7-2019

REDPC: A residual error-based density peak clustering algorithm

Milan PARMAR

Di WANG

Xiaofeng ZHANG

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Chunyan MIAO

See next page for additional authors

REDPC: A Residual Error-based Density Peak Clustering Algorithm

Milan Parmar[a,b,*], Di Wang[c,*], Xiaofeng Zhang[d], Ah-Hwee Tan[c,e], Chunyan Miao[c,e], Jianhua Jiang[b], You Zhou[a,**]

[a] *College of Computer Science and Technology, Jilin University, Changchun, China*
[b] *School of Management Science and Information Engineering, Jilin University of Finance and Economics, Changchun, China*
[c] *Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly, Nanyang Technological University, Singapore*
[d] *Department of Computer Science, Harbin Institute of Technology, Shenzhen, China*
[e] *School of Computer Science and Engineering, Nanyang Technological University, Singapore*

Abstract

The density peak clustering (DPC) algorithm was designed to identify arbitrary-shaped clusters by finding density peaks in the underlying dataset. Due to its aptitudes of relatively low computational complexity and a small number of control parameters in use, DPC soon became widely adopted. However, because DPC takes the entire data space into consideration during the computation of local density, which is then used to generate a decision graph for the identification of cluster centroids, DPC may face difficulty in differentiating overlapping clusters and in dealing with low-density data points. In this paper, we propose a residual error-based density peak clustering algorithm named REDPC to better handle datasets comprising various data distribution patterns. Specifically, REDPC adopts the residual error computation to measure the local density within a neighbourhood region. As such, comparing to DPC, our REDPC algorithm provides a better decision graph for the identification of cluster centroids and better handles the low-density data points. Experimental results on both synthetic and real-world datasets show that REDPC performs better than DPC and other algorithms.

*Milan Parmar and Di Wang contributed equally to this work.
**Corresponding author
Email address: zyou@jlu.edu.cn (You Zhou)

Keywords: clustering, density peak clustering, anomaly detection, residual error, low-density data points

1. Introduction

Clustering algorithms aim to analyze data by discovering their underlying structure and organize them into different categories according to certain characteristic measures, such as internal homogeneity and external bifurcation, without priori-knowledge. Successful applications of clustering techniques are evident in various domains, such as pattern recognition in general [1–4], image understanding [5, 6], bioinformatics [7], lifestyle identification [8], disease diagnosis [9], cyber security [10], risk analysis [11] [12], etc. Moreover, some emerging topics, such as big data [13], virtual reality [14], and Internet of Things (IoT) [15], also avail from clustering methods. In general, clustering methods can be broadly categorized into five groups based on their dynamics, namely partitioning [16] [17], hierarchical [18], density-based [19] [20], model-based [21], and grid-based [22].

Density-based clustering algorithms have been widely applied to form arbitrary-shaped clusters by detecting high-density regions in the data space. Basically, the region with high-density, or a set of densely connected data points, is treated as a cluster. Density-based spatial clustering of applications with noise (DBSCAN) [23] is probably the most well-known density-based clustering algorithm engendered from the basic notion of local density. Recently, density-based clustering methods have attracted more attention since Rodriguez and Liao proposed their density peak clustering (DPC) algorithm [24] in 2014. The desirable features of DPC include i) relatively low computational complexity, ii) small number of control parameters in use, and iii) identification of cluster centroids of varying cluster sizes based on the generated decision graph.

Nonetheless, the performance of DPC highly relies on the value of the cutoff distance parameter C_d, which serves as a threshold to distinguish the level of

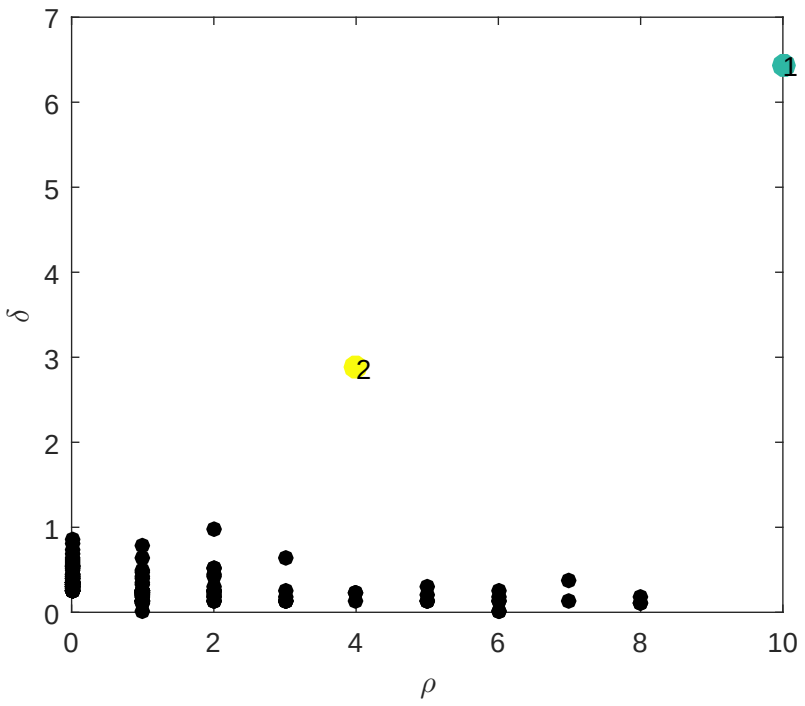
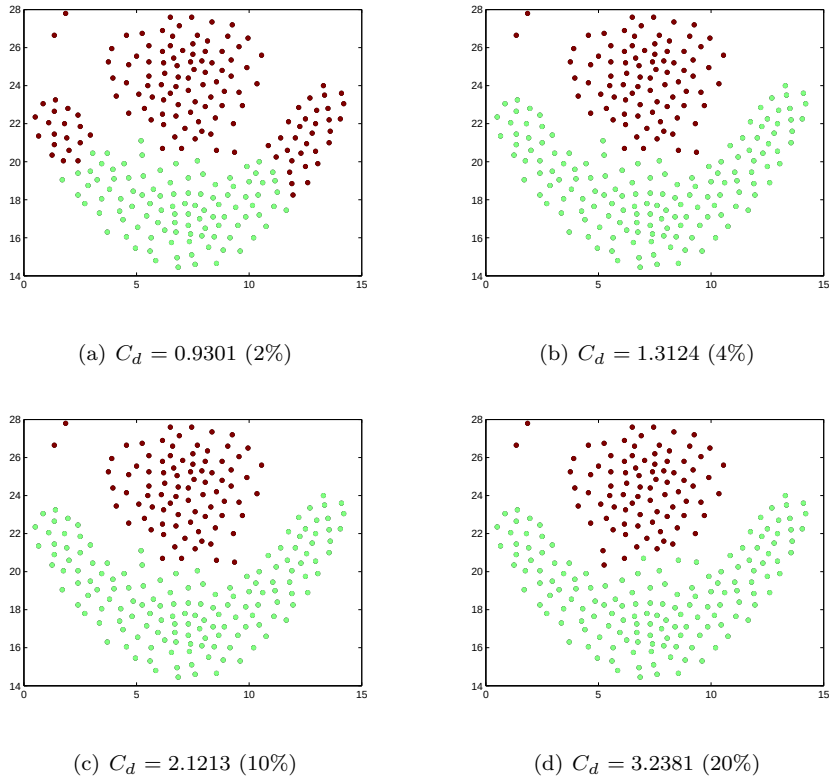


Figure 1: The decision graph generated by DPC on the *Iris* dataset ($C_d = 0.2449$).

density in terms of distance between data points. Specifically, the identification of cluster centroids in DPC is performed manually with the facilitation of a generated decision graph, which is regulated by the value of C_d (see Section 2.2 for more technical details). For example, it is clearly shown in Figure 1 that only two cluster centroids (indexed as '1' and '2' with different colors) in the well-known 3-cluster *Iris* dataset are straightforwardly identified by the decision graph generated by DPC, even if the value of C_d is assigned in a systematic manner (cutoff at 1% of the sorted distances among all data points, see Section 2.2 for more technical details). As such, the performance of DPC is sometimes limited by its way of generating decision graphs.

Moreover, DPC does not perform well on anomaly detection, which is a beneficial function of clustering algorithms that the presence of anomalies indicates possible erroneous conditions that may lead to significant performance degrada-



(a) $C_d = 0.9301$ (2%)

(b) $C_d = 1.3124$ (4%)

(c) $C_d = 2.1213$ (10%)

(d) $C_d = 3.2381$ (20%)

Figure 2: Clusters identified by DPC with different C_d parameter values on the *Flame* dataset.

tion [25]. As shown in Figure 2, DPC does not well handle the uneven cluster distribution (also pointed out in [26]) that the two anomalies (in the top left corner) are always considered as part of a larger cluster regardless of different C_d values in use, because there is no "noise-signal cutoff" used in DPC [24]. In such cases, DPC faces the difficulty in identifying the outliers even with varying C_d values and it may not be able to find clusters of small sizes or consisting of outliers (relatively speaking) only.

Therefore, to generate better decision graphs for cluster centroid identifications by adopting more effective density measurement, and to better detect anomalies for comprehensive clustering results by further examining the borderline data points, in this paper, we propose a density-based clustering method

named Residual Error-based Density Peak Clustering (REDPC). Specifically, REDPC adopts the residual error computation to measure the local density within a neighbourhood region so that the generated decision graphs are better suited for cluster centroid identifications (see Section 3.1 for more technical details). Moreover, REDPC treats low-density data points as *halo points* (see Section 3.3 for more technical details) and further processes them to detect anomalies.

The term *halo* was originally defined by Rodriguez and Laio in [24] as a set of data points in certain *halo* regions that are "suitable to be considered as noise" (see Section 2.2). In this paper, we adopt the similar usage that let *halo points* refer to the set of low-density data points that require further analysis. Due to the further analysis applied to halo points, REDPC is shown to be capable of better identifying and handling various types of anomalies manifested in different patterns in different datasets (see Section 4).

In terms of performance evaluations, we apply REDPC on four UCI datasets and five synthetic datasets (two synthetic datasets are own-defined but publicly available online). For comparison purposes, we also apply K-Means [27], affinity propagation (AP) [18], DBSCAN [23] and DPC [24] on the same datasets. Experimental results show that our algorithm achieves the best performance on most datasets (specifically, best on eight out of nine datasets and the second best on the remaining dataset).

Our main contributions in this paper are listed as follows:

1. We adopt the residual error computation to measure local density within a neighbourhood region. As such, the generated decision graphs are better suited for cluster centroid identifications.
2. We perform further analysis on low-density data points after obtaining the intermediate clustering results. As such, the anomalies and borderline data points are better distinguished.
3. We empirically show with the experimental results on nine datasets that our proposed REDPC clustering method performs better than DPC and

other benchmarking clustering algorithms.

The rest of the paper is organized as follows. We briefly introduce the dynamics of DBSCAN and DPC as related work in Section 2. We present our proposed residual error-based clustering method in Section 3. We report the experimental results with comparisons and discussions in Section 4. We draw the conclusion and propose future work in Section 5.

2. Related Work

In this literature review section, we present the technical concepts and dynamics of two density-based clustering methods, which are closely related to ours. Specifically, we introduce the pros and cons of DBSCAN [23] and DPC [24] in the following two subsections, respectively.

2.1. DBSCAN: Density-Based Clustering Approach with Noise

DBSCAN is probably the most well-known and widely applied density-based clustering algorithm due to its following desirable features: i) efficient identification of arbitrary-shaped clusters, ii) scalability to large datasets [19], iii) small number of control parameters in use, iv) no predetermination on the number of clusters *a priori*, and v) ability to identify outliers. The dynamics of DBSCAN are based on the notion that clusters are defined as regions of considerably higher density, wherein the density, represented by the number of data points in the neighbourhood, exceeds a predefined threshold value. Moreover, the region with relatively lower density is kept outside the cluster and denoted as outliers. Definitions used in DBSCAN are given as follows:

Definition 1: *Eps*-neighbourhood of a data point

The *Eps*-neighbourhood of a data point p is defined as follows:

$$N_{Eps}(p) = q \in D \mid dist(p, q) \leq Eps, \tag{1}$$

where q denotes another data point in the dataset D and $dist(\cdot)$ denotes the function used to compute distance between two data points.

Definition 2: Directly density-reachable

A data point p is directly density-reachable from another data point q with respect to $N_{Eps}(p)$ and *MinPts* (minimum number of data points in the *Eps* neighbourhood of that data point), if $p \in N_{Eps}(q)$ and $N_{Eps}(q) \geq MinPts$.

Definition 3: Density-reachable

A data point p is density-reachable from another data point q with respect to $N_{Eps}(p)$ and *MinPts*, if there exists a chain of data points p_1, p_2, ..., p_n, $p_1 = q$, $p_n = p$ such that $p_i + 1$ is directly density-reachable from p_i.

Definition 4: Density-connected

A data point p is density-connected to another data point q with respect to $N_{Eps}(p)$ and *MinPts*, if there exists a data point o such that both p and q are density-reachable from o with respect to $N_{Eps}(p)$ and *MinPts*.

Definition 5: Cluster

Let D be a set of data points. A cluster C with respect to $N_{Eps}(p)$ and *MinPts* is a non-empty subset of D satisfying the following two conditions:

1. $\forall$ p, q: if $p \in C$ and q is density-reachable from p with respect to $N_{Eps}(p)$ and *MinPts*, then $q \in C$ (maximality).
2. $\forall\, p, q \in C$: p is density-connected to q with respect to $N_{Eps}(p)$ and *MinPts* (connectivity).

Definition 6: Noise

Let $C1, C2, ..., Ck$ denote the clusters identified in D with respect to *Eps* and *MinPts*, then we can define the noise as the set of data points in D not belonging to any cluster Ci, $i = 1, ..., k$, i.e., noise $= p \in D \mid \forall i : p \notin Ci$.

During cluster formation, *direct density-reachability*, *density-reachability* and *density-connectivity* (see Definitions 2-4) are used by DBSCAN to characterize symmetric and asymmetric relations between core points (i.e., high-density data points within clusters) and borderline points. Based on the pre-determined density parameters *Eps* and *MinPts*, clusters are formed comprising reachable core points and their corresponding borderline points. When there are no more data points that can be further added into any cluster, DBSCAN terminates. Re-

cently, an efficient distributed clustering scheme is proposed in [28], which uses only the borderline data points identified by DBSCAN to assist the clustering across multiple datasets that may be stored in local sites.

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 the identification of outliers as low-density data points. However, DBSCAN is sensible to the user-defined parameter values and does not perform well on highly overlapped dense regions [29].

2.2. DPC: Density Peak Clustering

In a nutshell, DPC generates clusters by assigning data points to the same cluster of its nearest neighbour with higher density. Specifically, DPC uses the decision graph approach to identify cluster centroids that have the highest density. A decision graph is derived based on the following two fundamental properties of each data point x_i: i) local density ρ_i and ii) individual distance of each data point from other data points of higher density δ_i.

Assume a dataset consists of $X_{Y\times Z} = [x_1, x_2, ..., x_Y]^T$, where $x_i = [x_{1i}, x_{2i}, ..., x_{Zi}]$ denotes a vector with Z number of attributes and Y denotes the total number of data points. The distance between two data points x_i and x_j is computed as follows:

$$d(x_i, x_j) = || \, x_i - x_j \, ||, \tag{2}$$

where $|| \cdot ||$ denotes Euclidean distance.

The local density of a data point x_i, denoted as ρ_i, is then defined as:

$$\rho_i = \sum_j \chi(d(x_i, x_j) - C_d), \tag{3}$$

$$\chi(a) = \begin{cases} 1, & \text{if } a < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where C_d denotes the cutoff distance that user specified to distinguish the level of density. In DPC, the value of C_d can be autonomously determined in a

systematic way as follows:

$$C_d = D_{Y_d \times \frac{p}{100}}, \tag{5}$$

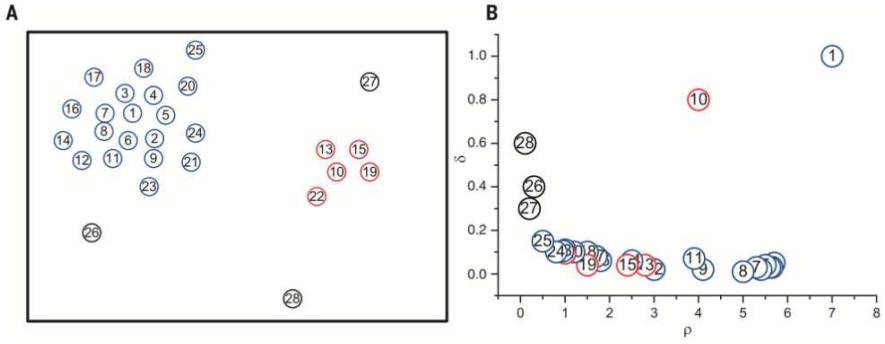
where $Y_d = \binom{Y}{2}$, $D_{Y_d \times \frac{p}{100}} \in D = [d_1, d_2, ..., d_{Y_d}]$, wherein D denotes the set of all the distances between every pair of two data points in a dataset, sorted in ascending order, and p denotes the user specified cutoff percentile.

On the other hand, δ_i is defined as the shortest distance from any other data point that has a higher density value than x_i. If x_i has the highest density value, δ_i is assigned to the longest distance to any other data point. Specifically, 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 the computation of ρ_i and δ_i for each data point in the given dataset, DPC autonomously generates a decision graph based on the computed ρ and δ values (see Figure 3) and subsequently asks the user to determine cluster centroids. As a rule of thumb, data points with higher ρ and higher δ values should be selected as cluster centroids. However, as shown previously in Figure 1 that because DPC considers all the data points during the computation of local density (see (3)), it may not perform well on overlapping clusters.

In terms of dealing with the cluster boundaries, for each cluster, DPC defines a border region, which is a set of data points that are assigned to the underlying cluster but within certain distance (i.e., C_d) from any data point belonging to another cluster. Furthermore, DPC locates the data point with the highest density within this border region of the corresponding cluster and makes use of its computed density ρ_b. Subsequently, "the points of the cluster whose density is higher than ρ_b are considered as part of the cluster core (robust assignation)" and "the others are considered as part of the cluster halo (suitable to be considered as noise)" [24]. However, according to the definition, a halo point has to be close to at least one data point belonging to another cluster. Therefore, DPC may not well handle certain low-density data points, especially



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

Figure 3: An example of DPC's decision graph (excerpted from [24]).

if they are not near to other identified clusters. As shown previously in Figure 2, the two data points in the top left corner are always considered as part of a larger cluster regardless of different C_d values in use.

3. REDPC: Residual Error-based Density Peak Clustering Algorithm

To better deal with overlapping clusters and low-density data points, we propose Residual Error-based Density Peak Clustering (REDPC) algorithm. Specifically, we learn from DPC in using decision graphs to identify cluster centroids, learn from DBSCAN in determining density connectivities within a neighbourhood, and learn from the residual error theory in measuring density.

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

1. *Preprocessing*: Firstly, the residual errors of individual data points are computed as local density measurement and then δ (see the following subsection) is computed as distance measurement.
2. *Initial assignment*: Secondly, the decision graph is generated, cluster centroids are identified, and data points are intermediately assigned to the respective clusters.

3. *Halo identification*: halo points (consists of both borderline points and anomalies) are identified.
4. *Anomaly refinement*: Finally, anomalies are isolated from *halo points* and further processed before presenting the final clustering result.

3.1. Preprocessing

To construct better decision graphs for more distinguishable cluster centroid identifications, we adopt the residual error computation to measure the density of each data point within its neighbourhood region. Specifically, the residual error e_{ij} between data point x_i to its neighbour x_j is computed as follows:

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

where N is a user-defined constant parameter denoting the neighbourhood size. Specifically, N is an integer used to find N number of the nearest neighbours of x_i, wherein Euclidean distance is used as the same as DPC (see (2)). Furthermore, the residual error of x_i can be computed as follows:

$$e_i = \sum_j e_{ij} = \sum_j \frac{|| \, x_i - x_j \, ||}{N}. \tag{8}$$

Comparing (8) to (3), it is obvious that by adopting the residual error computation, when measuring the local density, REDPC only takes the data points within the neighbourhood into consideration. In contrast, DPC takes all the data points in the entire dataset into consideration. By only considering the local regional density, REDPC is capable of generating better decision graphs for cluster centroid identifications (see Section 4).

Moreover, we use δ_i to denote the minimum distance of data point x_i to another data point with lower residual error. Specifically, δ_i is computed as follows:

$$\delta_i = \begin{cases} \min\limits_{j:e_j<e_i} || \, x_i - x_j \, ||, & \text{if } \exists \, j \text{ s.t. } e_j < e_i, \\ \max\limits_{j} || \, x_i - x_j \, ||, & \text{otherwise.} \end{cases} \tag{9}$$

The dynamics of the preprocessing procedures in REDPC are summarized in Algorithm 1.

Algorithm 1 The preprocessing procedures in REDPC

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

Output: Euclidean distance matrix DM of size $n * n$, residual error vector e, *sortd_e* (e sorted in ascending order), minimum distance vector δ, and index vector of the nearest neighbour 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 neighbours 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*;
compute δ (see (8) and (9));
obtain *NNeigh* based on *sortd_e*;

3.2. Initial Assignment

After preprocessing, REDPC then generates a decision graph based on e_i (see (8)) and δ_i (see (9)) computed for each data point x_i in the underlying dataset (see Figure 7(a) in Section 4). As a rule of thumb, data points with lower e_i values and higher δ_i values should be selected as cluster centroids. After the identification of cluster centroids, based on the shortest distance between any unassigned data point x_j to any data point x_i that has been assigned with its corresponding cluster label, assign x_j to the same cluster as x_i. Repeat this assignment procedure until all the data points have been assigned with certain cluster labels. As such, we obtain the initial assignments of all data points.

The dynamics of initial assignment procedures in REDPC are summarized in Algorithm 2.

Algorithm 2 The initial assignment procedures in REDPC

Input: user identified cluster centroids CC, *sortd_e* and *NNeigh* obtained from Algorithm 1

Output: cluster labels assigned to all the data points Cl

assign cluster labels to all cluster centroids in CC to obtain initial Cl;
while there still exists a data point with no cluster label assigned **do**
based on Cl, *sortd_e* and *NNeigh*, find data point x_i to be assigned;
assign x_i with the cluster label of its nearest neighbour;
update Cl accordingly;
end while

3.3. Halo Identification

After the initial assignment of cluster labels, we first identify the halo points from each intermediate cluster for later anomaly detections (in the final stage). To identify the halo points, we need to determine the value of the cutoff parameter C_d. In REDPC, C_d is defined as the same as that in DPC (see (5)), which is systematically determined based on the user specified cutoff percentile p.

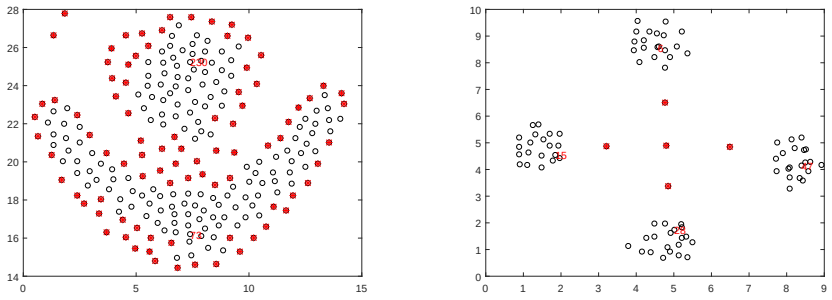
Subsequently, for each data point x_i in cluster A, if there exists a data point x_j in cluster B and $B \neq A$ that the distance between x_i and x_j is less than the cutoff threshold, i.e.,

$$|| x_i - x_j || < C_d, \tag{10}$$

then we compute the mean of the residual values of the two data points as follows:

$$mean\_e_{ij} = \frac{1}{2}(e_i + e_j). \tag{11}$$

For each identified cluster K, we use a variable named residual threshold rt_K to denote the threshold for the residual value of each corresponding cluster.



(a) Flame (b) D2 (see Section 4)

Figure 4: Illustrations of identified halo points, represented with asterisks in red color.

The value of rt_K is initialized to a large number. If x_i is found in cluster A, x_j is found in cluster B, and they fulfil the inequality defined in (10), then the residual thresholds for both clusters are updated as follows:

$$rt_K = mean\_e_{ij}, \text{ if } rt_K > mean\_e_{ij}, K = A, B. \tag{12}$$

After updating rt for all the clusters, we can then determine the set of halo points in each cluster, denoted as *haloset*, by adopting the following criterion:

$$haloset_K = haloset_K \cup x_i, \ \textit{iff}\ e_i > rt_K. \tag{13}$$

The dynamics of halo identification procedures in REDPC are summarized in Algorithm 3. The exact mechanisms adopted can be simply described as using borderline data points to determine the level of overlap between adjacent clusters, and subsequently identify the halo points in the border regions for further identification of possible anomalies. As illustrated in Figure 4, halo points are well identified by REDPC around the cluster borders.

3.4. Anamaly Refinement

It is of great importance to distinguish the anomalies from other normal data points in the identified clusters because anomalies highly likely represent the abnormal patterns or malicious activities in real-world scenarios. For example, unusual road traffic patterns may suggest nearby accidents or emergencies, unusual

Algorithm 3 The halo identification procedures in REDPC

Input: dataset D, Cl obtained from Algorithm 2, C_d computed according to (5) and e_{ij} obtained from Algorithm 1

Output: the set of halo points *haloset*

```
initialize residual threshold rt to a large value for each cluster in Cl;
for each data point x_i in D do
  for each data point x_j in D, i ≠ j && Cl(i) ≠ Cl(j) do
    if distance between x_i and x_j is within C_d (see 10) then
      compute the mean residual mean_e_ij (see 11);
      update rt_Cl(i) and rt_Cl(j) if necessary (see 12);
    end if
  end for
end for
initialize haloset to φ;
for each data point x_i in D do
  if halo point identification criterion is met (see 13) then
    update haloset accordingly;
  end if
end for
```

credit card transactions may indicate identity theft, unusual computer network loads should alert the cyber security division, etc. Therefore, in REDPC, we further detect the anomalies and highlight them during visualization.

During anomaly detection, a halo point with high residual error and low δ value is recognized as an anomaly and highlighted with a special symbol for clearer graphical representation (e.g., see Figure 8(f) in Section 4.2). The threshold values used to distinguish e and δ are heuristically determined. Specifically, the threshold for residual error th_e is defined as

$$th_e = mean(e_i) + \frac{1}{2}(min(e_i) + max(e_i)) \tag{14}$$

and the threshold th_δ is defined as

$$th_\delta = \frac{1}{2}(min(\delta_i) + max(\delta_i)). \tag{15}$$

Therefore, after the determination of th_e and th_δ values, we can then identify the set of anomalies in the dataset, denoted as *anoset*, by adopting the following criterion:

$$anoset = anoset \cup x_i, \ \textit{iff}\ e_i > th_e \ \&\& \ \delta_i < th_\delta. \tag{16}$$

Moreover, if *anoset* is non-empty, we further investigate the most possible cluster label of each detected anomaly. This refinement is also intuitively designed that for each anomaly in *anoset*, we first find its nearest neighbours with the neighbourhood size as the same as N defined in (7). Furthermore, if within the neighbourhood of an anomaly, there exist other anomalies, we then remove these anomalies from the neighbour because their cluster labels are not yet properly assigned. At last, we assign the cluster label of each anomaly to the majority cluster label in its neighbourhood (if the majority ties, we assign the cluster label of the nearest data point belonging to any of the tying clusters). Due to the incorporation of such refinement of the identified anomalies, not only the clustering results may be improved, but also these anomalies are highlighted visually for human inspections (see Section 4).

The dynamics of anomaly refinement procedures in REDPC are summarized in Algorithm 4).

3.5. Overall REDPC Dynamics and Its Computational Complexity

The dynamics of REDPC is depicted in Figure 5, wherein the information flow among the underlying dataset, user inputs, and the REDPC algorithms are explicitly shown. Moreover, the computational complexity of REDPC is shown in Table 1, wherein n denotes the number of data points in the underlying dataset and m denotes the number of halo points obtained from Algorithm 3. In normal circumstances, $m \ll n$.

Comparing to other clustering algorithms benchmarked in this paper, REDPC has a middle level of computational complexity as shown in Table 2,

Algorithm 4 The anomaly refinement procedures in REDPC

Input: DM, e and δ obtained from Algorithm 1, Cl obtained from Algorithm 2, *haloset* obtained from Algorithm 3 and user specified neighbourhood size N

Output: the set of anamalies *anoset* and refined Cl

initialize *anoset* to ϕ;
compute thresholds th_e and th_δ (see (14) and (15));
for each data point x_i in *haloset* **do**
  if x_i fulfils the anomaly criterion **then**
    append x_i to *anoset* (see (16));
  end if
end for
for each data point x_j in *anoset* **do**
  find the neighbours 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

Table 1: Computational complexity of REDPC

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

wherein I denotes the number of iterations and K denotes the user predefined number of clusters. Moreover, in the following experiment section, we also compare the computational time taken by all the clustering algorithms.

4. Experiments

To evaluate the performance of REDPC, we apply it on four UCI datasets, namely *Iris*, *Seeds*, *Wine* and *Thyroid*, three widely used synthetic datasets,

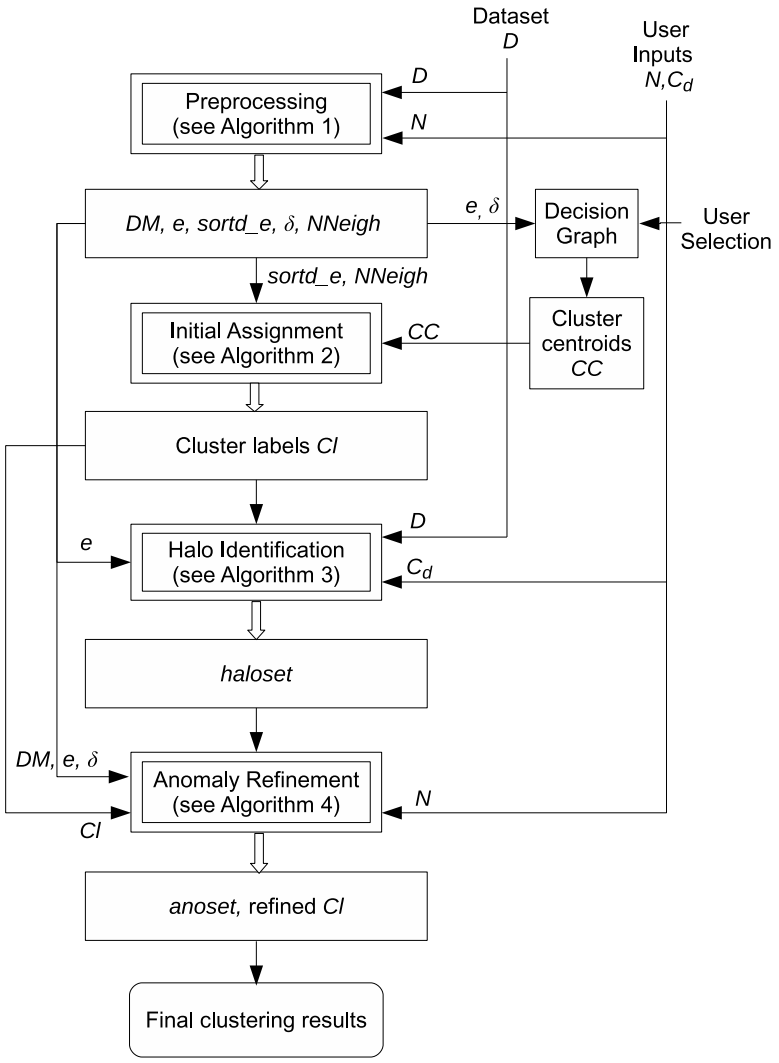


Figure 5: Dynamics of the overall REDPC algorithm.

Table 2: Computational complexity comparisons

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

Table 3: Properties of the UCI and synthetic datasets

Datasets	Properties		
	# Points	*# Dimensions*	*# Clusters*
Iris	150	4	3
Seeds	210	7	3
Wine	178	13	3
Thyroid	215	5	3
Flame	240	2	2
Aggregation	788	2	7
Spiral	312	2	3
D1	87	2	3
D2	85	2	4

namely *Flame*, *Aggression* and *Spiral*, and two own-defined datasets *D1*[1] and *D2*[2]. The properties of these nine datasets are listed in Table 3. Moreover, we conduct experiments using K-Means [27] (to minimize the difference caused by randomness, averaged results of 10 independent runs are reported), AP [18], DBSCAN [23] and DPC [24] on the same datasets for comparisons.

In this paper, we use F-score to measure the accuracy of the clustering results. The performance comparisons among all the benchmarking models are reported in Table 4 and visualized in Figure 6. It is encouraging to find that REDPC achieves the highest F-score on eight out of nine datasets. Although REDPC only achieves the second best on *Aggregation*, the difference between the winner (DPC) and REDPC's result is as small as 1 - 0.9983 = 0.0017 or 0.17%. When we further examine the difference in terms of the number of correctly labelled data points, we find that the difference between DPC and REDPC is as small as 1 (out of the total number of 788 data points). As such,

[1]The *D1* dataset (with cluster labels) is available online: `https://www.dropbox.com/s/f3ynvml53i2500u/D1_with_label.csv?dl=0`

[2]The *D2* dataset (with cluster labels) is available online: `https://www.dropbox.com/s/899xltgq3gg09bg/D2_with_label.csv?dl=0`

Table 4: Performance Comparsion

Datasets	K-Means	AP	DBSCAN	DPC	REDPC
Iris	0.8149	0.3924	0.7462	**0.8404**	**0.8404**
Seeds	0.8091	0.2833	0.6137	0.8025	**0.8106**
Wine	0.5896	0.2521	0.5052	0.5699	**0.5900**
Thyroid	0.7251	0.1814	0.6933	0.7545	**0.7890**
Flame	0.7432	0.1538	0.8833	**1**	**1**
Aggregation	0.7890	0.2117	0.8885	**1**	0.9983
Spiral	0.3278	0.1505	**1**	**1**	**1**
D1	0.8352	0.5445	**1**	**1**	**1**
D2	0.9976	0.9332	0.9332	0.9756	**1**

although DPC achieves slightly better performance than REDPC in terms of clustering accuracy, this small amount of difference may not be significant. In the following subsections, we elaborate on the performance of REDPC in various aspects, respectively.

In addition, the computational time spent by each algorithm (average of 10 runs for all models) is shown in Table 5. The comparison results are consistent with Table 2 that DBSCAN with the lowest computational complexity achieves the shortest computational time and REDPC with a middle level of computational complexity achieves a middle level of computational time. Comparing to DPC, although they both have the same computational complexity of $O(n^2)$, REDPC is always approximately 20 ms slower. This is mainly due to the additional refinement procedures taken by REDPC to better handle the anomalies (see Algorithms 3 and 4). However, this compensation on computational time greatly improves REDPC's performance (see Table 4). Note that all the clustering algorithms were implemented using MATLAB and the experiments were conducted using the same 64-bit computer installed with Intel(R) Core(TM) i3-4160 CPU at 3.60 GHz and 8 GB RAM.

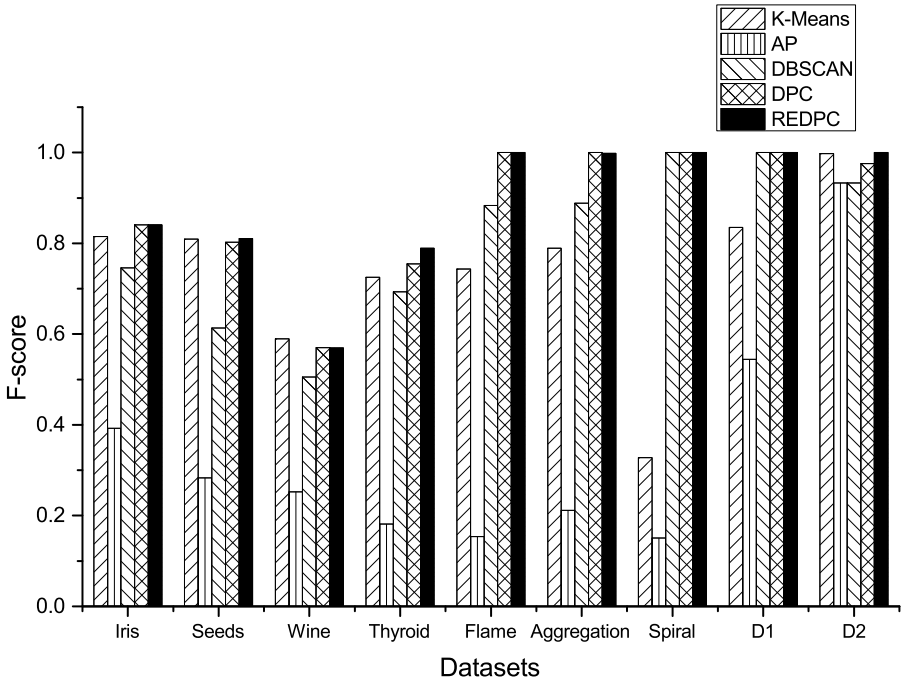
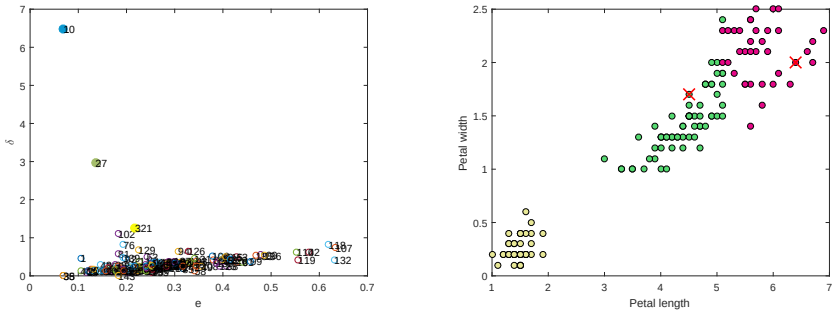


Figure 6: Visualization of performance comparison on nine datasets.

Table 5: Comparisons on computational time spent (in ms)

Datasets	K-Means	AP	DBSCAN	DPC	REDPC
Iris	148.2681	67.6571	1.9672	70.6112	93.5109
Seeds	148.0411	95.5450	1.8822	71.4411	82.6917
Wine	156.5255	70.2968	1.4468	69.4427	90.1444
Thyroid	149.1138	5551.4389	3.3243	70.8240	102.1959
Flame	150.5154	119.2546	1.7628	72.8839	92.1913
Aggregation	174.6097	1853.7493	10.2566	123.0901	148.9328
Spiral	160.1743	180.8485	3.1349	89.5048	105.0130
D1	149.6506	34.3190	1.1136	67.6673	90.7307
D2	146.9794	32.2693	0.8710	68.8392	92.7008



(a) *Iris* decision graph by REDPC (b) Cluster formation by REDPC

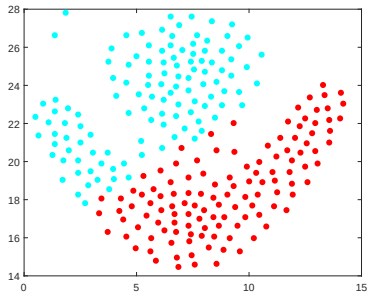
Figure 7: Determination of cluster centroids and the resulting cluster formation based on the decision graph generated by REDPC on *Iris* dataset.

4.1. Identifying Cluster Centroids

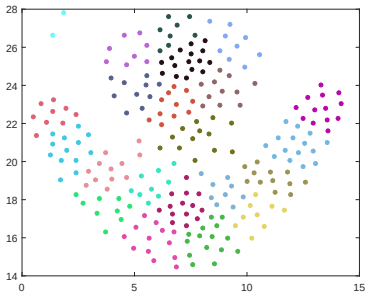
As aforementioned that the performance of DPC is sometimes limited by its generated decision graph. In comparison, REDPC employs the residual error computation to measure the local density within a neighbourhood region. As a result, REDPC generates decision graphs that are better suited for cluster centroid identifications. As shown in Figure 7(a), the third cluster centroid in the *Iris* dataset is relatively more identifiable in the decision graph generated by REDPC comparing to that generated by DPC (see Figure 1).

4.2. Identifying Clusters with Anomalies

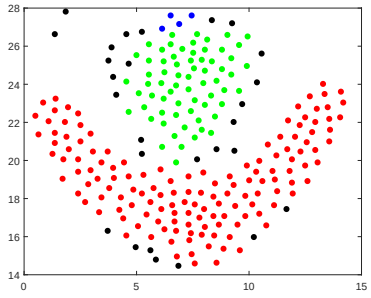
To illustrate the capability of REDPC in anomaly detection, we present the clustering results of all clustering methods on the *Flame* and *D2* datasets in Figures 8 and 9, respectively. The two anomalies in *Flame* are located in the top left corner and the five anomalies in *D2* are located in the center. It is clearly shown that K-Means and AP fail in identifying anomalous data points. DBSCAN [23] adopts *MinPts* and *density-connectivity* to detect outliers, but along with outliers, it detects much more borderline data points incorrectly on *Flame* dataset. DPC does not perform anomaly detections as aforementioned.



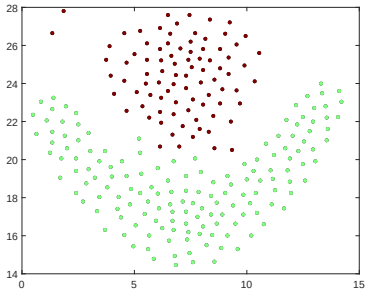
(a) K-Means, $K = 2$



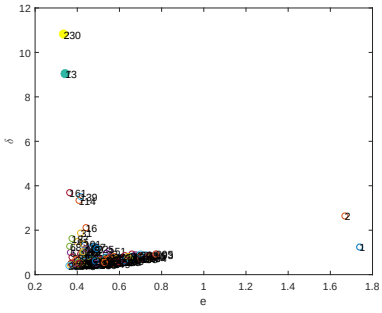
(b) AP, *preference* = -5.9261



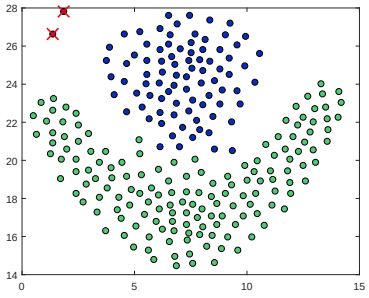
(c) DBSCAN, $Eps = 0.85$, $MinPts = 5$



(d) DPC, $C_d = 8\%$



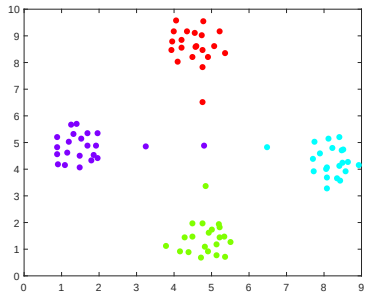
(e) *Flame* decision graph by REDPC



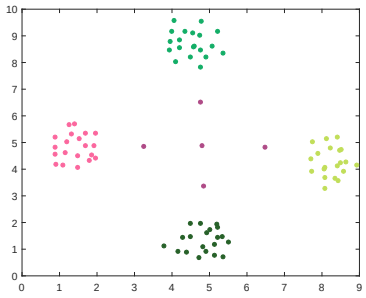
(f) REDPC, $N = 4$, $C_d = 1\%$ (#clusters = 2, #anomalies = 2)

Figure 8: Cluster formation results on *Flame* dataset.

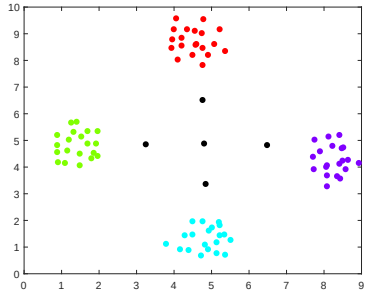
On the other hand, as shown in Figures 8(f) and 9(f), REDPC consistently detects the correct possible anomalies.



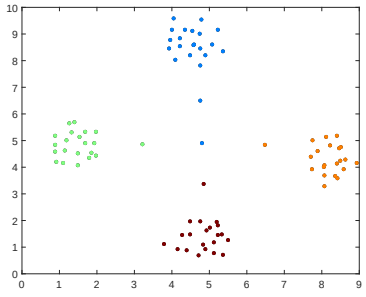
(a) K-Means, $K = 4$



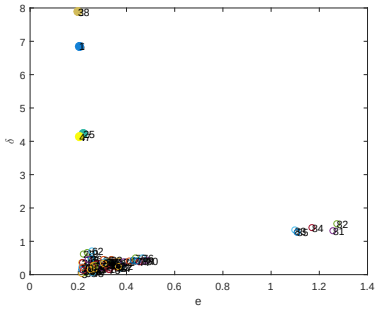
(b) AP, *preference* = -4.8674



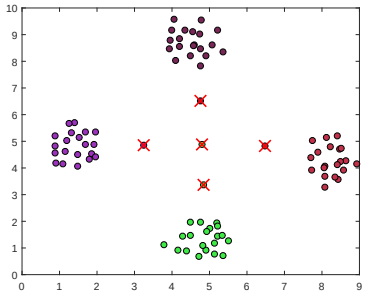
(c) DBSCAN, $Eps = 1$, $MinPts = 5$



(d) DPC, $C_d = 5\%$



(e) *D2* decision graph by REDPC



(f) REDPC $N = 5$, $C_d = 1\%$ (#clusters = 4, #anomalies = 5)

Figure 9: Cluster formation results on *D2* dataset.

4.3. Identifying Clusters of Varying Sizes

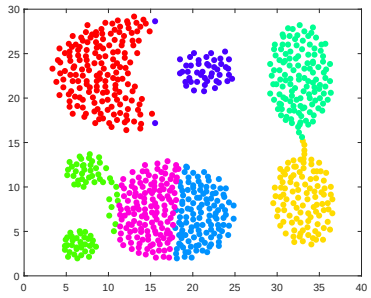
To illustrate the performance of REDPC on datasets comprising of clusters of varying sizes, such as *Aggregation* and *D1*, we refer you to Figures 10 and 11, respectively. It is clearly shown in these two figures that both K-means and AP do not perform well on these datasets. The performance of DBSCAN is much better, however, it still faces problems in a small set of borderline points (see Figure 10(c)). In comparison, DPC correctly identifies all the clusters of varying sizes and REDPC only mislabels one data point.

4.4. Identifying Clusters of Irregular Shapes

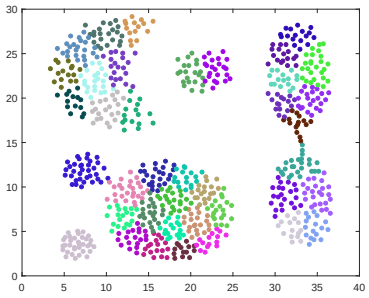
Density-based clustering algorithms have been shown to perform well in identifying arbitrary-shaped clusters in the literature. In this subsection, we use Figures 8, 10 and 12 to show REDPC's performance in identifying irregular shaped clusters. K-means and AP are partition based clustering methods and hence cannot well handle such datasets. DBSCAN performs perfectly in *Spiral* (see Figure 12(c)), however, not so well in the other datasets. In comparison, both DPC and REDPC are shown to be capable of correctly identifying the natural clusters of irregular shapes.

4.5. Identifying Clusters of Different Densities

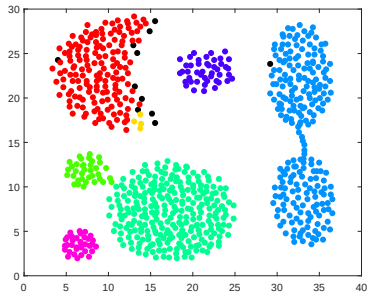
Among all the datasets used in this paper, our own-defined synthetic dataset *D1* best illustrates the scenario of having clusters of different densities (see Figure 11). It is clearly shown in the figure that K-Means and AP do not well handle a dataset of varying densities. DBSCAN only detects two clusters and deems the third cluster as a set of outliers (i.e., the top cluster in Figure 11(c) only consists of outliers, which is due to the relatively low density of those data points that they do not fulfil the density-reachability definition, see Section 2.1). In comparison, both DPC and REDPC correctly identify all the natural clusters (the two anomalies reported by REDPC are assigned to the correct cluster after refinement, see Algorithm 4).



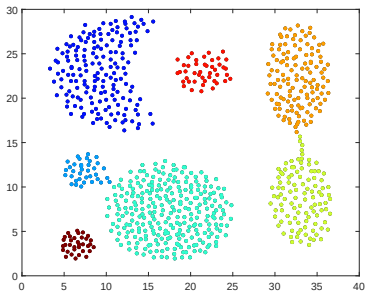
(a) K-Means, $K = 7$



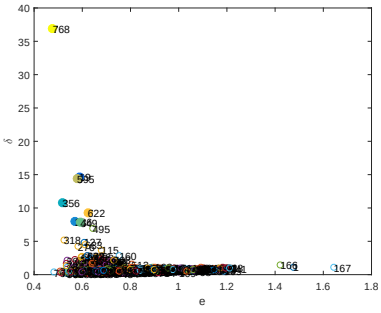
(b) AP, *preference* = -16.5170



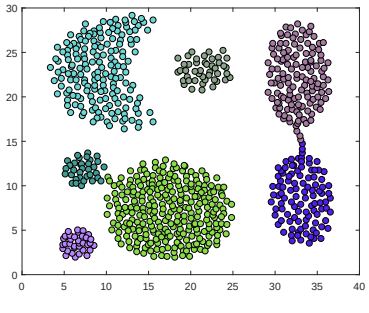
(c) DBSCAN, $Eps = 1.1$, $MinPts = 5$



(d) DPC, $C_d = 5\%$

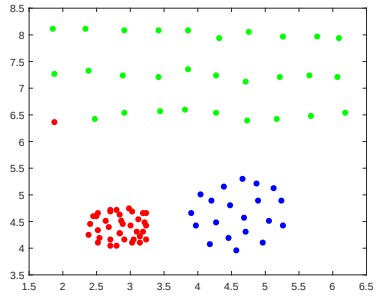


(e) *Aggregation* decision graph by REDPC

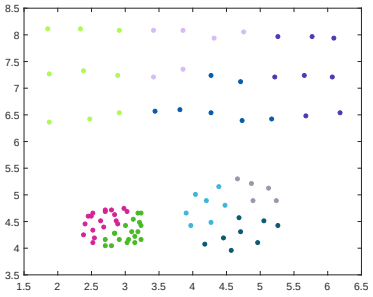


(f) REDPC, $N = 7$, $C_d = 1\%$ (#clusters = 7)

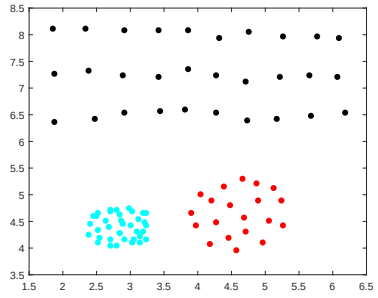
Figure 10: Cluster formation results on *Aggregation* dataset.



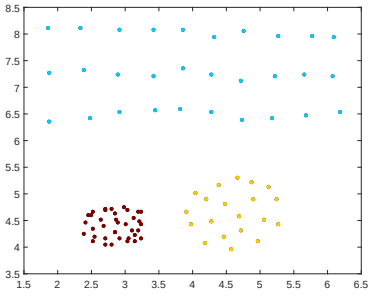
(a) K-Means, $K = 3$



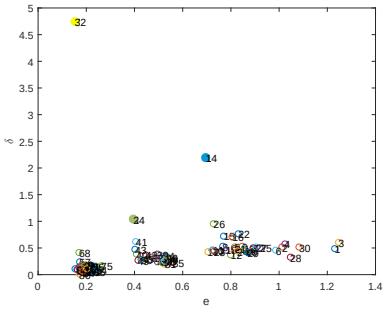
(b) AP, *preference* = -2.1523



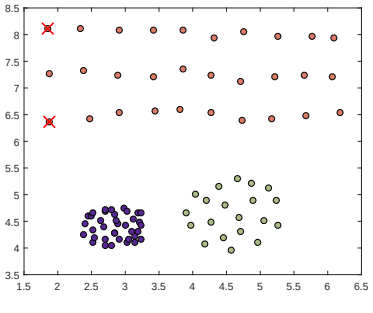
(c) DBSCAN, $Eps = 0.5$, $MinPts = 5$



(d) DPC, $C_d = 20\%$



(e) *D1* decision graph by REDPC

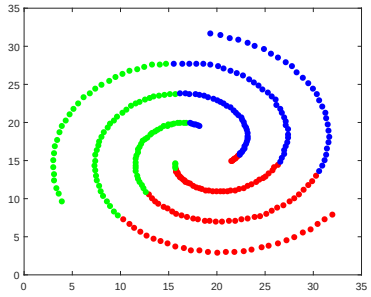


(f) REDPC $N = 11$, $C_d = 1\%$ (#clusters = 3, #anomalies = 2)

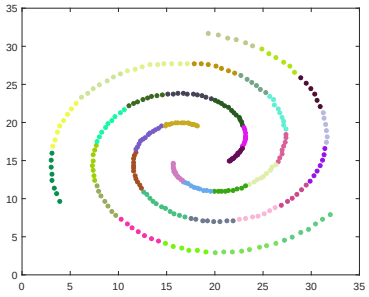
Figure 11: Cluster formation results on *D1* dataset.

4.6. Overall Comparison of All Benchmarking Models

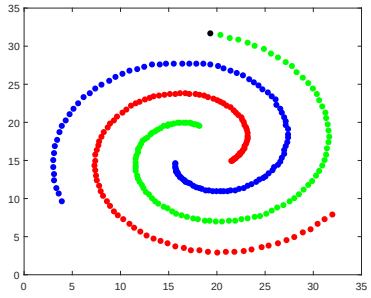
Our proposed REDPC algorithm has shown to perform better than the other benchmarking models in identifying clusters of various properties in the previous



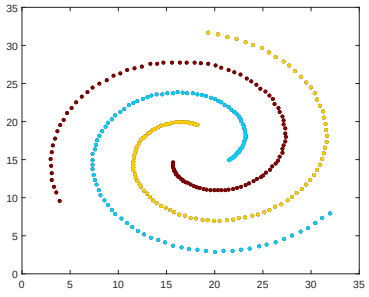
(a) K-Means, $K = 3$



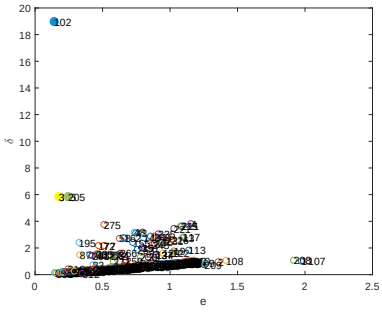
(b) AP, *preference* = -12.3868



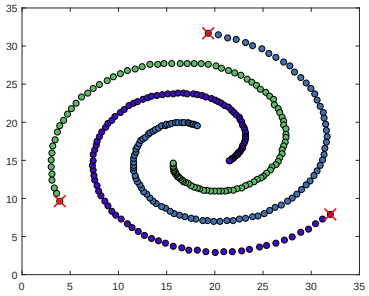
(c) DBSCAN, $Eps = 2.5$, $MinPts = 5$



(d) DPC, $C_d = 5\%$



(e) *Spiral* decision graph by REDPC



(f) REDPC, $N_i = 5$, $C_d = 1\%$ (#clusters = 3, #anomalies = 3)

Figure 12: Cluster formation results on *Spiral* dataset.

subsections, respectively. In this subsection, we summarize the performance of all clustering algorithms on each property of the clusters in Table 6. As shown

Table 6: Overall Performance Comparison on Different Cluster Properties

Property	Algorithm				
	K-Means	AP	DBSCAN	DPC	REDPC
Centroid detection	$\surd$	$\times$	∂	$\surd$	$\surd$
Anomaly detection	∂	$\times$	$\surd$	$\surd$	$\surd$
Variable sizes	∂	$\times$	$\surd$	$\surd$	$\surd$
Variable shapes	$\times$	$\times$	$\surd$	$\surd$	$\surd$
Variable densities	$\surd$	$\times$	$\surd$	$\surd$	$\surd$

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 4.1 to 4.5). 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.

in the table, it is fair to say that REDPC is a well-designed algorithm working well in various performance evaluation aspects.

Another note on the comparison between DPC and REDPC is that although REDPC uses one more parameter than DPC does, which is the neighbourhood size N (they both use the distance cutoff parameter C_d), in practice, we spend almost equally amount of effort in determining the best performing parameter values for both methods. The reason is because for REDPC, we always fix the value of C_d at 1% and tune N in an incremental manner. On the other hand, in order to get DPC's best clustering accuracy, we need to tune C_d with varying values. Therefore, in terms of conducting the experiments shown in this paper, both methods require the tuning of only one parameter.

5. Conclusion

In this paper, we propose the Residual Error-based Density Peak Clustering (REDPC) algorithm by using residual error computation to measure the local density within a neighbourhood region and further process the identified low-

density data points. As such, REDPC may generate better decision graphs for cluster centroid identifications and better identify the possible anomalies. The experimental results on both synthetic and real-world UCI datasets demonstrate that REDPC performs better than DPC and other algorithms.

Going forward, we will improve REDPC for better autonomy in parameter value determination and apply it to more complex and challenging datasets.

Acknowledgements

This research is supported in part by the the National Science Fund Project of China No. 61772227 and Science & Technology Development Foundation of Jilin Province under the grant No. 20160101259JC, 20180201045GX. This research is also supported in part by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

References

[1] D. Wang, C. Quek, G. S. Ng, MS-TSKfnn: Novel Takagi-Sugeno-Kang fuzzy neural network using ART like clustering, in: Proceedings of International Joint Conference on Neural Networks, 2004, pp. 2361–2366.

[2] J. Wen, D. Zhang, Y. Cheung, H. Liu, X. You, A batch rival penalized expectation-maximization algorithm for gaussian mixture clustering with automatic model selection, Computational and Mathematical Methods in Medicine (2012) Article ID: 425730.

[3] Z. Li, J. Liu, Y. Yang, X. Zhou, H. Lu, Clustering-guided sparse structural learning for unsupervised feature selection, IEEE Transactions on Knowledge and Data Engineering 26 (9) (2014) 2138–2150.

[4] J. Zhou, L. Chen, C. P. Chen, Y. Zhang, H.-X. Li, Fuzzy clustering with the entropy of attribute weights, Neurocomputing 198 (2016) 125–134.

[5] J. Yu, R. Hong, M. Wang, J. You, Image clustering based on sparse patch alignment framework, Pattern Recognition 47 (11) (2014) 3512–3519.

[6] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Transactions on Image Processing 24 (12) (2015) 5659–5670.

[7] C. Xu, Z. Su, Identification of cell types from single-cell transcriptomes using a novel clustering method, Bioinformatics 31 (12) (2015) 1974–1980.

[8] D. Wang, A.-H. Tan, Self-regulated incremental clustering with focused preferences, in: Proceedings of International Joint Conference on Neural Networks, 2016, pp. 1297–1304.

[9] D. Wang, C. Quek, G. S. Ng, Ovarian cancer diagnosis using a hybrid intelligent system with simple yet convincing rules, Applied Soft Computing 20 (2014) 25–39.

[10] A. Karami, M. Guerrero-Zapata, A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks, Neurocomputing 149 (2015) 1253–1269.

[11] G. Kou, Y. Peng, G. Wang, Evaluation of clustering algorithms for financial risk analysis using MCDM methods, Information Sciences 275 (2014) 1–12.

[12] D. Wang, C. Quek, G. S. Ng, Bank failure prediction using an accurate and interpretable neural fuzzy inference system, AI Communications 29 (4) (2016) 477–495.

[13] 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 Transactions on Emerging Topics in Computing 2 (3) (2014) 267–279.

[14] A. Chaudhary, J. L. Raheja, K. Das, S. Raheja, Intelligent approaches to interact with machines using hand gesture recognition in natural way: A survey, arXiv preprint arXiv:1303.2292.

[15] C.-W. Tsai, C.-F. Lai, M.-C. Chiang, L. T. Yang, Data mining for Internet of Things: A survey., IEEE Communications Surveys and Tutorials 16 (1) (2014) 77–97.

[16] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. A. C. Coello, Survey of multiobjective evolutionary algorithms for data mining: Part II, IEEE Transactions on Evolutionary Computation 18 (1) (2014) 20–35.

[17] H. Chen, W. Wang, X. Feng, R. He, Discriminative and coherent subspace clustering, Neurocomputing 284 (2018) 177–186.

[18] B. J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.

[19] R. Mehmood, G. Zhang, R. Bie, H. Dawood, H. Ahmad, Clustering by fast search and find of density peaks via heat diffusion, Neurocomputing 208 (2016) 210–217.

[20] X. Li, Q. Han, B. Qiu, A clustering algorithm using skewness-based boundary detection, Neurocomputing 275 (2018) 618–626.

[21] T. Chen, N. L. Zhang, T. Liu, K. M. Poon, Y. Wang, Model-based multidimensional clustering of categorical data, Artificial Intelligence 176 (1) (2012) 2246–2269.

[22] M. Parikh, T. Varma, Survey on different grid based clustering algorithms, International Journal of Advance Research in Computer Science and Management Studies 2 (2) (2014) 427–430.

[23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[24] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[25] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computing Surveys 41 (3), (2009) Article 15.

[26] M. Wang, W. Zuo, Y. Wang, An improved density peaks-based clustering method for social circle discovery in social networks, Neurocomputing 179 (2016) 219–227.

[27] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the Royal Statistical Society. Series C (Applied Statistics) 28 (1) (1979) 100–108.

[28] Q. Tong, X. Li, B. Yuan, Efficient distributed clustering using boundary information, Neurocomputing 275 (2018) 2355–2366.

[29] Y. Lv, T. Ma, M. Tang, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, An efficient and scalable density-based clustering algorithm for datasets with complex structures, Neurocomputing 171 (2016) 9–22.