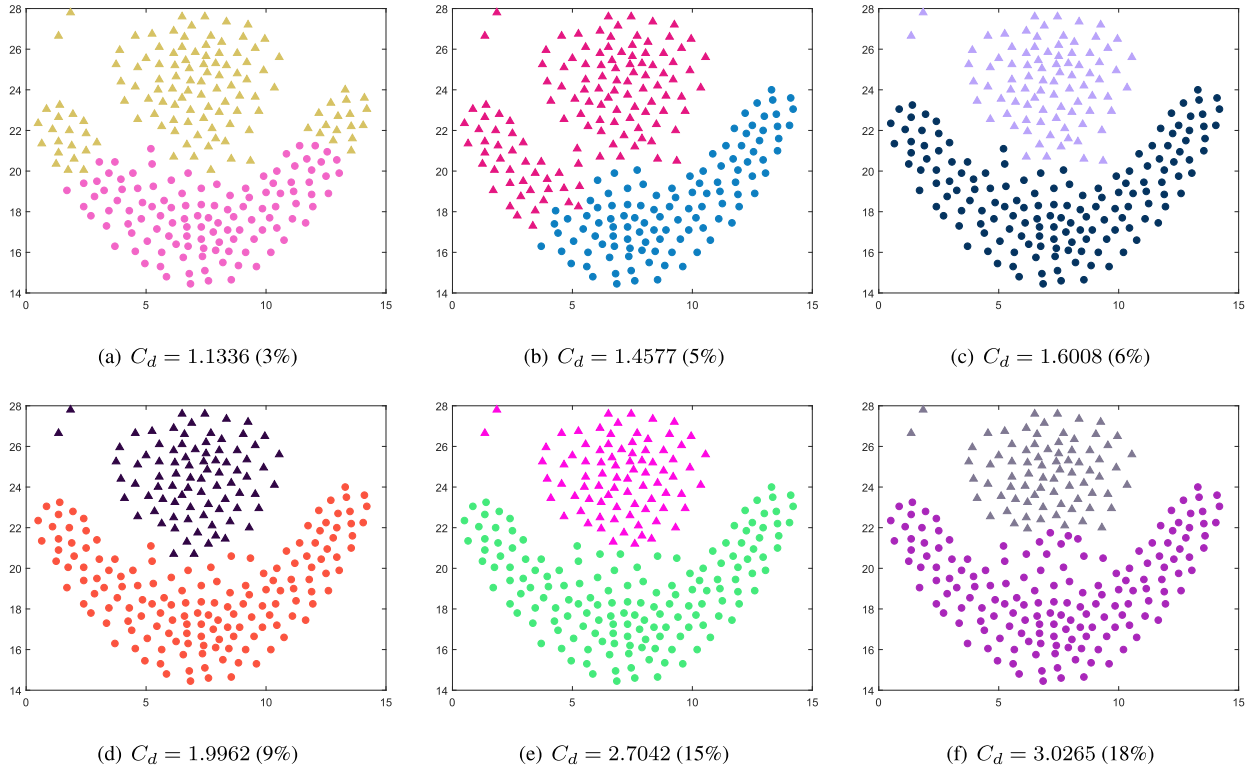
FIGURE 1. Determination of cluster centroids from the decision graph generated by DPC on the *Iris* (a) and *Glass* (b) datasets with $C_d = 0.3000$ and $C_d = 0.5230$, respectively.

generally divided into five groups: hierarchical-based clustering [21]–[23], partitioning-based clustering [24], [25], density-based clustering [26], [27], grid-based clustering [28], and model-based clustering [29], [30].

Density-based clustering algorithms excel in detecting arbitrary-shaped clusters even in the presence of noise in large problem space. Areas of higher density or a set of more densely connected data points than the remainder of the dataset are called clusters. Density is estimated as the number of points in a local environment. Among many density-based spatial clustering algorithms which deal with noise, DBSCAN [31] is one of the most well-known ones, and it uses the concept of local density. In DBSCAN, with the optimal parameter setting, high-density connected regions are merged into a single cluster and noise is detected as data points having low density than the threshold value. Nevertheless, the task of optimizing parameters of DBSCAN can be non-trivial [32]. Recently, Rodriguez and Liao proposed a density-based clustering algorithm called density peak clustering (DPC) [33], which adopted the idea of local density maxima from the mean-shift method [34] and the concept of implementing only single parameter of the distance between data points from K-Medoids [35]. DPC has distinctive features such as i) being able to detect non-spherical clusters based on generated decision graph, ii) less number of control parameters, and iii) relatively low computational complexity. Much research has been carried out on this method [36]–[40].

However, the effectiveness of DPC depends greatly on the cutoff distance parameter C_d to estimate the accurate density in terms of distance between data points. Concretely speaking, DPC uses the heuristic approach of a decision graph to manually select cluster centers, which is regulated by the value of C_d (see Section II-C). Manual selection of cluster centroids is a major limitation of DPC in many applications. For example, as illustrated in Fig. 1(a), apparently only two cluster centroids (labeled as '1' and '2') are clearly identified by the decision graph generated by DPC in the three-cluster *Iris* dataset. Moreover, as shown in Fig. 1(b) that two cluster centroids (labelled as '6' and '4') are misidentified as a single centroid as they are overlapped by the generated decision graph in the six-cluster *Glass* dataset, which makes it extremely difficult for a user to select the exact six clusters. Also, the values of C_d in both cases of the *Iris* and *Glass* datasets are assigned in a systematic manner (cutoff at 1% of the sorted distances among all data points, see Section II-C for more details). A better choice of selecting cluster centroids is related to the user's observation with respect to the nature of the dataset. As such, the performance of DPC is sometimes limited by manual identification of cluster centroids. To the best of our knowledge, robust methods for calculating accurate densities are not available [41], [42], and different methods are required to estimate density based on the nature of the dataset.

Furthermore, DPC lacks robustness when dealing with anomalies. Anomalies are the abnormal patterns found in the dataset, and the presence of anomalies indicate malicious activities that may lead to performance degradation [43]. For example, as illustrated in Fig. 2, it is difficult for DPC to get natural clusters if local densities are randomly distributed [44], such that two anomalies in the top left corner are always considered as part of a larger cluster regardless of different C_d values being used, because there is no "noise-signal cutoff" used in DPC [33]. In such cases, DPC faces the difficulty in identifying the outliers even with varying C_d values, and it may not be able to identify clusters of small sizes or clusters consisting of outliers (relatively speaking) only. In order to improve its capability, Parmar et al. [7] proposed the Residual Error-based Density Peak Clustering (REDPC) that measures local density within a neighborhood



(a) $C_d = 1.1336$ (3%) (b) $C_d = 1.4577$ (5%) (c) $C_d = 1.6008$ (6%)

(d) $C_d = 1.9962$ (9%) (e) $C_d = 2.7042$ (15%) (f) $C_d = 3.0265$ (18%)

FIGURE 2. Visualizations of clusters identified in the *Flame* dataset by DPC with different C_d parameter values.

region by adopting residual error computation so that generated decision graph is better suited for cluster centroid identification. Furthermore, REDPC treats low-density data points as *halo points* and further processes them to detect anomalies. However, the limitation of manual selection of cluster centroid still exists in REDPC.

To overcome the aforementioned issues, in this research we propose the Feasible Residual Error-based Density Peak Clustering (FREDPC) algorithm with the fragment merging strategy. FREDPC adopts residual error computation to better estimate the local density of a dataset such that the generated set of residual errors are used to form *residual fragments* (see Section III-B for more details) and further process them to identify cluster centroids without using the heuristic approach of decision graph. Specifically, unlike DPC, FREDPC performs the reverse approach in the process of cluster formation. Initially FREDPC forms the cluster by merging *residual fragments* with higher similarities (see Section III-C for more details). Furthermore, FREDPC identifies the cluster centroids as the data points with relatively low residual error (see Section III-D for more details), which eventually eliminates the need for decision graph.

The term *density fragment* was originally defined by Jiang et al. [45] as ''a set of data points that consist of density decreasing points with relatively nearby distance''. In this paper, we adopt similar usage: *residual fragments* refer to the set of data point linked with its adjoin points and their respective neighborhood points that require further analysis to detect natural clusters and centroids (see Section III-B for more details). Due to the further analysis applied to *residual fragments*, FREDPC is capable of better identifying and handling various types of anomalies manifested in different patterns in different datasets (see Section III-D for more details).

In order to assess the performance of FREDPC, we compare FREDPC with K-Means [46], affinity propagation (AP) [21], DBSCAN [31], and DPC [33] on twelve UCI datasets and seven synthetic datasets (three synthetic datasets are self-defined but publicly available online). Experimental results show that our algorithm achieves best results on eighteen out of nineteen datasets and the second best on the remaining datasets.

Our main contributions are as follows:

1) We implement the residual error computation so as to compute local densities in underlying datasets within a neighborhood region. As such, the generated set of residual errors are used to form *residual fragments* and further process them to identify clusters and cluster centroids without using the heuristic approach of decision graph.
2) We perform further analysis on *residual fragments* after obtaining the intermediate clustering results. As such, anomalies are effectively identified.
3) We present experimental results on nineteen datasets. In addition, we experimentally show that our proposed FREDPC clustering method performs better than DPC and other benchmark clustering algorithms.

The rest of the paper is structured as follows: Section II presents a brief introduction of DBSCAN, SCAN, and DPC as related work to ours. In Section III we present our proposed residual error-based fragment merging clustering method. In Section IV, to measure the performance of FREDPC, extensive experiments on both real-world and synthetic datasets are conducted with comparisons and discussions. Finally, conclusions and possible further work are given in section V.

II. RELATED WORK

In this section, to introduce some basic ideas and concepts used in our method, we briefly review the technical concepts and detailed steps of three density-based clustering methods: DBSCAN [31], SCAN [47], and DPC [33].

A. DBSCAN: DENSITY-BASED CLUSTERING APPROACH WITH NOISE

DBSCAN [31] is the first and the most well-known representative of the density-based clustering algorithm, and it has been demonstrated to be effective in a lot of real-world applications. It is popular because of the following reasons: i) it is capable of identifying arbitrary-shaped clusters; ii) the specifications of clusters a *priori* is not required; iii) a smaller number of control parameters is needed; iv) it is scalable to large datasets [38]; and v) it is robust against noise. The rationale of this algorithm is to obtain high-density regions as possible clusters ensuring that the density, represented by the number of objects in the neighborhood, exceeds certain specified thresholds. Also, regions with relatively lower density are isolated from the cluster denoted as noise.

In DBSCAN, the definitions of *direct density-reachability*, *density-reachability*, and *density-connectivity* (Definitions 2-4) [31] are used during cluster formation which in turn have asymmetric and symmetric relation between data points of each individual cluster. DBSCAN mainly uses two pre-determined density parameters ε and *MinPts*, and if a data point contains more *MinPts* than the ε-neighborhood a new cluster with core points (i.e., high-density data points within clusters) will be created, then the DBSCAN will gather the density reachable data points from these core data points. When there are no new data points that can be further added into the cluster, DBSCAN will terminate.

DBSCAN has two major advantages in identifying arbitrary-shaped clusters with outlier detection, namely, the formation of a chain structure of high-density data points (i.e., core points) and identification of outliers as low-density data points. Nevertheless, there are distinct limitations of DBSCAN: i) the performance of clustering highly depends on the user-defined parameter values. It is sometimes difficult to estimate appropriate values for various datasets without prior knowledge; ii) it is sensitive to the order of the input parameters: different ordering of data points in the same dataset results in various consequences [48]; and iii) the adjacent clusters of different densities cannot be properly identified possibly due to the use of the global density parameters [49].

B. SCAN: A STRUCTURAL NETWORK CLUSTERING APPROACH

SCAN [47] is a well-known graph partitioning clustering method to understand the elementary notions of structures presented in graphs. It has been successfully applied in many applications due to its two distinctive features: i) it is able to detect not only densely connected data points as clusters but also identifies sparsely connected data points as hubs or outliers using the structure and the connectivity of the vertices as clustering criteria; and ii) it is fast with relatively low computational complexity on a given graph. SCAN is based on the notion that vertices sharing a certain quantity of neighbors should be grouped into one cluster, hubs and outliers should be isolated.

During cluster formation, SCAN identifies data points that have a lot of neighbors with a highly dense connection, i.e., the core point and then uses *vertex structure* [47] to evaluate density. The *vertex structure* (see (1)) of a data point is a set of data points composed of the data point itself and all its neighborhood data points.

Definition 1 (Vertex Structure [47]): Let $v \in V$, the structure of v is defined by its *structural neighborhood*, denoted by $\Gamma(v)$

$$\Gamma(v) = \{w \in V : (v, w) \in E\} \cup \{v\}. \tag{1}$$

The density of neighborhood nodes is computed by the common nodes in the *vertex structure*. SCAN identifies and normalizes the number of common neighborhood data points in two *vertex structures* by the geometric mean of the two *vertex structure's* size. This process is called *structural similarity* and it is defined as follows:

Definition 2 (Structural Similarity [47]): The *structural similarity* between data points v and w, denoted by $\sigma(v, w)$, is defined as follows:

$$\sigma(v, w) = (|\Gamma(v) \cap \Gamma(w)|)/\sqrt{(|\Gamma(v)||\Gamma(w)|)} \tag{2}$$

where $\Gamma(v)$ is defined in (1). When neighborhood data points share many components of its *vertex structures*, their structural similarity is high. SCAN detects core points by evaluating structural similarities for all neighborhoods from Definition 2.

SCAN identifies not only clusters but also outliers. However, its performance highly depends on sensitive input parameters and assumes that the network is homogeneous and the adjacency matrix is already defined. An inspiration drawn from SCAN is that the role of each vertex in a graph can be efficiently measured by *structural similarity* and hence graph partitioning could be an efficient way to aggregate clusters.

C. DENSITY PEAK CLUSTERING (DPC)

DPC is based on the straightforward idea about cluster centroids: i) cluster centroids are characterized by high-density

compare to its neighborhood points, and ii) cluster centroids are positioned at relatively higher distances from other data points with high-density. A decision graph is generated for centroid selection based on two basic attributes of each data point: i) ρ_i (local density) and ii) δ_i (distance between a data point and its nearest neighbor with higher ρ).

For example, a dataset is $X_{P\times Q} = [x_1, x_2, \ldots, x_P]^T$, where $x_i = [x_{1i}, x_{2i}, \ldots, x_{Qi}]$ is a vector with Q number of attributes and P denotes the total number of data points. Initially, the distance matrix of the dataset needs to be computed. Let $d(x_i, x_j)$ denotes the distance between data points x_i and x_j, and it is computed as follows:

$$d(x_i, x_j) = || x_i - x_j ||. \tag{3}$$

For a data point x_i, local density ρ_i is defined as follows:

$$\rho_i = \sum_j \chi \cdot (d(x_i, x_j) - C_d), \tag{4}$$

where $\chi(d(x_i, x_j) - C_d) = 1$ if $d(x_i, x_j) - C_d < 0$ and $\chi(d(x_i, x_j) - C_d) = 0$ otherwise, and C_d represents the cutoff distance. Unlike DBSCAN, the neighborhood radius of DPC is not determined by the direct value (see Section II-A), but by the percentage, and it is the only user defined parameter to distinguish the level of density. In DPC, parameter C_d can be autonomously determined in a systematic way as follows:

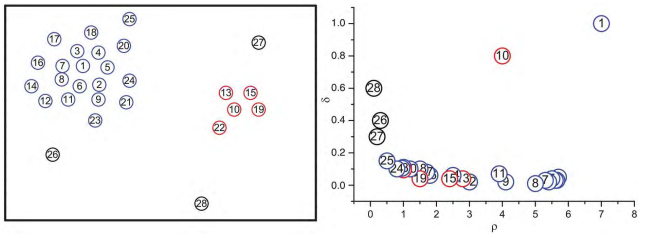
$$C_d = D_{P_d \times \frac{c}{100}}, \tag{5}$$

where $P_d = \binom{P}{2}$ and $D_{P_d \times \frac{c}{100}} \in D = \{d_1, d_2, \ldots, d_{P_d}\}$, wherein D is the set of all distances between every two data points in the dataset, where all the distances are ordered from smallest to largest, and c denotes the user-specified cutoff percentile.

δ_i denotes the minimum distance between the data point x_i and any other data point with higher density. The data points with the highest density locally or globally will have larger values of δ. δ_i is computed as follows:

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho_i} d(x_i, x_j), & \text{if } \exists\, j \text{ s.t. } \rho_j > \rho_i, \\ \max\limits_{j} d(x_i, x_j), & \text{otherwise.} \end{cases} \tag{6}$$

After calculating values of ρ_i and δ_i for each data point in a dataset, DPC generates a decision graph (see Fig. 3) which is plotted with ρ_i as x-axis and δ_i as y-axis and ask a user to identify cluster centroids. According to the guideline, only those points with larger ρ and large δ compared to other data points in the dataset are considered as cluster centroids (see data points '1' and '10' in Fig. 3(b)). However, as shown previously in Fig. 1 that because DPC considers all the data points during the computation of local density (see (4)), it may not perform well on overlapping clusters.



(a) Data points distribution (b) Corresponding decision graph

FIGURE 3. An example of DPC's decision graph (excerpted from [33]).

Furthermore, DPC identifies the border region for each cluster, which contains data points that are part of the underlying cluster and also fall within the C_d range of another cluster. Moreover, DPC traces the data point with maximum density within its border region of the cluster and denotes its density as ρ_b. The data points of the cluster whose density is higher than ρ_b are considered as part of the cluster core and others are considered as part of the cluster halo (suitable to be considered as noise or outliers) [33]. DPC may not be able to process certain low-density data points when they are far from other identified clusters because, according to the definition, halo point has to be close to at least one data point belonging to another cluster. Hence as shown previously in Fig. 2, two data points in the top left corner are always part of the nearest identified cluster regardless of different values of C_d in use.

III. FREDPC: FEASIBLE RESIDUAL ERROR-BASED DENSITY PEAK CLUSTERING ALGORITHM WITH THE FRAGMENT MERGING STRATEGY

In this section, we introduce our proposed clustering algorithm named Feasible Residual Error-based Density Peak Clustering (FREDPC) with the fragment merging strategy for better identification of cluster centroids and detection of anomalies. The proposed FREDPC algorithm inherits the strength of density estimation from DPC, density-connectivity within a neighborhood from DBSCAN, structural similarity measure from SCAN, and density measure from residual error theory.

The overall process of FREDPC consists of the following four stages and each stage is elaborated in the following subsections, respectively.

1) *Preprocessing*: Firstly, the residual error of each data points are computed as local density measurement (see the following subsection III-A).
2) *Residual fragment generation*: Secondly, the residual fragment is generated based on the identification of adjoin points of respective data points along with its link structure and their respective neighborhood points.
3) *Residual fragment aggregation*: Based on the principle of *structural similarity* (see (2)), *structure similarity index* (SSI) between each of residual fragments is computed. As such, the higher the structural similarity value, the higher probability of aggregation between each fragment will be, and similar clusters are formed with fragments of high structural similarity. Moreover, the cluster centroid of the generated cluster is identified as the data point with the lowest residual error.
4) *Final refinements*: Finally, anomalies are isolated as the fragments with the least structural similarity with

other residual fragments and the final clustering results are presented (with anomalies represented using special symbols).

A. PREPROCESSING

For better accurate estimation of local density, which may lead to better cluster formation and centroid identification, we adopt the residual error computation to measure the density of each data point within its neighborhood region. Specifically, the residual error e_{ij} between data point x_i and its neighbor x_j is computed as follows:

$$e_{ij} = \frac{|| x_i - x_j ||}{N}, \tag{7}$$

where N denotes the neighborhood size. It is a user-defined constant parameter used to find N number of the nearest neighbors of x_i, wherein the Euclidean distance is used the same as in DPC (see (3)). Furthermore, the residual error of x_i can be computed as follows:

$$e_i = \sum_j e_{ij} = \sum_j \frac{d(x_i, x_j)}{N}. \tag{8}$$

Comparing (8) with (4), it is obvious that by adopting the residual error computation, when measuring the local density, FREDPC only takes the data points within the neighborhood into consideration. On the other hand, DPC takes all the data points in the entire dataset into consideration. By only considering the local regional density, FREDPC is capable of measuring local density efficiently for better clustering results (see Section IV).

The detailed steps of computing and sorting e_{ij} is summarized in Algorithm 1.

Algorithm 1 The Preprocessing Procedures in FREDPC

Input: Dataset D comprising n number of data points and a user predefined neighborhood size N

Output: Euclidean distance matrix DM of size $n*n$, residual error vector e, $sortd\_e$ (e sorted in ascending order), and index vector of the nearest neighbor of each data point $NNeigh$

```
Compute the Euclidean distance between data points to obtain DM;
for each data point x_i in D do
    find its neighbors N_i based on DM and N;
    for each data point x_j in N_i do
        compute e_ij (see (7));
    end for
end for
aggregate e_ij to obtain e and sort e in ascending order to obtain sortd_e;
obtain NNeigh based on sortd_e;
```

B. RESIDUAL FRAGMENT GENERATION

After preprocessing, we first identify the adjoin point, neighborhood points for each data point to generate residual fragment for cluster formation and later anomaly detection (in the final stage). To identify the adjoin points and neighborhood points, we need to determine the value of the cutoff parameter C_d. Similar to DPC, a cutoff residual C_d value is predefined and the process for selecting C_d is actually that for selecting the average number of neighbors of all data points in the dataset. In FREDPC, C_d can be defined the same as that in DPC (see (5)). FREDPC then initiates the process to generate residual fragments in four phases:

1) *Adjoin point identification*: from the obtained set of sorted residual errors in ascending order $sortd\_e$ only the nearest neighbor of point x_i can be adjoin points only if the distance between x_i and adjoin points x_j is less than the cutoff threshold C_d. Moreover, once the adjoin points of data point x_i is identified, the data point x_i will be excluded from being identified as adjoin points of its adjoin points, and stored in adjoin points set aps, i.e.,:

$$aps = aps \cup x_i, \quad iff \, || x_i - x_j || \; < \; C_d \; \forall \, x_j. \tag{9}$$

2) *Neighborhood points within C_d identification*: from the obtained set of sorted residual errors in ascending order $sortd\_e$, the neighborhood points of each data point within the range of C_d are identified. Also, once the neighborhood points of data point x_i are identified data point x_i will be excluded from being identified as one of the neighborhood points of its identified neighborhood points, and store in $nneighset$, i.e.,:

$$nneigh_x = \{x_j \mid \; || x_i - x_j || \; < \; C_d\}. \tag{10}$$

3) *Adjoin points link generation*: based on aps if the adjoin points of each data point x_i is identified then each data point will connect to its adjoin points to form a link, i.e., $x_i + 1$ within C_d with the principle of *density-reachability* to generate $adjptlink$ and store it in $adjptlinkset$ (see Fig. 7(e)).

4) *Residual fragment generation*: a single *residual fragment* is a structural network composed of link structure of data point and its identified adjoin points, i.e., $adjptlink$ and their respective neighbors $nneighset$:

$$e\mathbb{F}_x = adjptlink \cup (nneigh_x \cup nneigh_y). \tag{11}$$

Similarly, all *residual fragments* can be generated for further processing.

C. RESIDUAL FRAGMENT AGGREGATION

After the generation of *residual fragment*, the *Residual fragment aggregation* can be aggregated based on the principles of *Structural Similarity* (see (2)) and *priori* likelihood. The structural similarity is a score varying from 0% to 100%

Algorithm 2 The Residual Fragment Generation Procedure in FREDPC

Input: DM, e_{ij}, and $sortd\_e$ obtained from Algorithm 1, C_d
Output: Residual Fragment set *fragmentset*

```
for each data point x_i in sortd_e do
  if adjoin point identification criterion is met (see (9))
  then
    update aps accordingly;
    for each neighborhood points of data point x_i do
      if nneigh identification criterion is met (see (10))
      then
        update nneighset accordingly;
      end if
    end for
    connect each data point to its adjoin points to form a
    link, i.e., x_i + 1 and generate adjptlink;
    update adjptlinkset accordingly;
    generate residual fragments based on (11);
    update fragmentset accordingly;
  end if
end for
```

Algorithm 3 The Residual Fragment Aggregation Procedures in FREDPC

Input: Residual fragment set *fragmentset* obtained from Algorithm 2, si_t
Output: Cluster labels assigned to all the data points Cl

```
for each residual fragment do
  Compute structural similarity index based on (12);
  for each e𝔽_sim(x, y) do
    if e𝔽_sim(x, y) > si_t then
      Aggregate residual fragment x and y as one cluster;
    else
      Generate new cluster;
    end if
  end for
end for
for each generated cluster do
  find the data point with the lowest residual error (cluster
  centroid);
  assign x_i with the cluster label of identified centroid of
  respective cluster;
  update Cl accordingly;
end for
```

indicates the scale of matching degree of structural neighborhoods. When adjacent data points share many members of their structural neighborhoods, their structural similarity becomes high. Similarly, the structural network similarity between each *residual fragment* can be defined as:

$$e\mathbb{F}_{sim}(x, y) = \frac{|e\mathbb{F}_x \cap e\mathbb{F}_y|}{\sqrt{|e\mathbb{F}_x||e\mathbb{F}_y|}}, \tag{12}$$

where $e\mathbb{F}_x$ and $e\mathbb{F}_y$ refer to the residual fragments of x and y respectively and $e\mathbb{F}_{sim}(x, y)$ is the *structural similarity index* (SSI) between two residual fragments. The larger the value of $e\mathbb{F}_{sim}(x, y)$, the higher the probability of aggregation between $e\mathbb{F}_x$ and $e\mathbb{F}_y$ is for each pair of *residual fragment*, and the threshold value for $e\mathbb{F}_{sim}(x, y)$ can be denoted as si_t. The threshold values used to distinguish si_t is heuristically determined to be 25%. If the value of $e\mathbb{F}_{sim}(x, y)$ between any pair of the *residual fragment* is more than the value of si_t, the aggregation of those residual fragments is processed with a priori likelihood that the residual fragments with the lowest residual have a higher priority to amalgamate with other residual fragments to form a cluster, as shown below:

$$e\mathbb{F}_{xy} = e\mathbb{F}_x \cup e\mathbb{F}_y, \ \ iff \ \ e\mathbb{F}_{sim}(x, y) > si_t. \tag{13}$$

After cluster formation, the cluster centroid for each generated cluster is identified as the data point with the relatively lowest residual error and the cluster labels Cl of the remaining data points are assigned according to the identified cluster centroid. The detailed steps of residual fragment aggregation procedures in FREDPC are summarized in Algorithm 3.

D. FINAL REFINEMENTS

Anomaly (outlier) detection is a common problem for clustering algorithms in data analysis. The anomalous data points in the dataset can be defined as a deviation from normal behavior and can be associated with the erroneous conditions or malevolent activities that may evolve gradually over time. Therefore, in FREDPC, we further detect the anomalies and highlight them during visualization.

After residual fragment aggregation based on $e\mathbb{F}_{sim}(x, y)$, the clusters that are generated which are composed of only a single data point is considered as anomalies (e.g., see Fig. 7(f) and Fig. 8(f) in Section IV-B). All the anomalies are collected in a set called *anoset*.

Furthermore, for each detected anomalies we carry out further investigation for the most possible cluster label. First, we find the nearest neighbors of each anomaly in *anoset* with the neighborhood size as the same as N defined in (7). Moreover, if there exist other anomalies within the neighborhood of anomaly, we then reject these anomalies from the neighborhood as their cluster labels are yet to be decided. Finally, we assign the cluster label of each anomaly to the majority cluster label in its neighborhood (if the majority ties, we assign the cluster label of the nearest data point belonging to any of the tying clusters). After implementing anomaly refinement process, clustering results may be improved and the detected anomalies are also highlighted visually for human inspections (see Section IV-B).

The detailed steps of anomaly refinement procedures in FREDPC are summarized in Algorithm 4.

E. COMPLEXITY ANALYSIS

The detailed steps of FREDPC is depicted in Fig. 4, wherein the information flow among the underlying dataset, user inputs, and the FREDPC algorithms are explicitly shown.

Algorithm 4 The Final Refinement Procedures in FREDPC

Input: Clusters with cluster labels *Cl*

Output: *anoset* (vector of anomalies points)

```
for each generated cluster do
  if the cluster consist of only one data point then
    Cluster is an anomaly;
    update anoset accordingly;
  end if
end for
for each data point x_j in anoset do
  find the neighbors N_j according to N;
  remove anomalies (both assigned and yet-to-be-
  assigned) from N_j;
  assign the cluster label of x_j to the majority cluster label
  in N_j;
end for
```

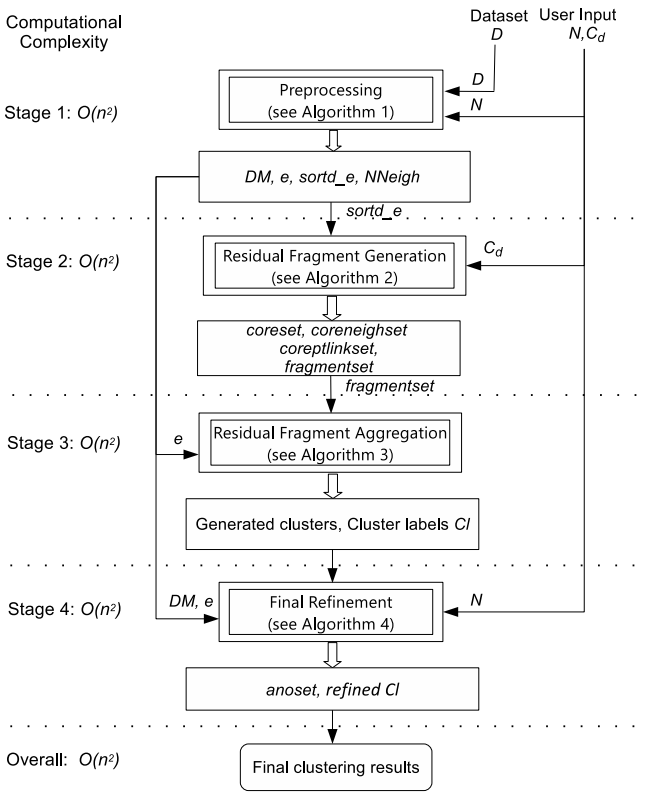


FIGURE 4. The work flow of the overall FREDPC algorithm.

The computational complexity of FREDPC for each of the stages i.e., *Preprocessing*, *Initial assignments*, *Residual fragment aggregation*, and *Final Refinement* are shown in the Table 1, wherein n denotes the number of data points in the underlying dataset. In comparison with other clustering algorithms, Table 2 shows that FREDPC has a moderate level of complexity, i.e., $O(n^2)$ in comparison with other clustering algorithms benchmarked in this paper, where I denotes the number of iterations and K denotes the user predefined number of clusters. Furthermore, in the following experiment section, for the further analysis we also compare the computational time taken by all the clustering algorithms.

TABLE 1. Computational complexity of FREDPC.

Stage:	Algo. 1	Algo. 2	Algo. 3	Algo. 4	Overall Algo
Complexity:	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n^2)$

TABLE 2. Computational complexity comparisons.

Algorithm:	K-Means	AP	DBSCAN	DPC	FREDPC
Complexity:	$O(IKn)$	$O(In^2)$	$O(n \log n)$	$O(n^2)$	$O(n^2)$

IV. EXPERIMENTS

To evaluate the robustness and test the feasibility of the proposed FREDPC, we compare its performance on twelve UCI[1] datasets, namely *Iris*, *Thyroid*, *Liver*, *Ecoli*, *Pima*, *Breast*, *Glass*, *Wine*, *Vehicle*, *German*, *Ionosphere*, and *Sonar*, four widely used synthetic datasets,[2] namely *Flame*, *Aggression*, *Spiral*, and *R15*, and three self-defined datasets,[3] namely *Twenty*, *D1* and *D2* with K-Means [46], AP [21], DBSCAN [31] and DPC [33]. The properties of all nineteen datasets are listed in Table 3.

TABLE 3. Properties of the UCI and synthetic datasets.

Datasets		Properties		
		# Data Points	*# Dimensions*	*# Clusters*
UCI	Iris	150	4	3
	Thyroid	215	5	3
	Liver	345	6	2
	Ecoli	366	7	8
	Pima	768	8	2
	Breast	277	9	2
	Glass	214	9	6
	Wine	178	13	3
	Vehicle	846	18	4
	German	1000	24	2
	Ionosphere	351	34	2
	Sonar	208	60	2
synthetic	Flame	240	2	2
	Aggregation	788	2	7
	Spiral	312	2	3
	R15	600	2	15
self-defined	Twenty	1000	2	20
	D1	87	2	3
	D2	85	2	4

To demonstrate the effectiveness of our FREDPC algorithm, we use F-score to assess the quality of clustering results. In Table 4, we compare the performance of our proposed algorithm along with all benchmarking models (average of 10 independent runs) and visualize

[1]The UCI datasets are available online: http://archive.ics.uci.edu/ml/datasets.php

[2]The synthetic datasets (with cluster labels) are available online: http://cs.joensuu.fi/sipu/datasets/

[3]The self-defined datasets (with cluster labels) are available online: https://github.com/milaan9/Clustering-Datasets/tree/master/02.%20Synthetic

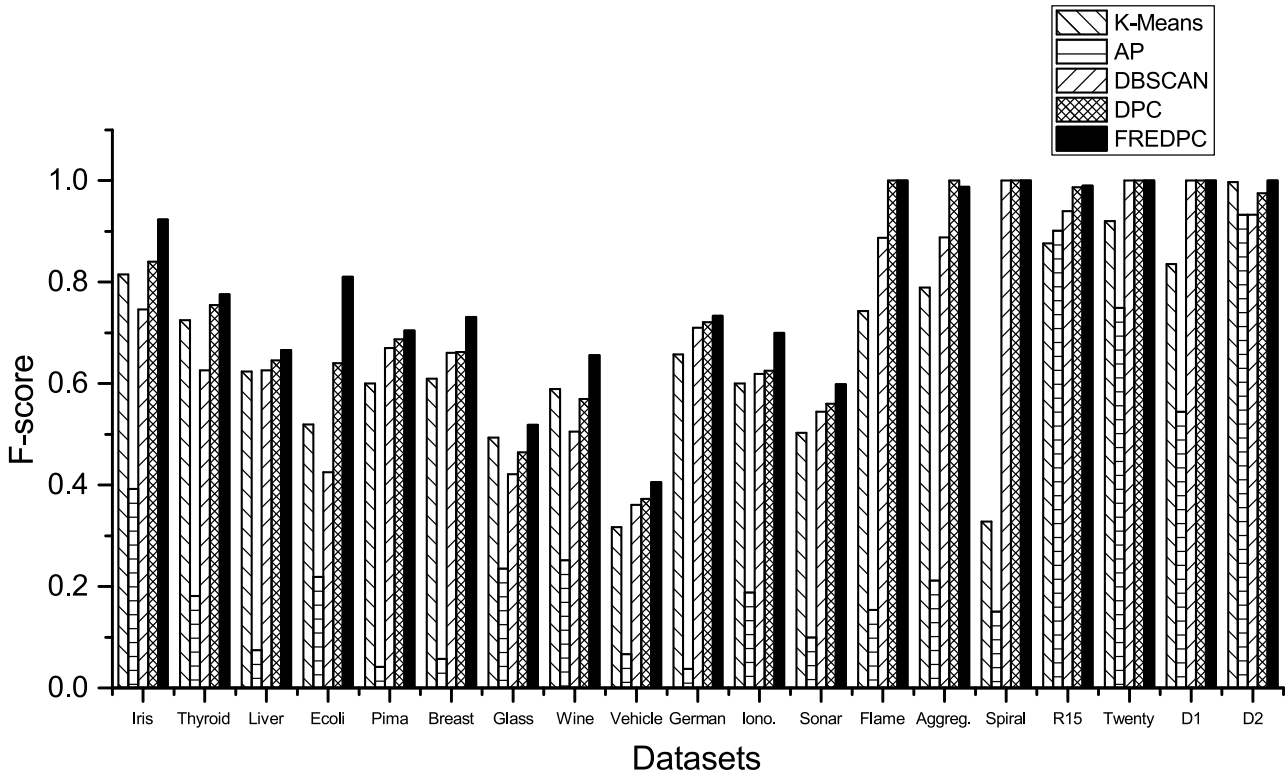


FIGURE 5. Visualization of performance comparison on nineteen datasets.

TABLE 4. Performance comparsion.

Datasets	Clustering Algorithms				
	K-Means	AP	DBSCAN	DPC	FREDPC
Iris	0.8149	0.3924	0.7462	0.8404	**0.9234**
Thyroid	0.7251	0.1814	0.6264	0.7545	**0.7760**
Liver	0.6237	0.0748	0.6264	0.6460	**0.6661**
Ecoli	0.5192	0.2191	0.4253	0.6404	**0.8104**
Pima	0.6001	0.0412	0.6700	0.6874	**0.7045**
Breast	0.6100	0.0569	0.6606	0.6625	**0.7316**
Glass	0.4936	0.2355	0.4213	0.4641	**0.5183**
Wine	0.5896	0.2521	0.5052	0.5699	**0.6560**
Vehicle	0.3168	0.0669	0.3606	0.3728	**0.4058**
German	0.6577	0.0377	0.7104	0.7214	**0.7333**
Ionosphere	0.6004	0.1882	0.6188	0.6252	**0.6999**
Sonar	0.5032	0.0995	0.5443	0.5600	**0.5988**
Flame	0.7432	0.1538	0.8873	**1.0000**	**1.0000**
Aggregation	0.7890	0.2117	0.8885	**1.0000**	0.9880
Spiral	0.3278	0.1505	**1.0000**	**1.0000**	**1.0000**
R15	0.8765	0.9018	0.9398	0.9867	**0.9900**
Twenty	0.9202	0.7496	**1.0000**	**1.0000**	**1.0000**
D1	0.8352	0.5445	**1.0000**	**1.0000**	**1.0000**
D2	0.9976	0.9332	0.9332	0.9756	**1.0000**

them in Fig. 5. The number highlighted in bold indicates the corresponding algorithm has the best performance in terms of its corresponding evaluation, i.e., the corresponding column. As we can see, the F-score obtained by FREDPC are best on eighteen out of nineteen datasets. Nevertheless, FREDPC achieves second best on *Aggregation*, with minor difference of 1.000 - 0.9880 = 0.012. After a further investigation in terms of correctly identified labeled data points, we find that the difference between DPC and FREDPC is three data points out of 788 data points. Nevertheless, the clustering accuracy of DPC is slightly better than FREDPC, this small amount may not be significant. In the following subsections, we further examine the capability of FREDPC in various aspects, respectively.

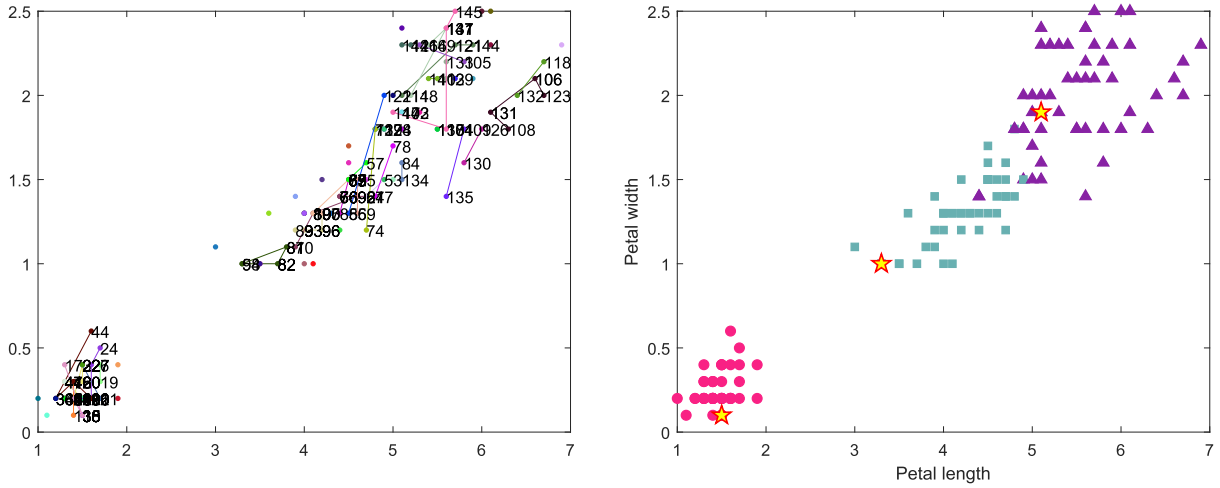
Moreover, we also compare the computational time taken by each algorithm (average of 10 runs for all the method) shown in Table 5. The comparison results are consistent with Table 2 indicating that the algorithm with lowest computational complexity such as DBSCAN requires the minimum

TABLE 5. Comparisons on computational time spent (in ms).

Datasets	Clustering Algorithms				
	K-Means	AP	DBSCAN	DPC	FREDPC
Iris	148.2681	67.6571	1.9672	70.6112	99.6703
Thyroid	149.1138	551.4389	3.3243	70.8240	99.3667
Liver	165.4510	276.5972	3.1133	65.1465	131.2290
Ecoli	215.1643	284.7146	12.8666	81.8478	125.8634
Pima	180.6465	162340.5254	20.2750	100.8641	338.0764
Breast	154.2995	358.9359	2.8905	62.1962	123.6868
Glass	213.3651	150.1824	4.0812	50.3656	119.3801
Wine	156.5255	70.2968	1.4468	69.4427	100.5876
Vehicle	172.2212	2307.8098	21.8605	105.8216	332.6377
German	152.0075	3503.5385	38.6902	114.6180	346.9613
Ionosphere	155.3105	20188.9467	5.7305	61.2720	131.9599
Sonar	160.9591	156.4610	3.1025	58.3100	98.0136
Flame	150.5154	119.2546	1.7628	72.8839	97.6558
Aggregation	174.6097	1853.7493	10.2566	123.0901	333.1829
Spiral	160.1743	180.8485	3.1349	91.4048	111.9358
R15	201.1591	1247.3145	8.4695	92.3365	210.5319
Twenty	187.5755	3602.2910	19.7907	132.0903	350.3081
D1	149.6506	34.3190	1.1136	67.6673	90.0294
D2	146.9794	32.2693	0.8710	68.8392	92.2798

TABLE 6. Parameter settings.

Datasets	Clustering Algorithms				
	K-Means	AP	DBSCAN	DPC	FREDPC
Iris	$K = 3$	Preference = -2.3292	$\varepsilon = 1.0$, *MinPts* = 0.5	$C_d = 8\%$	$N = 2$, $C_d = 18\%$
Thyroid	$K = 3$	Preference = -15.0645	$\varepsilon = 0.5$, *MinPts* = 5.0	$C_d = 5\%$	$N = 3$, $C_d = 40\%$
Liver	$K = 2$	Preference = -40.6202	$\varepsilon = 5.0$, *MinPts* = 1.1	$C_d = 1\%$	$N = 3$, $C_d = 40\%$
Ecoli	$K = 8$	Preference = -0.5573	$\varepsilon = 1.0$, *MinPts* = 5.0	$C_d = 1\%$	$N = 35$, $C_d = 3.8\%$
Pima	$K = 2$	Preference = -0.5611	$\varepsilon = 0.3$, *MinPts* = 5.0	$C_d = 6\%$	$N = 2$, $C_d = 23\%$
Breast	$K = 2$	Preference = -3.9573	$\varepsilon = 0.5$, *MinPts* = 3.0	$C_d = 5\%$	$N = 2$, $C_d = 14\%$
Glass	$K = 6$	Preference = -2.3337	$\varepsilon = 0.5$, *MinPts* = 5.0	$C_d = 3\%$	$N = 2$, $C_d = 47\%$
Wine	$K = 3$	Preference = -280.2412	$\varepsilon = 0.5$, *MinPts* = 5.0	$C_d = 5\%$	$N = 2$, $C_d = 30\%$
Vehicle	$K = 4$	Preference = -2.0601	$\varepsilon = 0.5$, *MinPts* = 5.0	$C_d = 5\%$	$N = 3$, $C_d = 20\%$
German	$K = 2$	Preference = -29.7321	$\varepsilon = 10.0$, *MinPts* = 4.0	$C_d = 4\%$	$N = 4$, $C_d = 38\%$
Ionosphere	$K = 2$	Preference = -4.0204	$\varepsilon = 0.4$, *MinPts* = 5.0	$C_d = 5\%$	$N = 3$, $C_d = 32\%$
Sonar	$K = 2$	Preference = -1.7763	$\varepsilon = 0.5$, *MinPts* = 4.0	$C_d = 8\%$	$N = 3$, $C_d = 30\%$



(a) Adjoin points link by FREDPC (b) Cluster formation by FREDPC

FIGURE 6. Cluster formation based on the adjoin points links generated by FREDPC on the *Iris* dataset.

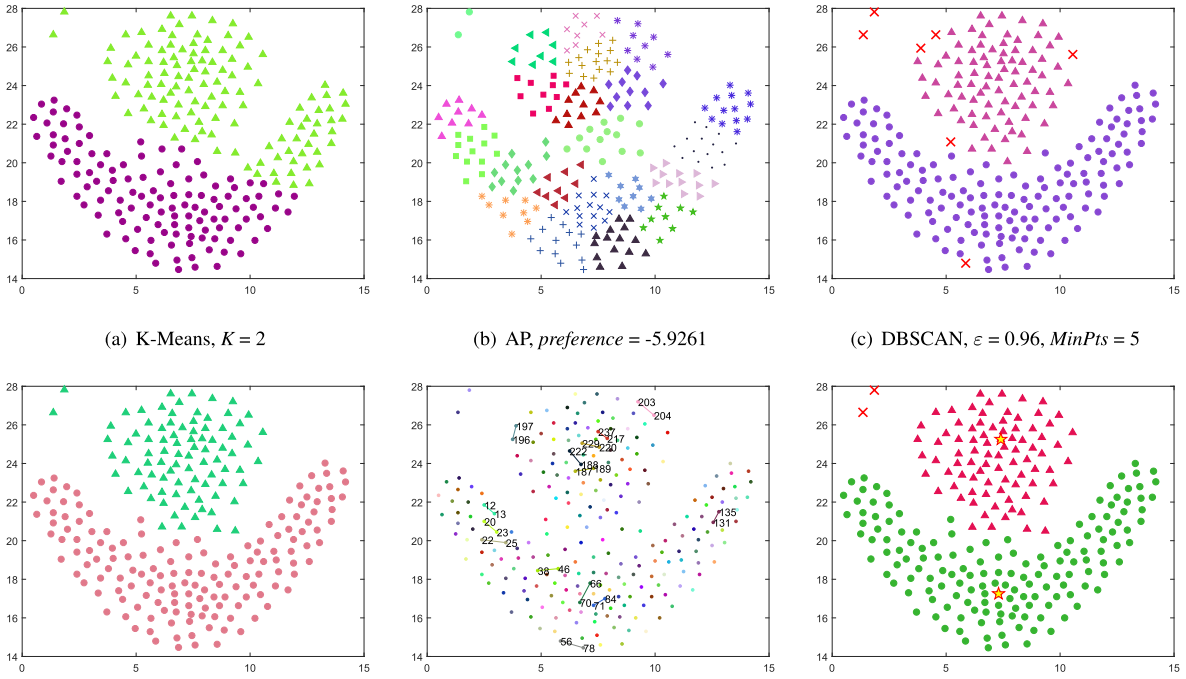
computational time and the algorithms with moderate level of computational complexity such as DPC and FREDPC require a moderate amount of computational time. Although, DPC and FREDPC both have the same computational complexity of $O(n^2)$, overall FREDPC is approximately 60 ms slower. This is because the additional anomaly refinement procedure implemented by FREDPC for better handling the anomalies (see Algorithm 4). However, this compensation on computation time significantly improves the FREDPC's performance (see Table 4). The detailed parameter settings used for each method for the purpose of evaluating the performance and computational time spent by each method for UCI datasets are reported in Table 6 and for synthetic and self-defined datasets are shown in Fig. 7 to Fig. 13. We implemented all clustering algorithms using MATLAB 2016 and the experiments were conducted on the same 64-bit desktop computer installed with Intel(R) Core(TM) i3-4160 CPU at 3.60 GHz and 8 GB RAM.

A. DETECTING CLUSTER CENTROIDS

As aforesaid that the performance of DPC is sometimes limited by its heuristic approach of decision graph and manual selection of cluster centroids. In comparison, FREDPC has a relative advantage in automatic centroid detection. The reason being is that FREDPC measure the local density by employing residual error computation which facilitates in the generation of residual fragments, and then the clusters are generated by aggregating residual fragments. The cluster centroid is identified as the data point of the cluster with the lowest residual error. The cluster labels *Cl* of the remaining data points of the cluster are assigned according to the cluster centroid. As shown in Fig. 6, FREDPC achieves satisfactory results without human intervention.

B. DETECTING CLUSTERS WITH ANOMALIES

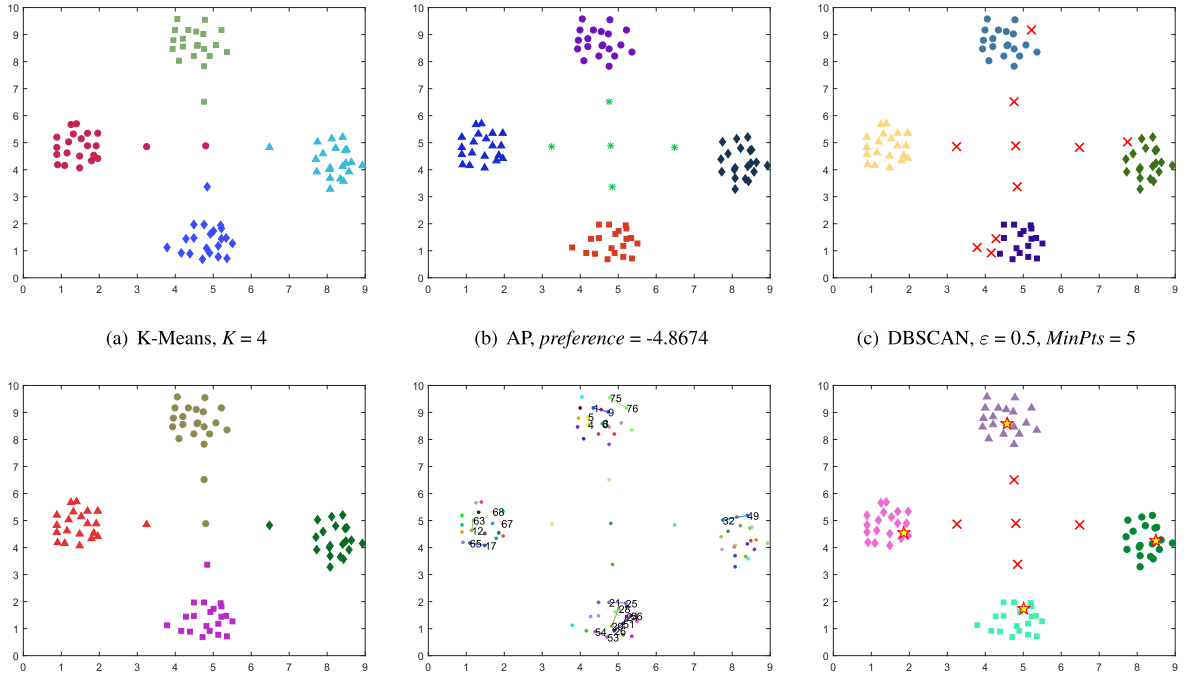
Anomaly detection is a fundamental feature of the clustering algorithm. Datasets *Flame* and *D2* can be adopted to test the capability of FREDPC in anomaly detection in Fig. 7 and Fig. 8, respectively. In the *Flame* dataset the two anomalies are located in the top left corner and in the *D2* dataset the five anomalies are located in the center. As clearly shown in Fig. 7 and Fig. 8, neither K-Means nor AP has the capability to identifying anomalies. While DBSCAN adopts *MinPts* and *density-reachability* to identify anomalies,



(a) K-Means, $K = 2$ (b) AP, *preference* = -5.9261 (c) DBSCAN, $\varepsilon = 0.96$, *MinPts* = 5

(d) DPC, $C_d = 7\%$ (e) Adjoin points links by FREDPC (f) FREDPC, $N = 29$, $C_d = 3\%$

FIGURE 7. **Clustering results on *Flame* dataset.**



(a) K-Means, $K = 4$ (b) AP, *preference* = -4.8674 (c) DBSCAN, $\varepsilon = 0.5$, *MinPts* = 5

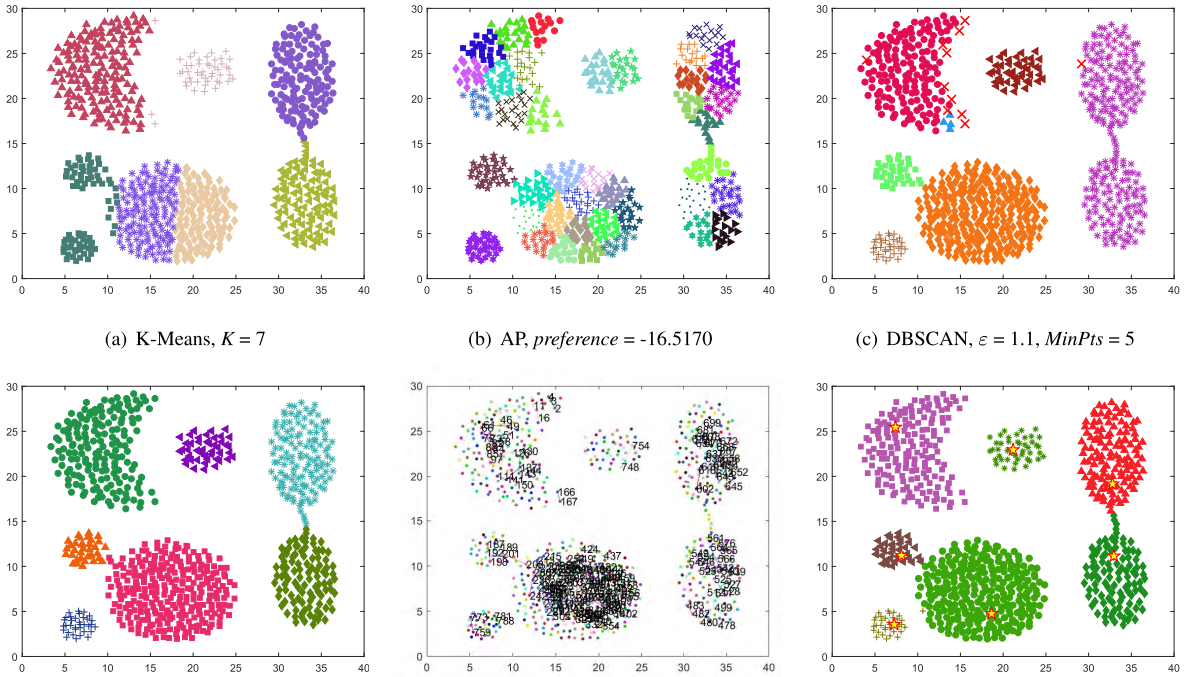
(d) DPC, $C_d = 3\%$ (e) Adjoin points link by FREDPC (f) FREDPC, $N = 4$, $C_d = 10\%$

FIGURE 8. **Clustering results on *D2* dataset.**

the overall performance in anomaly detection is unsatisfactory. As previously discussed DPC fails in anomaly detection. However, as illustrated in Fig. 7(f) and Fig. 8(f), only FREDPC can correctly identify all the possible anomalies.

C. DETECTING CLUSTERS OF ARBITRARY SHAPES

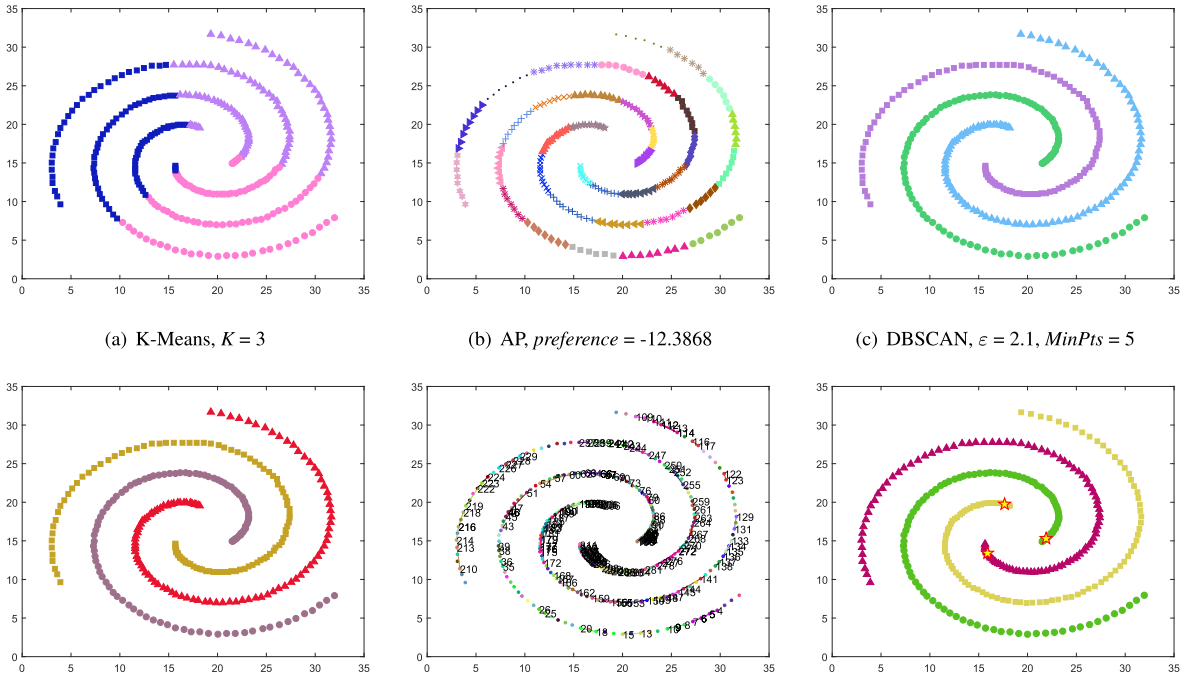
As discussed in literature review (see Section II-A) that the density-based clustering algorithms have the capability of identifying arbitrary-shaped clusters. As illustrated in Fig. 7,



(a) K-Means, $K = 7$ (b) AP, *preference* = -16.5170 (c) DBSCAN, $\varepsilon = 1.1$, *MinPts* = 5

(d) DPC, $C_d = 5\%$ (e) Adjoin points link by FREDPC (f) FREDPC, $N = 45$, $C_d = 4.8\%$

FIGURE 9. **Clustering results on *Aggregation* dataset.**

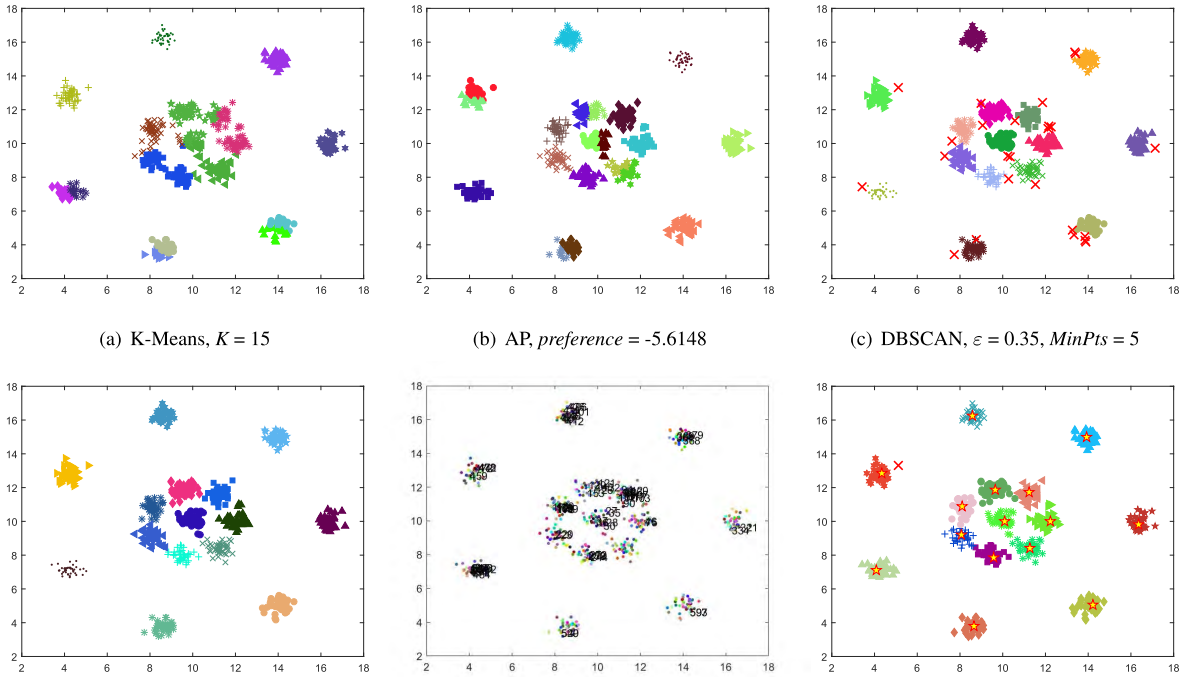


(a) K-Means, $K = 3$ (b) AP, *preference* = -12.3868 (c) DBSCAN, $\varepsilon = 2.1$, *MinPts* = 5

(d) DPC, $C_d = 5\%$ (e) Adjoin points link by FREDPC (f) FREDPC, $N = 4$, $C_d = 4\%$

FIGURE 10. **Clustering results on *Spiral* dataset.**

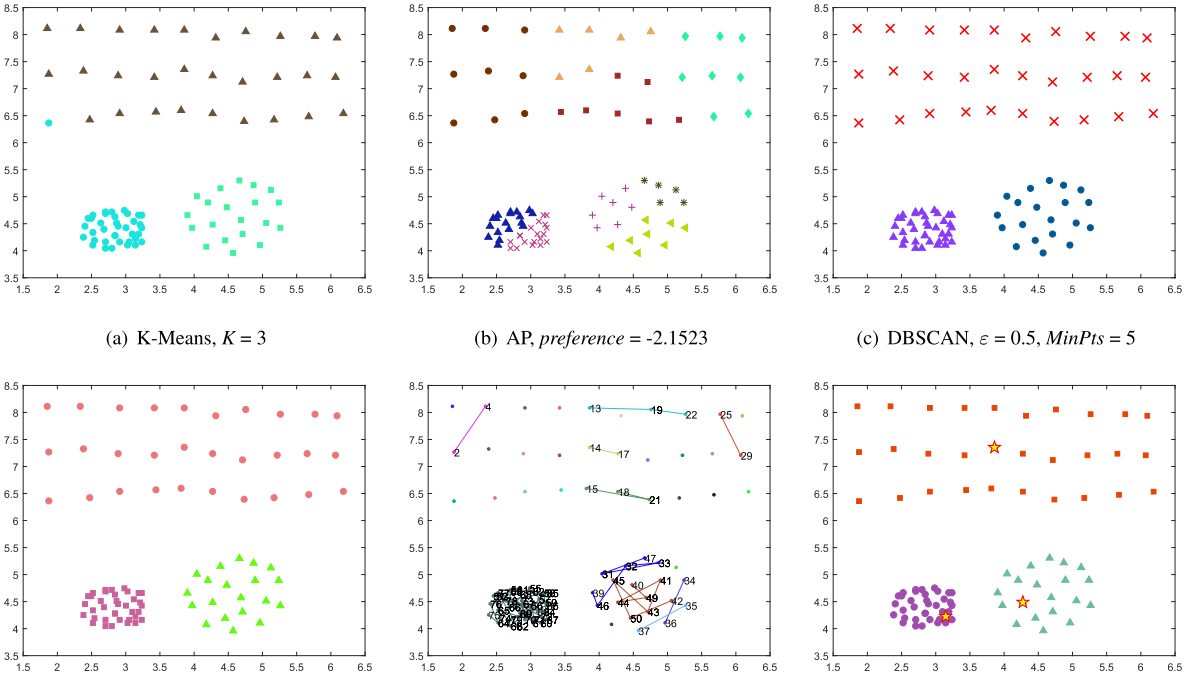
Fig. 9, Fig. 10, Fig. 11, and Fig. 13, K-Means and AP partition some of the natural clusters and hence they are unable to perform well on these datasets. DBSCAN can correctly identify all three clusters in *Spiral* and *Twenty* datasets but it fails to perform well in other datasets due to incorrect anomaly detection. On the other hand, both FREDPC and DPC can



(a) K-Means, $K = 15$ (b) AP, *preference* = -5.6148 (c) DBSCAN, $\varepsilon = 0.35$, *MinPts* = 5

(d) DPC, $C_d = 5\%$ (e) Adjoin points link by FREDPC (f) FREDPC, $N = 23$, $C_d = 4.2\%$

FIGURE 11. **Clustering results on *R15* dataset.**



(a) K-Means, $K = 3$ (b) AP, *preference* = -2.1523 (c) DBSCAN, $\varepsilon = 0.5$, *MinPts* = 5

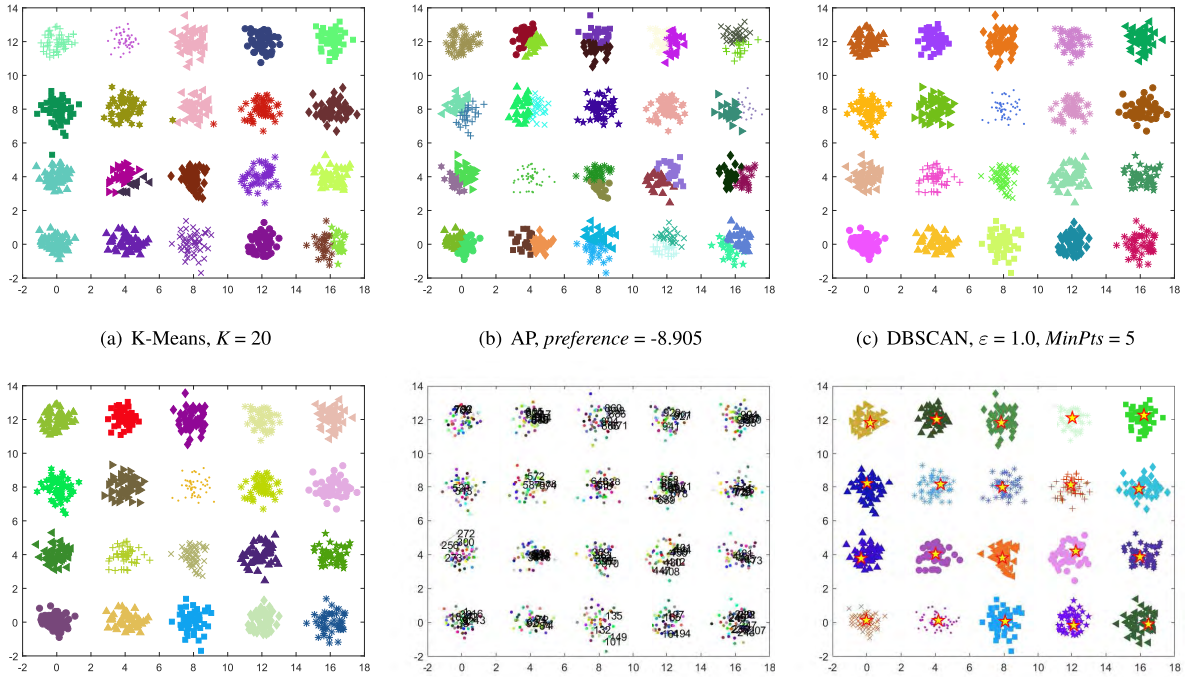
(d) DPC, $C_d = 15\%$ (e) Adjoin points link by FREDPC (f) FREDPC, $N = 11$, $C_d = 25\%$

FIGURE 12. **Clustering results on *D1* dataset.**

identify spherical shape clusters in Fig. 7(f) and Fig. 9(f), linear shape clusters in Fig. 10(f), and also arbitrary shape clusters in Fig. 11, and Fig. 13.

D. DETECTING CLUSTERS OF VARYING SIZES

Dataset *Aggregation* and *D1* are selected to evaluate the performance of FREDPC in processing dataset with variable



(a) K-Means, $K = 20$ (b) AP, *preference* = -8.905 (c) DBSCAN, $\varepsilon = 1.0$, $MinPts = 5$

(d) DPC, $C_d = 3\%$ (e) Adjoin points link by FREDPC (f) FREDPC, $N = 29$, $C_d = 4.2\%$

FIGURE 13. Clustering results on *Twenty* dataset.

TABLE 7. Overall performance comparison on different cluster features.

Algorithm	Features				
	Centroid detection	Anomaly detection	Arbitrary shapes	Variable sizes	Different densities
K-Means	$\surd$	∂	$\times$	∂	$\surd$
AP	$\times$	$\times$	$\times$	$\times$	$\times$
DBSCAN	∂	$\surd$	$\surd$	$\surd$	$\surd$
DPC	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
FREDPC	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Symbols '$\times$' denotes poor performance, '∂' denotes acceptable performance, and '$\surd$' denotes excellent performance, respectively. The assignment criteria are based on whether the clustering algorithm performs well on all the datasets used when analyzing the corresponding property (see Sections IV-A to IV-E). Specifically, '$\times$' is assigned if the F-score on any dataset is less than 0.6, '$\surd$' is assigned if the F-scores on all datasets are greater than 0.8, otherwise, '∂' is assigned.

sizes in Fig. 9 and Fig. 12, respectively. It is clearly evident that K-Means and AP are unable to process these two datasets. However, the performance of DBSCAN is better, but it still misidentifies some borderline points as anomalies. When compared with DPC, FREDPC show equal capability in clustering clusters with varying sizes.

E. DETECTING CLUSTERS OF DIFFERENT DENSITIES

To demonstrate the ability of FREDPC in identifying clusters of varying density we adopt the *D1* dataset in Fig. 12. It is evident in this figure that K-Means and AP do not perform well in the dataset of varying densities. DBSCAN can only identify two lower clusters and misidentify top cluster of very low density as anomalies as they do not fulfill the density-reachability definition (see Section II-A). In comparison, both DPC and FREDPC can correctly identify all the clusters intuitively.

F. OVERALL COMPARISON OF FREDPC OVER BENCHMARKING MODELS

Compared to other benchmarking clustering algorithms, FREDPC performs better in detecting clusters of various properties in the previous subsections, respectively. The clustering performance of all clustering algorithms is summarized in Table 7. As shown in the table, it can be concluded that the proposed FREDPC method is an effective and well-designed algorithm working well in various performance evaluation aspects.

V. CONCLUSION

In this paper, we propose the Feasible Residual Error-based Density Peak Clustering (FREDPC) algorithm inspired by the idea of residual error and residual fragments. This method measures the local density within a neighborhood region through residual error computation and further processes them to generate residual fragments. As such the generated residual fragments are amalgamated based on the principle of *structural similarity* to improve its competence in cluster centroid identification without human intervention and better identify the possible anomalies. The experimental results of the classical UCI and synthetic datasets show that FREDPC is very effective compared to DPC and other algorithms.

In our future work, we will aim to further reduce the runtime complexity and apply our algorithm to more complex and high dimensional datasets.

REFERENCES

[1] J. Yu, R. Hong, M. Wang, and J. You, "Image clustering based on sparse patch alignment framework," *Pattern Recognit.*, vol. 47, no. 11, pp. 3512–3519, Nov. 2014.

[2] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang, "Multimodal deep autoencoder for human pose recovery," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5659–5670, Dec. 2015.

[3] D. Wang, C. Quek, and G. S. Ng, "Novel self-organizing takagi sugeno kang fuzzy neural networks based on ART-like clustering," *Neural Process. Lett.*, vol. 20, no. 1, pp. 39–51, Aug. 2004.

[4] J. Wen, D. Zhang, Y. Cheung, H. Liu, and X. You, "A batch rival penalized expectation-maximization algorithm for Gaussian mixture clustering with automatic model selection," *Comput. Math. Methods Med.*, vol. 2012, Oct. 2012, Art. no. 425730.

[5] Z. Li, J. Liu, Y. Yang, X. Zhou, and H. Lu, "Clustering-guided sparse structural learning for unsupervised feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2138–2150, Sep. 2014.

[6] J. Zhou, L. Chen, C. L. P. Chen, Y. Zhang, and H.-X. Li, "Fuzzy clustering with the entropy of attribute weights," *Neurocomputing*, vol. 198, pp. 125–134, Jul. 2016.

[7] M. Parmar, D. Wang, X. Zhang, A. H. Tan, C. Miao, J. Jiang, and Y. Zhou, "REDPC: A residual error-based density peak clustering algorithm," *Neurocomputing*, vol. 348, pp. 82–96, Jul. 2019.

[8] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J. F. Chang, and L. Hua, "Data mining in healthcare and biomedicine: A survey of the literature," *J. Med. Syst.*, vol. 36, no. 4, pp. 2431–2448, Aug. 2012.

[9] D. Tomar and S. Agarwal, "A survey on data mining approaches for healthcare," *Int. J. Bio-Sci. Bio-Technol.*, vol. 5, no. 5, pp. 241–266, 2013.

[10] C. S. Greene, J. Tan, M. Ung, J. H. Moore, and C. Cheng, "Big data bioinformatics," *J. Cellular Physiol.*, vol. 229, no. 12, pp. 1896–1900, Dec. 2014.

[11] O. B. Poirion, X. Zhu, T. Ching, and L. Garmire, "Single-cell transcriptomics bioinformatics and computational challenges," *Frontiers Genet.*, vol. 7, p. 163, Sep. 2016.

[12] G. Kou, Y. Peng, and G. Wang, "Evaluation of clustering algorithms for financial risk analysis using MCDM methods," *Inf. Sci.*, vol. 275, pp. 1–12, Aug. 2014.

[13] X. Ding, Y. Tian, and Y. Yu, "A real-time big data gathering algorithm based on indoor wireless sensor networks for risk analysis of industrial operations," *IEEE Trans. Ind. Informat.*, vol. 12, no. 3, pp. 1232–1242, Jun. 2016.

[14] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[15] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, and L. Jiao, "Greedy discrete particle swarm optimization for large-scale social network clustering," *Inf. Sci.*, vol. 316, pp. 503–516, Sep. 2015.

[16] M. M. A. Patwary, N. Satish, N. Sundaram, F. Manne, S. Habib, and P. Dubey, "Pardicle: Parallel approximate density-based clustering," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2014, pp. 560–571.

[17] E. Olshannikova, A. Ometov, Y. Koucheryavy, and T. Olsson, "Visualizing big data with augmented and virtual reality: Challenges and research agenda," *J. Big Data*, vol. 2, no. 1, p. 22, Oct. 2015.

[18] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 267–279, Sep. 2014.

[19] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data mining for the Internet of Things: Literature review and challenges," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 8, p. 431047, 2015.

[20] H. Zhang, T. W. S. Chow, and Q. M. J. Wu, "Organizing books and authors by multilayer SOM," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2537–2550, Dec. 2016.

[21] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

[22] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?" *J. Classification*, vol. 31, no. 3, pp. 274–295, 2014.

[23] P. Nayak and A. Devulapalli, "A fuzzy logic-based clustering algorithm for WSN to extend the network lifetime," *IEEE Sensors J.*, vol. 16, no. 1, pp. 137–144, Jan. 2016.

[24] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 20–35, Feb. 2014.

[25] J. Wu, H. Liu, H. Xiong, J. Cao, and J. Chen, "K-means-based consensus clustering: A unified view," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 155–169, Jan. 2015.

[26] A. Amini, T. Y. Wah, and H. Saboohi, "On density-based data streams clustering algorithms: A survey," *J. Comput. Sci. Technol.*, vol. 29, no. 1, pp. 116–141, Jan. 2014.

[27] X. Li, Q. Han, and B. Qiu, "A clustering algorithm using skewness-based boundary detection," *Neurocomputing*, vol. 275, pp. 618–626, Jan. 2018.

[28] C. Li, L. Li, J. Zhang, and E. Alexov, "Highly efficient and exact method for parallelization of grid-based algorithms and its implementation in DelPhi," *J. Comput. Chem.*, vol. 33, no. 24, pp. 1960–1966, Sep. 2012.

[29] C. Bouveyron and C. Brunet-Saumard, "Model-based clustering of high-dimensional data: A review," *Comput. Statist. Data Anal.*, vol. 71, pp. 52–78, Mar. 2014.

[30] J. Jacques and C. Preda, "Model-based clustering for multivariate functional data," *Comput. Statist. Data Anal.*, vol. 71, pp. 92–106, Mar. 2014.

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

[32] A. V. Krivtsov, D. Twomey, Z. Feng, M. C. Stubbs, Y. Wang, J. Faber, J. E. Levine, J. Wang, W. C. Hahn, D. G. Gilliland, and T. R. Golub, "Transformation from committed progenitor to leukaemia stem cell initiated by MLL–AF9," *Nature*, vol. 442, no. 7104, p. 818, Aug. 2006.

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

[34] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.

[35] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Hoboken, NJ, USA: Wiley, 2009, vol. 344.

[36] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on k-nearest neighbors and principal component analysis," *Knowl.-Based Syst.*, vol. 99, pp. 135–145, May 2016.

[37] J. Xie, H. Gao, W. Xie, X. Liu, and P. W. Grant, "Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors," *Inf. Sci.*, vol. 354, pp. 19–40, Aug. 2016.

[38] R. Mehmood, G. Zhang, R. Bie, H. Dawood, and H. Ahmad, "Clustering by fast search and find of density peaks via heat diffusion," *Neurocomputing*, vol. 208, pp. 210–217, Oct. 2016.

[39] H. Yan, L. Wang, and Y. Lu, "Identifying cluster centroids from decision graph automatically using a statistical outlier detection method," *Neurocomputing*, vol. 329, pp. 348–358, Feb. 2019.

[40] H. Yan, Y. Lu, and L. Li, "A potential-based density estimation method for clustering using decision graph," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, 2017, pp. 73–82.

[41] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," *Ann. Statist.*, vol. 38, no. 5, pp. 2916–2957, 2010.

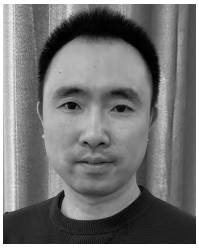
[42] S. Krishnaswamy, M. H. Spitzer, M. Mingueneau, S. C. Bendall, O. Litvin, E. Stone, D. Pe'er, and G. P. Nolan, "Conditional density-based analysis of T cell signaling in single-cell data," *Science*, vol. 346, no. 6213, 2014, Art. no. 1250689.
[43] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, 2009.
[44] M. Wang, W. Zuo, and Y. Wang, "An improved density peaks-based clustering method for social circle discovery in social networks," *Neurocomputing*, vol. 179, pp. 219–227, Feb. 2016.
[45] J. Jiang, X. Tao, and K. Li, "DFC: Density fragment clustering without peaks," *J. Intell. Fuzzy Syst.*, vol. 34, no. 1, pp. 525–536, 2018.
[46] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.
[47] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 824–833.
[48] A. Dockhorn, C. Braune, and R. Kruse, "Variable density based clustering," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[49] Y. Lv, T. Ma, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "An efficient and scalable density-based clustering algorithm for datasets with complex structures," *Neurocomputing*, vol. 171, pp. 9–22, Jan. 2016.

MILAN D. PARMAR received the B.Eng. degree in electronics engineering from the University of Mumbai, India, in 2011, and the M.S. degree in embedded microelectronics and wireless systems from Coventry University, U.K., in 2012. He is currently pursuing the Ph.D. degree in computer science and technology with the Machine Learning Engineering Laboratory, Jilin University. Since 2016, he has been a Foreign Expert Lecturer with the E-Commerce and Data Science Department, Jilin University of Finance and Economics. His current interests include pattern recognition, machine learning, and data mining.

WEI PANG received the Ph.D. degree in computing science from the University of Aberdeen, in 2009, where he is currently a Senior Lecturer. He has authored over 80 papers, including 30+ journal papers. His research interests include bio-inspired computing, data mining, machine learning, and qualitative reasoning.

DEHAO HAO received the B.S. degree in e-commerce from the Jilin University of Finance and Economics, China, in 2016, where he is currently pursuing the M.S. degree in management science and information engineering. His current research interests include clustering algorithms, machine learning, and data mining.

JIANHUA JIANG was born in Hangzhou, China, in 1979. He received the B.S., M.S., and Ph.D. degrees in computer science and technology from Jilin University, Changchun, China, in 2003, 2006, and 2010, respectively. From 2006 to 2013, he was a Lecturer with the Jilin Big Data Research Center for Business. Since 2013, he has been an Associate Professor with the E-Commerce and Data Science Department, Jilin University of Finance and Economics. He has authored three books and more than 30 articles. His research interests include clustering algorithms, cloud computing, finance technology, supply chain optimization, and optimization algorithms. He is a member of the China Computer Federation (CCF). He has three best international conference papers, in 2014 and 2015.



WANG LIUPU received the bachelor's, master's, and Ph.D. degrees from Jilin University, Changchun, China, in 2002, 2006, and 2010, respectively, where he is currently an Associate Professor with the College of Computer Science and Technology. His research interests include machine learning, computational biology, and bioinformatics.

LIMIN WANG received the master's and the Ph.D. degrees in computer science and technology from Jilin University, in 2004 and 2007, respectively. She has published more than 70 papers. Her research interests include machine learning and pattern recognition. She was an ACM member, a member of the China Computer Society CCF, and an Executive Council Member of society in Jilin province at the 7th International Conference of the Electronic Commerce and Electronic Government Affairs Program Committee, new century excellent talents in Jilin Province Colleges and Universities.

YOU ZHOU received the bachelor's and Ph.D. degrees from Jilin University, Changchun, China, in 2002 and 2008, respectively, where he is currently a Professor with the College of Computer Science and Technology. His research interests include machine learning, pattern recognition, and bioinformatics.