AIMS Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 10(4): 9748–9772.
DOI: 10.3934/math.2025447
Received: 11 March 2025
Revised: 09 April 2025
Accepted: 17 April 2025
Published: 27 April 2025

Research article

INSDPC: A density peaks clustering algorithm based on interactive neighbors similarity

Shihu Liu[1,3], **Yirong He**[1,*], **Xiyang Yang**[2,3], **and Zhiqiang Yu**[1]

[1] School of Mathematics and Computer Science, Yunnan Minzu University, Kunming 650504, China
[2] School of Mathematics and Computer Science, Quanzhou Normal University, Quanzhou 362000, China
[3] Fujian Provincial Key Laboratory of Data-Intensive Computing, Quanzhou Normal University, Quanzhou 362000, China

* **Correspondence:** Email: heyr1004@163.com.

Abstract: The density peaks clustering (DPC) algorithm has gained significant attention in various fields due to its simplicity and effectiveness. However, its performance is constrained by the local density calculation method and the selection of the cutoff distance d_c, which is a parameter primarily dependent on global data distribution, while neglecting local characteristics. Additionally, the one-step assignment strategy in DPC is prone to chain errors caused by single-point misassignment, adversely affecting clustering performance. To address these limitations, this paper proposes the interactive neighbors similarity-based density peaks clustering (INSDPC) algorithm. The algorithm introduces an interactive neighbors similarity measure that combines the information of interactive neighbors and shared neighbors to redefine local density. Furthermore, a two-step assignment strategy, leveraging interactive neighbors similarity and neighborhood information, is designed to avoid further errors when a point is incorrectly assigned. Experimental results on synthetic and real-world datasets demonstrate that INSDPC improves cluster centers identification and enhances clustering precision.

Keywords: density peaks clustering; interactive neighbors; shared neighbors; neighborhood information
Mathematics Subject Classification: 62H30

1. Introduction

Cluster analysis, a pivotal branch of data mining, plays a significant role in statistical analysis and is one of the core methods in unsupervised machine learning [1]. Clustering operates by evaluating the similarity between points, dividing the dataset into multiple categories to maximize intra-cluster similarity and minimize inter-cluster similarity [2]. More importantly, cluster analysis reveals underlying patterns and trends within a dataset, thus supporting a deeper understanding of the data. This technique has been extensively utilized across multiple domains, including community detection [3], market research [4], document clustering [5], and gene expression analysis [6].

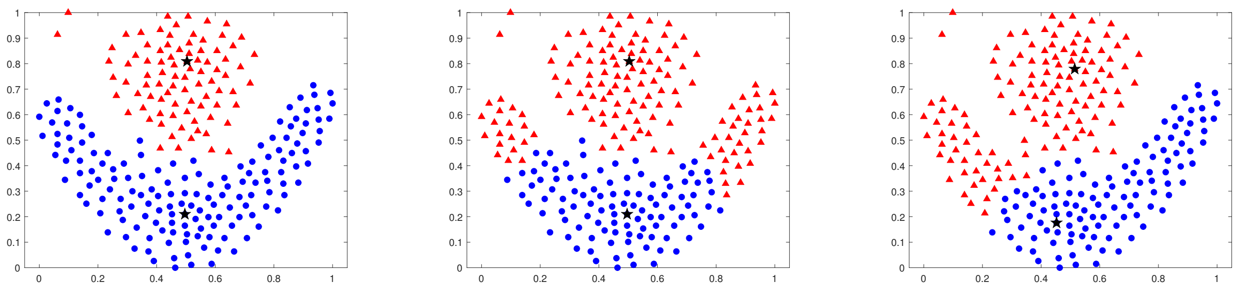
Over the past few decades, the field of clustering has undergone rapid development, with algorithms generally categorized into partition-based, hierarchical, grid-based, model-based, and density-based clustering methods. Partition-based clustering algorithms, such as K-means, aims to minimize the distance between points within a cluster and their centroid, but is highly sensitive to center initialization. To address this limitation, refined variants such as K-means++ [7] and K-medoids [8] have been proposed. Hierarchical clustering algorithms, such as clustering using representatives (CURE) [9] and balanced iterative reducing and clustering using hierarchies (BIRCH) [10], perform effectively but are constrained by high computational complexity. Grid-based clustering algorithms, such as statistical information grid (STING) [11], partition the feature space into grid cells, with the effectiveness of the algorithm strongly influenced by the choice of grid size. The expectation-maximization (EM) algorithm [12], a model-based clustering algorithm, utilizes maximum likelihood estimation to determine the probabilistic model parameters of the data.

Density-based clustering algorithms are widely applicable due to their ability to detect clusters of arbitrary shapes. Among them, density-based spatial clustering of applications with noise (DBSCAN) [13] is a classic algorithm known for its effectiveness in handling irregularly shaped clusters. However, its performance heavily depends on two key parameters: the neighborhood radius (Eps) and the minimum number of points within a neighborhood (MinPts). The lack of theoretical guidance for parameter selection makes clustering results highly sensitive to these settings. Consequently, determining optimal parameter values is critical for optimizing the algorithm's performance.

A novel density-based algorithm, named clustering by fast search and find of density peaks (DPC), was proposed by Rodriguez and Laio in 2014 [14]. Unlike traditional clustering algorithms, DPC requires only a single parameter d_c and eliminates the need for iterative optimization of an objective function to complete clustering. By calculating the local density and relative distance, DPC efficiently identifies cluster centers and assigns points to clusters. This method demonstrates exceptional clustering performance when handling data of various forms and achieves high computational efficiency [15]. As a result, DPC has been widely applied in various domains, including image recognition, computer vision, and scheduling [16].

Although the DPC algorithm offers notable advantages, it also has certain limitations. The computation of local density relies heavily on the choice of the cutoff distance d_c [17]. If d_c is too large, the local densities of most points converge, potentially classifying all points into a single cluster. Conversely, an overly small d_c can result in numerous clusters, incorrectly partitioning points that belong to the same cluster. However, the selection of d_c depends on the global distribution of the dataset, ignoring local data characteristics. Consequently, the clustering performance of the DPC algorithm might not be optimal when applied to datasets with cross-winding structures or uneven density distributions. Additionally, its one-step assignment strategy means that if a point is misassigned, the subsequent assignment process may lead to more errors.

Figure 1 illustrates these issues. In the Flame dataset, Figure 1(a,b) both apply the Gaussian kernel function for local density estimation, though with different cutoff distances, and Figure 1(a) successfully achieves perfect clustering. When Figure 1(b,c) use the same cutoff distance but different kernel functions, the clustering results become completely inconsistent. It can be observed that the clustering results largely depend on the method used to compute local density and the choice of the cutoff distance. Moreover, in Figure 1(b), parts of the upper left and upper right corners are incorrectly assigned to the closer red cluster, further emphasizing the limitations of the DPC algorithm's assignment strategy.



(a) Gaussian kernel and $d_c = 2.7\%$ **(b)** Gaussian kernel and $d_c = 2\%$ **(c)** cutoff kernel and $d_c = 2\%$

Figure 1. Clustering results on the Flame dataset using different methods and cutoff distances.

The local density computation method in the DPC algorithm exhibits limitations when addressing data with diverse distributions [18]. To overcome these challenges, researchers have proposed various enhancements. Tong et al. [19] utilized continuous functions to effectively differentiate the density of points. Li et al. [20] introduced a fuzzy semantic unit model, reformulating the density estimation problem into an optimization framework. Chen et al. [21] optimized the cutoff distance using the sparrow search algorithm, with the ACC index as the objective function, effectively reducing the impact of the cutoff distance on clustering results. Zhang et al. [22] improved DPC in density measurement and cluster center identification. Gao et al. [23] improved the local density calculation by constructing an optimization function based on dataset uncertainty and using the extended pattern search algorithm to automatically optimize the cutoff distance.

To address the limitations of the assignment strategy in the DPC algorithm, Yang et al. [24] proposed a generalized density peak clustering algorithm based on sequential similarity, and optimized the point assignment strategy through a graph-based approach. Guo et al. [25] ensured that points are correctly assigned to appropriate clusters by selecting local centers, estimating connectivity, and applying distance penalties. Qin et al. [26] proposed a method based on label propagation and similarity metrics. This method initially assigns labels to cluster centers and propagates them to other points based on their proximity to the nearest labeled points, reducing error propagation and enhancing clustering accuracy. Yu et al. [27] optimized the assignment process using three-way decision theory and D-S evidence theory.

The above algorithms effectively improve the clustering performance of DPC and achieve better application results. However, most algorithms do not simultaneously consider the local neighborhood information of points when calculating local density and assigning points. They typically use traditional Euclidean distance to calculate local density, and although some methods improve by optimizing the cutoff distance, they still rely on global distribution, resulting in suboptimal performance on certain two-dimensional complex datasets or high-dimensional real-world datasets. Therefore, we propose a new similarity measure that not only considers the distance between two points but also takes into account their neighborhood distribution, integrating this similarity measure throughout the clustering process. Based on this, we propose the interactive neighbors similarity-based density peaks clustering (INSDPC) algorithm. The primary contributions of this study are listed as follows:

- A novel similarity measure, termed interactive neighbors similarity, has been introduced. This measure integrates information from interactive neighbors and shared neighbors.
- A local density calculation method based on interactive neighbors similarity has been proposed. Not only does it effectively handle simple datasets, but it also addresses complex datasets with varying densities and high-dimensional features.
- A two-step assignment strategy based on interactive neighbors similarity and neighborhood information has been proposed. This approach enhances the likelihood of correctly assigning non-center points, addressing the chain reaction of errors that occurs when a point is misclassified.

The remainder of this paper is organized as follows. Section 2 provides some preliminaries. Section 3 presents the proposed INSDPC algorithm in detail. Some necessary experimental components are presented in Section 4. Section 5 shows the experimental results and analysis. Section 5.3 concludes the paper and suggests potential directions for future research.

2. Preliminaries

In this section, we introduce the notations of this paper as well as some basic concepts that are closely related to this work, such as DPC, *KNN*, and *SNN*. For more detailed description, one can refer to the literatures [14, 28].

2.1. Notations and descriptions

Some of the necessary notations are summarized in Table 1 for an easy read.

Table 1. Notations and descriptions.

Notations	Descriptions
X	The dataset, i.e., $X = \{x_1, x_2, \cdots, x_n\}$
n	The number of points, i.e., $n = |X|$
D	The distance matrix, i.e., $D = (d_{ij})_{n\times n}$
d_{ij}	The distance between x_i and x_j
d_i^k	The k-th smallest distance of x_i and any $x_j \in X$
K	The number of clusters about X
ρ_i	The local density of $x_i \in X$
δ_i	The relative distance of $x_i \in X$
γ_i	The decision value of $x_i \in X$

2.2. *The DPC algorithm*

The mechanism of DPC can be divided into two stages: how to select the cluster centers, and how to assign non-center points to correct clusters. Taking $x_i \in X$ for example, can be selected as the cluster center if and only if it satisfy the following two conditions at the same time: (1) ρ_i, the local density of x_i, must be greater than that of its neighboring points; (2) x_i must be relatively distant from points with a bigger ρ_j for some $j = 1, 2, \cdots, n$.

The value of ρ_i can be computed by the equation

$$\rho_i = \sum_{i \neq j} \chi(d_{ij} - d_c), \ j = 1, 2, \cdots, n, \tag{2.1}$$

where d_c is the cutoff distance [21], and $\chi(x)$ represents the indicator function computed by the following equation:

$$\chi(x) = \begin{cases} 1, & if \ \ d_{ij} < d_c, \\ 0, & otherwise. \end{cases} \tag{2.2}$$

Moreover, the value of ρ_i can also be computed by equation

$$\rho_i = \sum_{i \neq j} \exp\left[-\left(\frac{d_{ij}}{d_c} \right)^2 \right], \ j = 1, 2, \cdots, n. \tag{2.3}$$

In fact, Eq (2.1) is called the cutoff kernel method, and Eq (2.3) is called the Gaussian kernel method. Regardless of which method is choosed, it can be observed that ρ_i is highly sensitive to d_c.

The value of δ_i can be computed by the following equation:

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho_i} \left(d_{ij} \right), & \rho_i \neq \max \ \{\rho_1, \rho_2, \cdots, \rho_n\}, \\ \max\limits_{i \neq j} \left(d_{ij} \right), & \rho_i = \max \ \{\rho_1, \rho_2, \cdots, \rho_n\}. \end{cases} \tag{2.4}$$

Once a decision graph with ρ_i as the $x-$axis and δ_i as the $y-$axis is constructed, the points located at the upper right corner can be selected as the cluster centers more easily. In numerical terms, the higher the value of γ_i, the more easily x_i is selected as the cluster center. Here, γ_i is the decision value of x_i and can be computed by the following equation:

$$\gamma_i = \rho_i \ \delta_i. \tag{2.5}$$

Bearing ρ_i, δ_i, and γ_i of $x_i \in X$ in mind, the cluster centers, take $\{x_{i1}, x_{i2}, \cdots, x_{ik}\}$ for example, can be determined. After that, DPC assigns the remaining points to the cluster with the closest high-density center, completing the clustering.

2.3. *KNN and SNN*

In this subsection, we define the k-nearest neighbors and shared nearest neighbors, which are essential components of the proposed method.

The **k-nearest neighbors** of $x_i \in X$ is a subset of X, and it can be calculated by

$$KNN(i) = \left\{ x_j \mid d_{ij} \leq d_i^k \right\}. \tag{2.6}$$

The **shared nearest neighbors** of x_i and x_j is also a subset of X, and has the mathematical expression

$$SNN(i, j) = \{x_k \mid x_k \in KNN(i) \cap KNN(j)\}. \tag{2.7}$$

3. The proposed INSDPC method

In this section, we provide detailed information about the proposed method, including the motivation, the relevant definitions, the main mechanism, the algorithm pseudo-code and the computational complexity analysis.

3.1. Motivation

The performance of the traditional DPC algorithm is often suboptimal on complex datasets, primarily due to its reliance on the calculation of ρ_i and its sensitivity to the parameter d_c. The parameter d_c is typically determined based solely on the global data distribution, neglecting local characteristics. However, during clustering, the nearest neighbors of a point often belong to the same cluster. Furthermore, the one-step assignment strategy of the DPC algorithm has led to this issue: if a high-density point is misassigned, the error may affect the assignment of its neighboring points, potentially resulting in more severe allocation errors. To address these issues, we propose a similarity measure that integrates information from both interactive neighbors and shared neighbors. Based on this similarity, we redefine ρ_i and design a new two-step assignment strategy.

3.2. Definitions of interactive neighbors similarity, local density and relative distance

3.2.1. Interactive neighbors similarity

We provide a detailed definition of interactive neighbors similarity, its related properties, and compare it with other similarity measures.

Definition 1 (Interactive Neighbors Similarity)**.** *Given that x_i, $x_j \in X$, the equation for calculating the interactive neighbors similarity between x_i and x_j can be expressed as follows:*

$$Ins(i, j) = \frac{k^2}{\sum\limits_{p \in KNN(i),\ q \in KNN(j)} d_{pq}} \cdot |SNN(i, j)|\,, \tag{3.1}$$

where the first term on the right-hand side of the equation represents the reciprocal of the average distance between the KNN of x_i and x_j, accounting for the k^2 points and reflecting the density around the two points. The second term represents the number of shared neighbors between x_i and x_j, considering the overlap among these neighbors.

Obviously, Eq (3.1) satisfies the following mathematical properties.

Property 1. *Given that x_i , $x_j \in X$, then $Ins(i, j) \geq 0$.*

Proof. Because $k^2 > 0$, $\sum d_{pq} > 0$, and $|SNN(i, j)| \geq 0$ are true forever, the inequality

$$\text{Ins}(i, j) \geq 0$$

is obvious.

This completes the proof. □

Property 2. *Given that x_i , $x_j \in X$, then $Ins(i, j) = Ins(j, i)$.*

Proof. Because the distance d_{pq} satisfies $d_{pq} = d_{qp}$, and the number of shared neighbors $|SNN(i, j)|$ satisfies

$$|SNN(i, j)| = |SNN(j, i)|,$$

$Ins(i, j)$ is independent of the order of the points.

This completes the proof. □

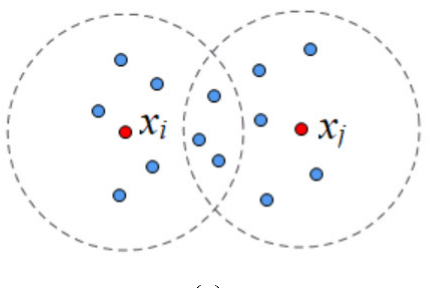
In addition to the above mathematical properties, interactive neighbors similarity also exhibits locality, which considers local neighborhood information. Specifically, it depends on the *KNN* of x_i and x_j, as well as the shared neighbors between them.

We compare interactive neighbors similarity with Euclidean distance and cosine similarity, as shown in Table 2.

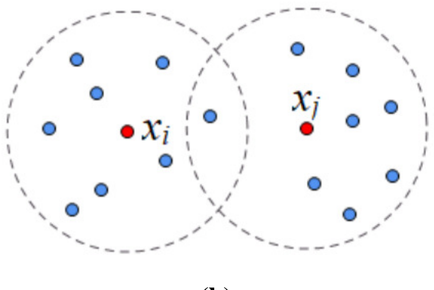
Table 2. Comparison of similarity measures.

Similarity measure	Core idea	Equation	Compared with INS
Euclidean distanc	Calculate the straight-line distance between two points.	$d(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{i=1}^{n}(x_{1,i} - x_{2,i})^2}$	Ignoring the neighborhood structure, it cannot distinguish density differences.
Cosine similarity	Measure the directional similarity between two vectors.	$\cos(x_1, x_2) = \frac{\mathbf{x_1} \cdot \mathbf{x_2}}{\|\mathbf{x_1}\|\|\mathbf{x_2}\|}$	Depends on the vector space and does not utilize geometric neighborhood information.

As shown in Figure 2, we consider two points x_i and x_j, with their respective *KNN* represented as blue dots, while x_i and x_j themselves are depicted as red dots. In Figure 2(a), the *KNN* of x_i and x_j are in close proximity, with some of them even overlapping. This close distribution means that x_i and x_j have more shared neighbors, making it more likely that they belong to the same cluster. In contrast, Figure 2(b) shows that the *KNN* of x_i and x_j are relatively far apart, with few shared neighbors, suggesting that x_i and x_j likely belong to two different clusters. Obviously, the proposed interactive neighbors similarity helps to identify similar points more accurately during the clustering process.



(a)



(b)

Figure 2. The neighborhood relationship between x_i and x_j.

3.2.2. Local density and relative distance

Following the definition of the interactive neighbors similarity between any two points, we define a new ρ_i for x_i.

Definition 2 (Local Density)**.** *The local density for $x_i \in X$ is defined as follows:*

$$\rho_i = \sum_{i \neq j} Ins(i, j), \; j = 1, 2, \cdots, n. \tag{3.2}$$

The summation process aggregates the interactive neighbors similarity between x_i and x_j for $j = 1, 2, \cdots, n$. It considers both distance factors and the number of shared neighbors, effectively capturing the clustering structure in the dataset and revealing the intrinsic relationships between points. This approach helps to identify dense and sparse regions in the dataset.

Definition 3 (Relative Distance)**.** *The relative distance of $x_i \in X$ is defined as follows:*

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho_i} (d_{ij}), & \rho_i \neq \max \{\rho_1, \rho_2, \cdots, \rho_n\}, \\ \max\limits_{i \neq j}(\delta_j), & \rho_i = \max \{\rho_1, \rho_2, \cdots, \rho_n\}. \end{cases} \tag{3.3}$$

The calculation method of relative distance is the same as that in the DPC algorithm.

3.3. The main mechanism of the INSDPC

The INSDPC algorithm retains the basic framework of the traditional DPC algorithm while improving its key stages. The following is a detailed explanation of the algorithm process.

3.3.1. Selection of cluster centers

Similar to the DPC algorithm, the initial stage of our method identifies cluster centers based on two critical parameters: ρ_i and δ_i. Before calculating these parameters, we propose the interactive neighbors similarity, as defined by Eq (3.1). Based on the given similarity measure, ρ_i is calculated using Eq (3.2), and δ_i is subsequently calculated using Eq (3.3). The γ_i for each point is then calculated using Eq (2.5) and sorted in descending order of their values. Finally, the top K points with the largest γ_i are selected as cluster centers, denoted as $\omega = \{c_1, c_2, \cdots, c_K\}$.

3.3.2. Assignment of core points based on interactive neighbors similarity

During the point assignment stage, instead of using the one-step assignment strategy in the traditional DPC, we adopt a two-step assignment method. First, core points are assigned to each cluster based on interactive neighbors similarity. These core points are close to the cluster center and have high ρ_i. The specific procedure for this step is described in Algorithm 2. In simple terms, if an unassigned point x_j are the k nearest neighbors of an assigned point x_i, and

$$Ins(i, j) > mean(Ins(i)),$$

then x_j is assigned to the cluster that x_i belongs to. Otherwise, if

$$Ins(i, j) \leq mean(Ins(i)),$$

x_j will be allocated in the second step.

The calculation of the average interactive neighbors similarity is as follows:

$$mean(Ins(i)) = \frac{1}{k} \sum_{x_j \in KNN(i)} Ins(i, j), \tag{3.4}$$

where k is the number of nearest neighbors, and $Ins(i, j)$ is the interactive neighbors similarity between the points x_i and x_j. This equation reflects the average similarity between x_i and the other points in its nearest neighborhood.

3.3.3. Allocation of remaining points

Core points have already been successfully assigned to their respective cluster, and it is necessary to allocate the remaining points in this step. The procedure for this step is outlined in Algorithm 3.

To begin with, the algorithm identifies all remaining points x_i for $i = 1, 2, \cdots, m$, and creates an allocation matrix $N \in \mathbb{R}^{m \times K}$, where each row corresponds to a remaining point and each column corresponds to a cluster in $\{C_1, C_2, \cdots, C_K\}$. For each remaining point x_i, the algorithm checks its k nearest neighbors to determine whether they have already been assigned to a cluster. If a neighbor x_j is assigned to cluster C_{c_j}, the count in the allocation matrix N corresponding to the row of x_i and the column of C_{c_j}, represented by N_{i,c_j}, is incremented by 1.

Then, the algorithm analyzes the matrix N to identify the cluster C_{c_i} with the highest count in each row, denoted as $max(N)$. If $max(N) > 0$, it indicates that at least one neighbor has already been assigned to cluster C_{c_i}, and the remaining point x_i will be allocated to this cluster. Conversely, if $max(N) = 0$, it means that none of the neighbors have been assigned to any cluster. In this case, the algorithm increases the value of k to expand the neighborhood for cluster assignment. This iterative process continues until all remaining points are successfully allocated to their respective clusters, ultimately forming the cluster set

$$\Omega = \{C_1, C_2, \cdots, C_K\}.$$

3.4. The pseudo-code of the proposed algorithm

The procedures of the proposed algorithm are outlined in Algorithms 1–3.

Algorithm 1 The interactive neighbors similarity based density peaks clustering algorithm

```
Input: Dataset $X = \{x_1, x_2, \cdots, x_n\}$, number of neighbors $k$, number of clusters $K$
Output: Final clustering results $\Omega = \{C_1, C_2, \cdots, C_K\}$
1: Calculate the distance matrix $D = (d_{ij})_{n \times n}$ according to Euclidean distance;
2: Calculate the interactive neighbors similarity between pairs of points according to Eq (3.1);
3: Calculate $\rho_i$ and $\delta_i$ according to Eqs (3.2) and (3.3), respectively;
4: Calculate $\gamma_i$ according to Eq (2.5) and sort it in descending order;
5: Select the top $K$ points with the largest $\gamma_i$ as cluster centers $\omega = \{c_1, c_2, \cdots, c_K\}$;
6: Use Algorithm 2 to assign core points based on interactive neighbors similarity;
7: Use Algorithm 3 to allocate remaining points to clusters.
```

Algorithm 2 Assignment of core points based on interactive neighbors similarity

```
Input: Dataset $X$, cluster centers $\omega = \{c_1, c_2, \cdots, c_K\}$, number of neighbors $k$, distance matrix $D$,
    interactive neighbors similarity $Ins(i, j)$ for $x_i \in X$
Output: Preliminary results $Pre = \{Pre(C_1), Pre(C_2), \cdots, Pre(C_K)\}$
 1: Initialize queue $Q \leftarrow \omega$;                          ▷ Push all cluster centers into $Q$
 2: while $Q \neq \emptyset$ do
 3:     $x_i \leftarrow$ Pop($Q$);                                     ▷ Pop the head point $x_i$
 4:     Find its $k$ nearest neighbors $KNN(i)$;
 5:     Calculate $mean(Ins(i))$ according to Eq (3.4);
 6:     for all unassigned $x_j \in KNN(i)$ do
 7:         if $Ins(i, j) > mean(Ins(i))$ then
 8:             Assign $x_j$ to the cluster that $x_i$ belongs to;
 9:             $Q \leftarrow Q \cup \{x_j\}$;                          ▷ Push $x_j$ into the tail of $Q$
10:         end if
11:     end for
12: end while
```

Algorithm 3 Allocation of remaining points

```
Input: Dataset $X$, preliminary results $Pre = \{Pre(C_1), Pre(C_2), \cdots, Pre(C_K)\}$, number of neighbors $k$
Output: Final clustering results $\Omega = \{C_1, C_2, \cdots, C_K\}$
 1: while not all points are allocated do
 2:     Find all remaining points and re-index them as $\{x_1, x_2, \cdots, x_m\}$;
 3:     Initialize allocation matrix $N \in \mathbb{R}^{m\times K}$;
 4:     for all remaining points $x_i$ do
 5:         for all $x_j \in KNN(i)$ do
 6:             if $x_j$ is assigned to cluster $C_{c_j}$ then
 7:                 $N_{i,c_j} \leftarrow N_{i,c_j} + 1$;
 8:             end if
 9:         end for
10:     end for
11:     Find max($N$);
12:     if max($N$) > 0 then
13:         Assign point $x_i$ to cluster $C_{c_i}$;
14:     else
15:         $k \leftarrow k + 1$;
16:     end if
17: end while
```

3.5. *Complexity analysis*

In this section, we analyze the time complexity of the proposed algorithm. Let n represent the total number of points, k represent the number of neighbors and K represent the number of clusters.

In the cluster centers selection stage of Algorithm 1, the computational complexity of the distance matrix calculation is $O(n^2)$. Subsequently, computing the interactive neighbors similarity for all point pairs has a time complexity of $O(kn^2)$. The computation of ρ_i and δ_i is the same as in the traditional DPC algorithm, with a time complexity of $O(n^2)$. Computing γ_i and selecting cluster centers require $O(n \log n)$. Therefore, the total time complexity of the cluster center selection stage is $O(kn^2)$.

The time complexity of Algorithm 2 is mainly determined by the initialization of the queue Q and the while loop. During the initialization phase, K cluster centers are added to the queue, and the time complexity is $O(K)$. In the while loop, the queue processes up to all n points, so the number of iterations is $O(n)$. Each iteration's time complexity is mainly determined by calculating the average interactive neighbors similarity and traversing and assigning unallocated neighbors, both of which have a time complexity of $O(k)$. In summary, the time complexity of each iteration is $O(k)$, and the overall complexity of the while loop is $O(nk)$. Including the initialization step, the total time complexity is $O(K + nk)$. Since K is usually much smaller than n, the time complexity of the core point assignment stage is $O(nk)$.

The time complexity of Algorithm 3 is mainly determined by the while loop and matrix operations. In the worst case, all points are allocated in this stage, so the while loop needs to be executed $O(n)$ times. During each iteration, the algorithm needs to traverse all remaining points and check the cluster assignments of each point's k nearest neighbors, constructing an $n \times K$ allocation matrix, the time complexity of this step is $O(nk + nK)$. Afterward, the algorithm needs to find the maximum value in the matrix to determine the final cluster assignment of the point, which has a time complexity of $O(nK)$. Therefore, the time complexity for a single iteration of the loop is $O(nk + nK)$, and the total time complexity is $O((k + K)n^2)$.

In conclusion, the time complexity of the INSDPC algorithm is $O((k + K)n^2)$. However, since both k and K are relatively small compared to n, the overall time complexity of the proposed INSDPC algorithm is $O(n^2)$.

4. Experimental components

In this section, we introduce several experimental components, including the datasets, evaluation metrics, compared algorithms, and parameter setting. The specified experimental environment we used is listed in Table 3.

Table 3. Experimental environment.

Parameter	Parameter value
RAM	16 GB
Speed	1.7 GHz
Programming	MATLAB R2022b
CPU	Intel(R) Core(TM) i5-1240P
System	Windows 11 system with 12 cores

4.1. Datasets

To assess the effectiveness of our algorithm, we use two categories of datasets for performance evaluation: synthetic and real-world datasets. Most synthetic datasets are two-dimensional, making them suitable for visualization and effective as evaluation tools for clustering algorithms. They are primarily used to test the performance of algorithms in identifying different types of clusters. Real-world datasets exhibit higher complexity, with varying dimensions and scales, enabling a more comprehensive evaluation of the algorithm's practical applicability. The synthetic and real-world datasets used in the experiments are detailed in Tables 4 and 5, respectively.

Table 4. Description of the used synthetic datasets.

Dataset	Source	Points	Attributes	Clusters
Flame	[29]	240	2	2
Pathbased	[30]	300	2	3
Spiral	[30]	312	2	3
R15	[31]	600	2	15
Aggregation	[32]	788	2	7
DIM512	[33]	1024	512	16
D31	[31]	3100	2	31
S2	[34]	5000	2	15

Table 5. Description of the used real-world datasets.

Dataset	Source	Points	Attributes	Clusters
Iris	[35]	150	4	3
Wine	[35]	178	13	3
Seeds	[36]	210	7	3
Glass	[35]	214	10	6
Ecoli	[35]	336	8	8
Ionosphere	[37]	351	34	2
Libras movement	[35]	360	90	15
Dermatology	[35]	366	33	6
WDBC	[35]	569	30	2
Waveform	[38]	5000	21	3

4.2. Evaluation metrics

We select four widely used evaluation metrics to assess the quality of the clustering results: accuracy (ACC) [39], adjusted mutual information (AMI) [40], adjusted rand index (ARI) [40] and Fowlkes-Mallows index (FMI) [41]. Among these, ACC and FMI range from 0 to 1, while AMI and ARI range from -1 to 1. In all cases, higher values indicate better clustering performance.

4.3. Compared algorithms

In the experiments, we select 9 clustering algorithms for comparison, which are DPC, K-means, FCM, DBSCAN, SNN-DPC, DPCSA , DPC-CE, HFDPC, and R-MDPC. Among them, DPC serves as the baseline method, K-means and FCM are distance-based clustering algorithms, DBSCAN is a representative density-based clustering algorithm, SNN-DPC is a satisfactory improvement of the DPC algorithm, and HFDPC and R-MDPC are two recent improvement algorithms for DPC. The description of each algorithm is as follows.

K-means [42]: The K-means algorithm iteratively assigns points to the nearest cluster center and updates the center's position until convergence.

FCM [43]: The FCM algorithm is a clustering algorithm that assigns membership degrees to points, enabling fuzzy classification into multiple clusters.

DBSCAN [13]: A density-based clustering algorithm that finds clusters of arbitrary shapes and detects noise points.

DPC [14]: This algorithm identifies high-density regions as cluster centers by combining density and distance, and then performs clustering.

SNN-DPC [44]: It combines shared nearest neighbor and density peak clustering, improving clustering performance on complex and noisy data by shared neighborhoods.

DPCSA [45]: Clustering by density peaks using weighted local density sequence and nearest neighbor assignment.

DPC-CE [25]: DPC-CE algorithm combines DPC and clustering effectiveness metrics, optimizing clustering results by introducing an evaluation mechanism.

HFDPC [46]: HFDPC preserves privacy via federated learning, integrates similar density chain, and enhances clustering performance through dimension reduction and image encryption.

R-MDPC [47]: It suppresses interference through relevant region allocation and a robust decision graph, accurately identifies main density peaks, and accelerates large-scale clustering based on KNN.

4.4. Parameter setting

To guarantee a fair comparison of experimental results across different algorithms, we perform parameter tuning for each algorithm to ensure optimal performance. The specific parameter settings are provided in Table 6.

Table 6. The parameter settings of the compared algorithms.

Algorithms	Parameter settings
K-means	Number of clusters K
FCM	$m = 1.5 \sim 3$
DBSCAN	Eps = 0.01 $\sim$ 1, MinPts = 1 $\sim$ 50
DPC	$dc = 2\% \sim 3\%$
SNN-DPC	$3 \leq k \leq 50$
DPCSA	Does not require tuning
DPC-CE	$dc = 2\%,\ Tr = 0.25,\ Pr = 0.3$
HFDPC	$\theta = 0.1 \sim 0.9, L_\theta = 0.1 \sim 0.9$
R-MDPC	$k = \lceil \sqrt{n} \rceil, c_v = 1$

For our algorithm, there is only one integer parameter k that requires tuning, and its range is restricted to [4, 50]. The lower bound is set to 4 because smaller values of k may cause the algorithm to become endless and cause an error. The upper bound is set at 50 because excessively large values of k have little impact on the algorithm's results, making further increases unnecessary [44].

We test the clustering results for each k value in the range [4, 50] with a step size of 1, and evaluate the clustering quality using four evaluation metrics. Finally, we select the k value that maximizes the ARI as the optimal value.

5. Results and analysis

In this section, we comprehensively evaluate the performance of the INSDPC algorithm on synthetic and real-world datasets, and finally analyze the impact of the parameter k on the results.

5.1. Results on synthetic datasets

In this subsection, we evaluate the performance of various clustering algorithms on 8 widely used synthetic datasets. Table 7 presents the performance of these algorithms on the datasets listed in Table 4, along with their parameter settings. The best performance is highlighted in bold. Overall, the results show that the proposed INSDPC algorithm outperforms other algorithms on most datasets, such as R15, D31, and S2, where it consistently demonstrates the best performance.

On the Aggregation and Flame datasets, the performance of the INSDPC algorithm is slightly lower than that of the R-MDPC algorithm. This may be because the R-MDPC algorithm's strategy is better suited for handling such datasets. Notably, on the Spiral and DIM512 datasets, the INSDPC algorithm achieves perfect scores of 1 across all four evaluation metrics, showcasing its ability to accurately classify all points in these datasets. Consequently, our algorithm exhibits advantages across most datasets.

Next, we present the visualization results of several synthetic datasets, as illustrated in Figures 3–8. In each figure, subplots (a)–(h) correspond sequentially to the clustering results from INSDPC, DPC, DPCSA, SNN-DPC, DBSCAN, DPC-CE, FCM , K-means, R-MDPC, and HFDPC. Points in different colors represent distinct clusters. Additionally, except for DBSCAN , cluster centers selected by other algorithms are marked with asterisks, whereas noise points identified by DBSCAN are marked with crosses. It is important to note that the HFDPC algorithm does not explicitly mark cluster centers, as it uses grid points as cluster centers instead of traditional sample points.

As illustrated in Figure 3, the INSDPC algorithm and most DPC variants can perform clustering effectively. For the DBSCAN algorithm, despite some points being classified as noise, the overall clustering shape is correct. In contrast, DPCSA and SNN-DPC exhibit certain misclassifications in the clusters on the right side of the figure. Additionally, FCM and K-means erroneously divide a single cluster into multiple clusters and selected cluster centers between clusters, resulting in less accurate clustering outcomes.

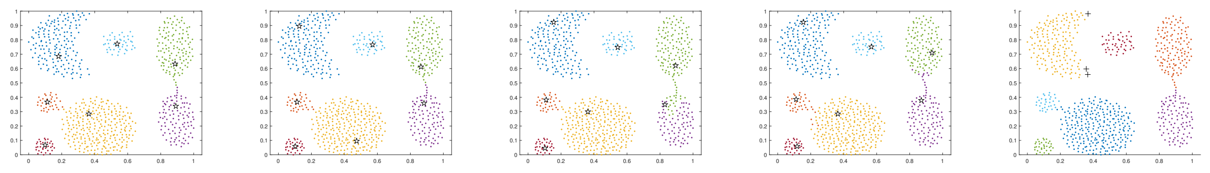
Table 7. Performance comparison of 10 clustering algorithms on synthetic datasets.

Algorithms	Aggregation					Spiral				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	29	0.9975	0.9905	0.9949	0.9960	5	**1.0000**	**1.0000**	**1.0000**	**1.0000**
HFDPC	0.7/0.7	0.9962	0.9892	0.9935	0.9949	0.4/0.6	**1.0000**	**1.0000**	**1.0000**	**1.0000**
R-MDPC	29/1	**1.0000**	**1.0000**	**1.0000**	**1.0000**	19/1	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DPCSA	–	0.9734	0.9537	0.9581	0.9673	–	**1.0000**	**1.0000**	**1.0000**	**1.0000**
SNN-DPC	15	0.9784	0.9500	0.9594	0.9681	5	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DPC	2.1	0.9949	0.9832	0.9898	0.9920	2	**1.0000**	**1.0000**	**1.0000**	**1.0000**
FCM	1.5	0.8591	0.7908	0.7272	0.7861	1.5	0.3429	-0.0054	-0.0059	0.3274
K-means	7	0.8261	0.8131	0.7694	0.8187	3	0.3429	-0.0055	-0.0060	0.3274
DBSCAN	0.08/21	0.9924	0.9774	0.9879	0.9905	0.04/1	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DPC-CE	–	0.9937	0.9802	0.9876	0.9903	–	**1.0000**	**1.0000**	**1.0000**	**1.0000**

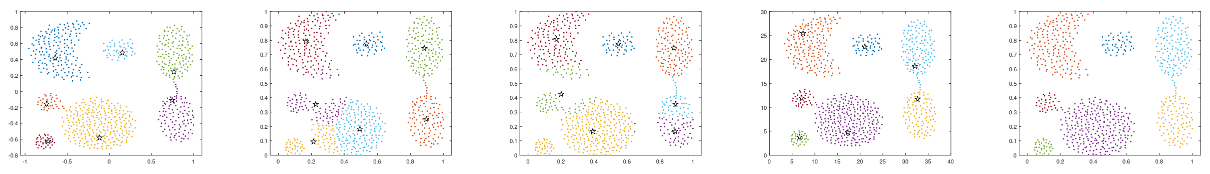
Algorithms	Flame					R15				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	10	0.9917	0.9324	0.9667	0.9845	13	**0.9967**	**0.9938**	**0.9928**	**0.9933**
HFDPC	0.6/0.5	0.9958	0.9615	0.9832	0.9922	0.5/0.5	0.9850	0.9761	0.9682	0.9703
R-MDPC	16/1	**1.0000**	**1.0000**	**1.0000**	**1.0000**	25/1	**0.9967**	**0.9938**	**0.9928**	0.9932
DPCSA	–	**1.0000**	**1.0000**	**1.0000**	**1.0000**	–	0.9933	0.9885	0.9857	0.9866
SNN-DPC	5	0.9875	0.8974	0.9502	0.9768	9	0.9917	0.9866	0.9823	0.9834
DPC	2.7	**1.0000**	**1.0000**	**1.0000**	**1.0000**	2.2	0.9966	0.9937	0.9927	0.9932
FCM	1.5	0.8500	0.4284	0.4880	0.7530	1.5	0.9965	0.9936	0.9926	0.9931
K-means	2	0.8458	0.4015	0.4762	0.7477	15	0.9966	0.9937	0.9927	0.9932
DBSCAN	0.1/9	0.9791	0.8551	0.9494	0.9763	0.05/26	0.9950	0.9907	0.9892	0.9899
DPC-CE	–	**1.0000**	**1.0000**	**1.0000**	**1.0000**	–	0.9966	0.9937	0.9927	0.9932

Algorithms	D31					DIM512				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	39	**0.9761**	**0.9649**	**0.9517**	**0.9532**	4	**1.0000**	**1.0000**	**1.0000**	**1.0000**
HFDPC	0.6/0.4	0.9323	0.9457	0.9039	0.9074	0.7/0.3	0.6504	0.7071	0.6332	0.6775
R-MDPC	57/1	0.9681	0.9549	0.9357	0.9378	33/1	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DPCSA	–	0.9677	0.9552	0.9353	0.9374	–	**1.0000**	**1.0000**	**1.0000**	**1.0000**
SNN-DPC	41	0.9758	0.9642	0.9509	0.9525	5	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DPC	2	0.9668	0.9539	0.9332	0.9353	2.3	**1.0000**	**1.0000**	**1.0000**	**1.0000**
FCM	1.5	0.8381	0.9077	0.8151	0.8220	1.5	0.5000	0.6891	0.5034	0.6142
K-means	31	0.8529	0.9214	0.8417	0.8486	16	0.8125	0.9030	0.8277	0.8510
DBSCAN	0.04/43	0.9042	0.9109	0.8518	0.8568	0.36/1	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DPC-CE	–	0.9690	0.9566	0.9378	0.9397	–	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Algorithms	Pathbased					S2				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	7	0.8967	0.7546	0.7272	0.8187	45	**0.9728**	**0.9500**	**0.9435**	**0.9472**
HFDPC	0.5/0.4	**0.9833**	**0.9302**	**0.9498**	**0.9665**	0.5/0.2	0.9670	0.9413	0.9314	0.9360
R-MDPC	18/1	0.5786	0.4293	0.3568	0.6058	72/1	0.9654	0.9399	0.9285	0.9333
DPCSA	–	0.8233	0.7073	0.6133	0.7511	–	0.9580	0.9333	0.9152	0.9209
SNN-DPC	9	0.9766	0.9001	0.9294	0.9529	35	0.9644	0.9386	0.9264	0.9313
DPC	2.1	0.7333	0.4997	0.4530	0.6585	2	0.9662	0.9409	0.9305	0.9352
FCM	1.5	0.7467	0.5138	0.4650	0.6634	1.5	0.9694	0.9451	0.9367	0.9409
K-means	3	0.7433	0.5098	0.4613	0.6619	15	0.9700	0.9461	0.9379	0.9420
DBSCAN	0.14/38	0.9000	0.7216	0.7552	0.8172	0.04/50	0.8376	0.8515	0.7593	0.7765
DPC-CE	–	0.7333	0.5084	0.4623	0.6652	–	0.9134	0.9200	0.8838	0.8938



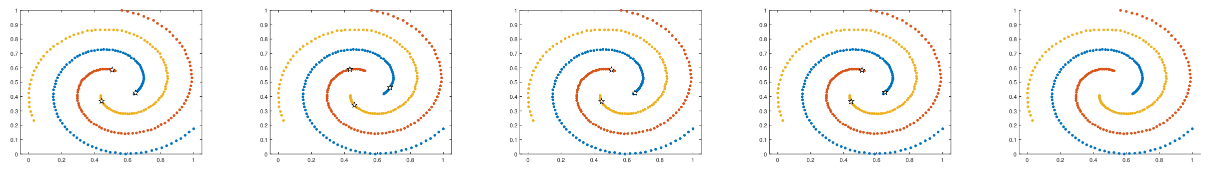
(a) INSDPC **(b)** DPC **(c)** DPCSA **(d)** SNN-DPC **(e)** DBSCAN



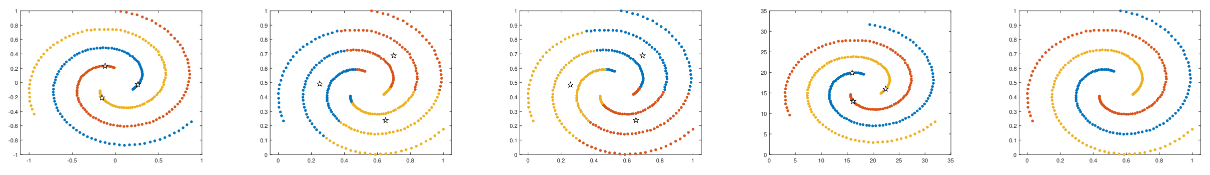
(f) DPC-CE **(g)** FCM **(h)** K-means **(i)** R-MDPC **(j)** HFDPC

Figure 3. Clustering results of Aggregation by 10 algorithms.

Figure 4 illustrates the ability of various algorithms to process cross-winding datasets. It is clear that our algorithm, along with other improved DPC variants, achieves exceptional performance in clustering. Notably, the positioning of cluster centers makes INSDPC distinct from traditional DPC. The cluster centers of INSDPC are closer to the endpoints of each cluster, reducing the risk of incorrect center selection and demonstrating the effectiveness of the proposed method. However, the clustering performance of FCM and K-means remains suboptimal. Even though the correct number of clusters is provided, these algorithms uniformly partition the dataset into three regions, selecting the centroids of each partition as cluster centers. This is because the Spiral dataset cannot be divided by straight lines, and algorithms like K-means and FCM, which rely solely on distance for clustering, perform poorly when handling this dataset.



(a) INSDPC **(b)** DPC **(c)** DPCSA **(d)** SNN-DPC **(e)** DBSCAN

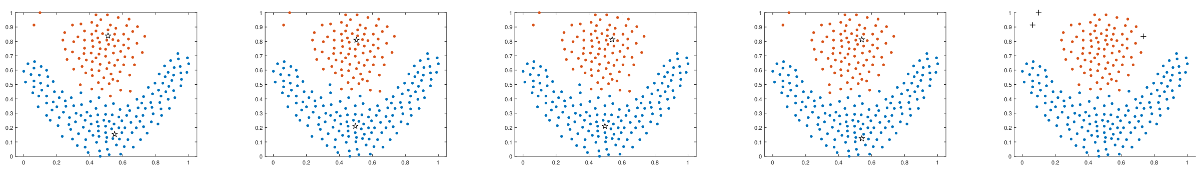


(f) DPC-CE **(g)** FCM **(h)** K-means **(i)** R-MDPC **(j)** HFDPC

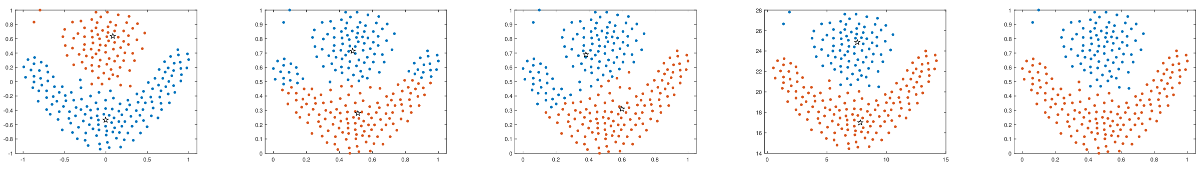
Figure 4. Clustering results of Spiral by 10 algorithms.

Figure 5 illustrates the clustering results of various algorithms on the Flame dataset. As the cluster shapes become increasingly complex, the performance of FCM and K-means is still not satisfactory. Both algorithms incorrectly assign some points from the lower cluster to the upper cluster, highlighting the limitations of these two algorithms in handling manifold-shaped clusters. The

DBSCAN algorithm effectively identifies the two clusters, yet it still struggles with managing noise points. In contrast, INSDPC, along with DPC and its improved variants, demonstrates outstanding performance by accurately identifying clusters and their centers.



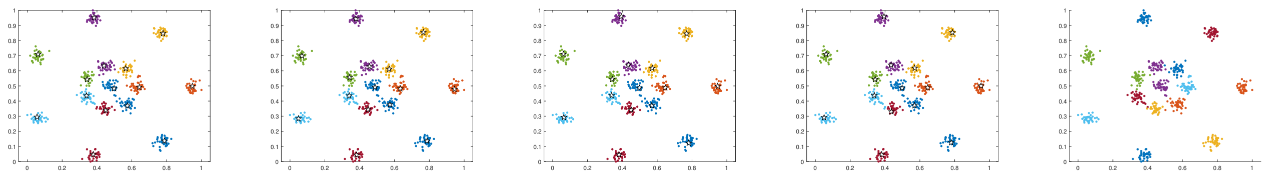
(a) INSDPC **(b)** DPC **(c)** DPCSA **(d)** SNN-DPC **(e)** DBSCAN



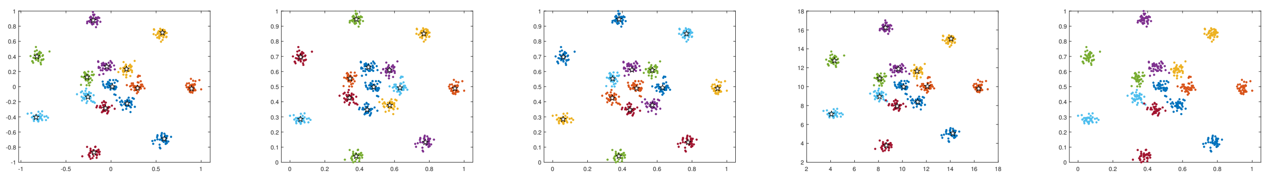
(f) DPC-CE **(g)** FCM **(h)** K-means **(i)** R-MDPC **(j)** HFDPC

Figure 5. Clustering results of Flame by 10 algorithms.

Figure 6 illustrates the clustering results for the R15 dataset, which contains 15 spherical clusters with uniformly distributed points. Although each algorithm has some minor imperfections, they all effectively identify the cluster centers and complete the clustering task.



(a) INSDPC **(b)** DPC **(c)** DPCSA **(d)** SNN-DPC **(e)** DBSCAN



(f) DPC-CE **(g)** FCM **(h)** K-means **(i)** R-MDPC **(j)** HFDPC

Figure 6. Clustering results of R15 by 10 algorithms.

Figure 7 illustrates the clustering results for the Pathbased dataset. With the exception of the R-MDPC algorithm, most algorithms accurately identify the cluster centers. The HFDPC and SNN-DPC algorithms perform the best, and our method outperforms other algorithms, with only a few misassignments in the lower-right region. The DPC algorithm and its variants, as well as distance-based algorithms, misassign some points from the left and right sides of the half-ring clusters to other clusters, leaving only a small portion of the half-ring cluster at the top. The DBSCAN algorithm is prone to point misassignments and generates considerable noise in the half-ring cluster. This issue arises because the density of this cluster is far below the value set by Minpts.

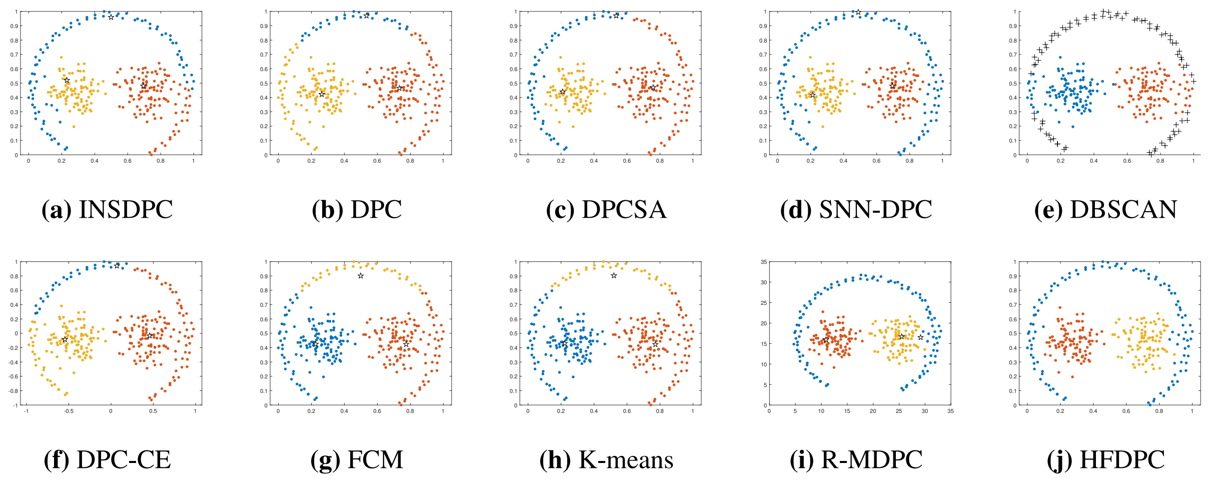


Figure 7. Clustering results of Pathbased by 10 algorithms.

Figure 8 illustrates the results of various clustering algorithms on the S2 dataset, which poses greater challenges due to its features, including noise, complex structures and uneven density distributions. The DPC-CE algorithm incorrectly identifies the cluster center, leading to suboptimal results. The DBSCAN algorithm fails to distinguish the three clusters in the lower-right region and erroneously classifies numerous boundary points as noise. The remaining algorithms correctly identify all clusters, but INSDPC outperforms them in capturing finer details.

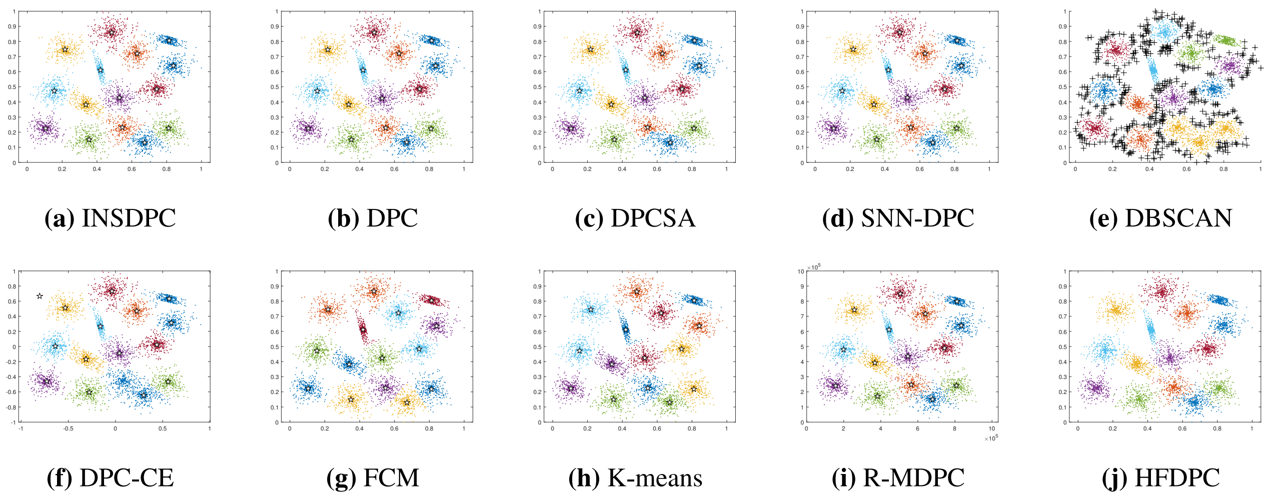


Figure 8. Clustering results of S2 by 10 algorithms.

From the results and analysis above, it is evident that the INSDPC algorithm outperforms other algorithms on most datasets. It not only accurately identifies cluster centers, but also effectively manages clusters with complex shapes. The main reason is that our algorithm takes into account the local neighborhood information of points. Specifically, in the selection of cluster centers, the calculation of local density relies on interactive neighbors similarity, which considers both the distance between two points and the number of shared nearest neighbors. The closer the points are, the greater their contribution to the density. During the point assignment stage, the remaining points are assigned based on the distribution of their neighbors. Considering these factors, our algorithm demonstrates relatively good clustering performance.

5.2. *Results on real-world datasets*

To further assess the effectiveness of the INSDPC, we conduct experiments on 10 real-world datasets. Table 8 summarizes the clustering performance of 10 algorithms on these datasets, along with the parameter settings for each algorithm. The best performance is highlighted in bold. Due to the increase in data dimensions and the presence of outliers in real-world datasets, all algorithms generally perform worse on real-world datasets compared to synthetic datasets. Nevertheless, the INSDPC algorithm demonstrates remarkable advantages, achieving the best performance on the Wine, Seeds, and Ionosphere datasets, which proves its excellent capability in handling high-dimensional and large-scale datasets. On the Iris dataset, both INSDPC and SNN-DPC achieve the highest clustering performance, significantly outperforming other algorithms. Although the INSDPC algorithm did not achieve the best AMI on the Waveform and the Glass datasets, it obtained the best ACC, ARI, and FMI, significantly better than those of the other algorithms. On the WDBC dataset, although the INSDPC algorithm has a lower FMI score than HFDPC, its ACC, AMI, and ARI are higher, making INSDPC the best in overall performance on the WDBC dataset. On the Dermatology dataset, INSDPC demonstrates excellent performance in the ACC metric, markedly outperforming other algorithms, although it is slightly surpassed by the SNN-DPC algorithm across other evaluation metrics. Overall, the INSDPC algorithm consistently achieves better performance than DPC across all datasets. The experimental results demonstrate that the algorithm exhibits good robustness in complex real-world environments.

5.3. *Parameter analysis*

In this subsection, we analyze the impact of the parameter k on the performance of the INSDPC algorithm. As a critical parameter, k specifies the number of neighboring points considered within the local neighborhood of each point during the clustering process. The value of k significantly affects the selection of cluster centers and the assignment of points. Consequently, the choice of k directly determines the performance of our algorithm.

Figure 9 illustrates the variations in AMI, ARI, ACC, and FMI for seven representative datasets with varying parameter k. We observe that the trends of the four metrics across datasets with varying k are generally consistent. The R15 dataset exhibits stability across all evaluation metrics, showing low sensitivity to k variations and maintaining consistently high scores for most k values. The Flame and D31 datasets show lower performance for small values of k ($k < 10$). However, as k increases, the performance gradually stabilizes, possibly because the expanded neighborhood range allows the algorithm to better identify the clustering structure. For most other datasets, the performance is better for lower k values, but declines gradually as k increases. A particularly notable exception is observed in the Ionosphere dataset, where an inflection point appears around $k = 17$. At this point, all evaluation metrics show a marked decline, followed by stabilization. This indicates that larger k are ineffective for this dataset. In summary, as k increases, the metrics for most datasets fluctuate within a small range or stabilize, which proves the robustness of our algorithm across different datasets.

Table 8. Performance comparison of 10 clustering algorithms on real-world datasets.

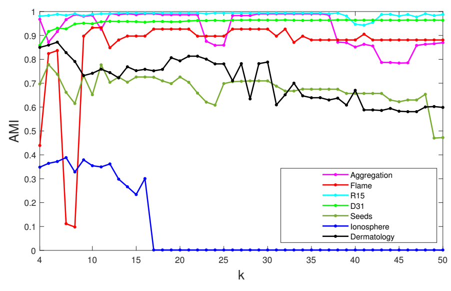
Algorithms	Wine					Iris				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	33	**0.9775**	**0.9083**	**0.9326**	**0.9552**	9	**0.9733**	**0.9124**	**0.9222**	**0.9479**
HFDPC	0.9/0.8	0.9382	0.7985	0.8137	0.8766	0.9/0.9	0.8800	0.7585	0.7074	0.8083
R-MDPC	14/1	0.9027	0.7365	0.7445	0.8277	13/1	**0.9733**	0.9074	0.8843	0.9315
DPCSA	–	0.8846	0.7480	0.7414	0.8283	–	0.9667	0.8831	0.9038	0.9355
SNN-DPC	18	0.9663	0.8735	0.8992	0.9329	15	**0.9733**	**0.9124**	**0.9222**	**0.9479**
DPC	2	0.8820	0.7065	0.6724	0.7835	2.9	0.8867	0.7668	0.7196	0.8159
FCM	1.5	0.9551	0.8416	0.8649	0.9102	1.5	0.8800	0.7198	0.7028	0.8020
K-means	3	0.9551	0.8427	0.8636	0.9094	3	0.5733	0.5275	0.4290	0.6613
DBSCAN	0.51/23	0.8146	0.5544	0.5379	0.7181	0.13/9	0.8600	0.6476	0.6246	0.7533
DPC-CE	–	0.6124	0.3223	0.2926	0.6192	–	0.9067	0.7934	0.7592	0.8407

Algorithms	Seeds					Dermatology				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	11	**0.9381**	**0.7770**	**0.8234**	**0.8817**	6	**0.9126**	0.8719	0.8658	0.8929
HFDPC	0.7/0.9	0.8952	0.6961	0.7244	0.8161	0.9/0.9	0.8302	0.8721	0.8455	0.8932
R-MDPC	15/1	0.8964	0.7133	0.6756	0.8143	20/1	0.7682	0.7054	0.6804	0.7622
DPCSA	–	0.8810	0.6609	0.6873	0.7918	–	0.8937	0.7451	0.6062	0.6896
SNN-DPC	6	0.9191	0.7387	0.7776	0.8513	19	0.8607	**0.8749**	**0.8689**	**0.9021**
DPC	2.4	0.9000	0.7172	0.7341	0.8231	2.1	0.8197	0.7470	0.6893	0.7512
FCM	1.5	0.8905	0.6705	0.7049	0.8026	1.5	0.7432	0.7675	0.6907	0.7498
K-means	3	0.8810	0.6528	0.6822	0.7875	6	0.6257	0.7771	0.5790	0.6664
DBSCAN	0.37/50	0.8476	0.5751	0.6036	0.7373	0.99/3	0.5902	0.5721	0.4165	0.5395
DPC-CE	–	0.8799	0.7320	0.7687	0.8184	–	0.7683	0.7239	0.6953	0.7437

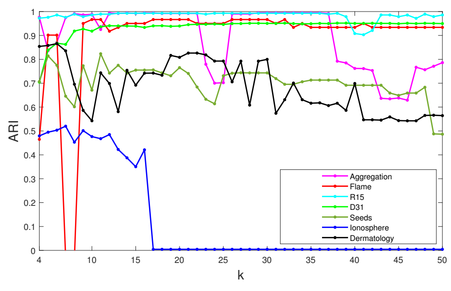
Algorithms	Ionosphere					Waveform				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	7	**0.8634**	**0.3884**	**0.5201**	**0.7903**	5	**0.7658**	0.4261	**0.4485**	**0.6439**
HFDPC	0.5/0.3	0.7151	0.1285	0.1821	0.6086	0.8/0.7	0.6658	**0.4372**	0.3367	0.5588
R-MDPC	20/1	0.6743	0.2021	0.2432	0.6043	72/1	0.7397	0.3877	0.3957	0.6158
DPCSA	–	0.7262	0.1335	0.2135	0.6390	–	0.6266	0.2510	0.2236	0.5327
SNN-DPC	5	0.8547	0.3644	0.4949	0.7798	7	0.7494	0.3984	0.4176	0.6164
DPC	2	0.5185	0.0728	0.0323	0.5984	2	0.5578	0.2214	0.1897	0.4767
FCM	1.5	0.5421	0.1827	0.2436	0.5331	1.5	0.4998	0.3508	0.2473	0.4989
K-means	2	0.7123	0.1294	0.1776	0.6053	3	0.5006	0.3641	0.2535	0.5036
DBSCAN	0.01/1	0.6410	0.3663	0.3730	0.7338	0.45/49	0.4874	0.1436	0.0727	0.4249
DPC-CE	–	0.7455	0.3112	0.2978	0.6874	–	0.5152	0.3650	0.2798	0.5305

Algorithms	Ecoli					Libras Movement				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	24	0.7679	0.5921	0.7157	0.8027	6	0.4250	0.4866	0.2975	0.3618
HFDPC	0.9/0.7	0.7976	0.5802	0.6674	0.7572	0.1/0.9	0.3750	0.3842	0.1714	0.2729
R-MDPC	19/1	0.8126	0.4537	0.5256	0.5887	20/1	0.2818	0.3229	0.1801	0.3352
DPCSA	–	0.6964	0.4170	0.4783	0.6742	–	0.2544	0.3127	0.1557	0.2611
SNN-DPC	6	**0.8452**	**0.6711**	**0.7547**	0.8243	11	0.4944	0.5834	0.3927	**0.4507**
DPC	2.1	0.5774	0.5269	0.4315	0.5714	2	0.4388	0.5083	0.3815	0.4388
FCM	1.5	0.5089	0.4602	0.3417	0.4882	1.5	0.4944	0.5423	0.3304	0.3750
K-means	8	0.5446	0.5170	0.4001	0.5384	15	0.4389	0.4667	0.2640	0.3206
DBSCAN	0.26/41	0.7440	0.5742	0.7046	0.7904	0.89/1	0.3167	0.3601	0.2447	0.2895
DPC-CE	–	0.7917	0.6439	0.7487	**0.8254**	–	**0.7539**	**0.6237**	**0.5218**	0.4020

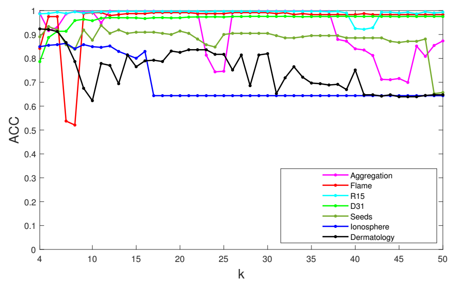
Algorithms	WDBC					Glass				
	Par	ACC	AMI	ARI	FMI	Par	ACC	AMI	ARI	FMI
INSDPC	8	**0.9385**	**0.6698**	**0.7668**	0.8950	31	**0.7851**	0.5486	**0.6566**	**0.7725**
HFDPC	0.7/0.6	0.8875	0.6204	0.7555	**0.8982**	0.7/0.9	0.4953	0.3871	0.2728	0.4473
R-MDPC	25/1	0.8383	0.3932	0.4452	0.7783	16/1	0.6916	0.5065	0.5422	0.6602
DPCSA	–	0.6963	0.3837	0.3771	0.3900	–	0.4533	0.1652	0.1772	0.5384
SNN-DPC	12	0.8765	0.6137	0.7433	0.8107	14	0.7243	**0.5779**	0.6051	0.7065
DPC	2.7	0.7206	0.1279	0.1916	0.6148	2.3	0.6682	0.5672	0.5885	0.6916
FCM	1.5	0.9279	0.6109	0.7302	0.8770	1.5	0.6495	0.5135	0.5001	0.6182
K-means	2	0.9315	0.6273	0.7423	0.8829	6	0.7025	0.5322	0.6157	0.7329
DBSCAN	0.45/46	0.8506	0.3706	0.4892	0.7592	0.27/1	0.4907	0.4255	0.3115	0.5634
DPC-CE	–	0.8664	0.4533	0.5284	0.8037	–	0.6542	0.4839	0.4175	0.6136



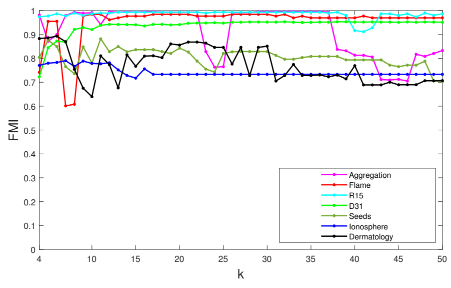
(a) AMI



(b) ARI



(c) ACC



(d) FMI

Figure 9. Results of different k on different datasets.

6. Conclusions and future work

This paper improves upon the traditional DPC algorithm and proposes a density peaks clustering algorithm based on interactive neighbors similarity, called INSDPC. The algorithm incorporates interactive neighbor relationships and shared neighbor information to redefine ρ_i, effectively addressing the limitations of the traditional DPC algorithm in handling clusters with varying densities. Additionally, INSDPC introduces a two-step assignment strategy that leverages interactive neighbors similarity and neighborhood information. This strategy addresses the joint assignment errors of the one-step assignment strategy in the traditional DPC algorithm. To evaluate the performance of INSDPC, we conduct experiments on both synthetic and real-world datasets. The results show that INSDPC more accurately identifies cluster centers and achieves higher clustering accuracy on most complex datasets.

Although the INSDPC algorithm exhibits outstanding clustering performance, determining the parameter k inevitably adds computational overhead. Future research will explore ways to automatically select the optimal k or propose an adaptive method to improve the algorithm's efficiency. On the application side, we aim to expand the algorithm to domains such as social network analysis, image segmentation, and bioinformatics to address practical challenges and further validate its applicability across various fields.

Author contributions

Shihu Liu: Writing-original draft, Software, Methodology, Conceptualization; Yirong He: Writing-review & editing, Data Curation, Visualization, Validation; Xiyang Yang: Formal analysis, Funding acquisition; Zhiqiang Yu: Supervision, Project administration. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

We are hugely grateful to the possible anonymous reviewers for their constructive comments with respect to the original manuscript. Additionally, we acknowledge the support received from the Fundamental Research Project of Yunnan Province (Grant No. 202301AT070015) , the Xingdian Talent Support Program for Young Talents (Grant Nos.XDYC-QNRC-2022-0518, XDYC-QNRC-2023-0388) and the Open Project of Fujian Provincial Key Laboratory of Data-Intensive Computing (Grant No. SJXY202401).

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. D. Wu, S. J. Zheng, X. P. Zhang, C. A. Yuan, F. Cheng, Y. Zhao, et al., Deep learning-based methods for person re-identification: A comprehensive review, *Neurocomputing*, **337** (2019), 354–371. https://doi.org/10.1016/j.neucom.2019.01.079
2. D. Jiang, W. Zang, R. Sun, Z. Wang, X. Liu, Adaptive density peaks clustering based on K-nearest neighbor and Gini coefficient, *IEEE Access*, **8** (2020), 113900–113917. https://doi.org/10.1109/ACCESS.2020.3003057
3. P. Jiao, W. Yu, W. Wang, X. Li, Y. Sun, Exploring temporal community structure and constant evolutionary pattern hiding in dynamic networks, *Neurocomputing*, **314** (2018), 224–233. https://doi.org/10.1016/j.neucom.2018.03.065
4. M. Nilashi, K. Bagherifard, M. Rahmani, V. Rafe, A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques, *Comput. Indust. Eng.*, **109** (2017), 357–368. https://doi.org/10.1016/j.cie.2017.05.016
5. L. Yang, X. Cai, S. Pan, H. Dai, D. Mu, Multi-document summarization based on sentence cluster using non-negative matrix factorization, *J. Intell. Fuzzy Syst.*, **33** (2017), 1867–1879. https://doi.org/10.3233/JIFS-161613
6. D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: a survey, *IEEE Trans. Knowl. Data Eng.*, **16** (2004), 1370–1386. https://doi.org/10.1109/TKDE.2004.68
7. S. Vassilvitskii, D. Arthur, k-means++: The advantages of careful seeding, In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2006, 1027–1035.

8. A. K. Jain, Data clustering: 50 years beyond K-means, *Pattern Recog. Lett.*, **31** (2010), 651–666. https://doi.org/10.1016/j.patrec.2009.09.011

9. S. Guha, R. Rastogi, K. Shim, CURE: An efficient clustering algorithm for large databases, *ACM Sigmod Record*, **27** (1998), 73–84. https://doi.org/10.1145/276305.276312

10. T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, *ACM Sigmod Record*, **25** (1996), 103–114. https://doi.org/10.1145/235968.233324

11. W. Wang, J. Yang, R. Muntz, STING: A statistical information grid approach to spatial data mining, In: *Proceedings of the 23rd International Conference on Very Large Data Bases*, **97** (1997), 186–195.

12. H. Asheri, R. Hosseini, B. N. Araabi, A new EM algorithm for flexibly tied GMMs with large number of components, *Pattern Recogn.*, **114** (2021), 107836. https://doi.org/10.1016/j.patcog.2021.107836

13. 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 the 2nd International Conference on Knowledge Discovery and Data Mining*, **96** (1996), 226–231.

14. A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, *Science*, **344** (2014), 1492–1496. https://doi.org/10.1126/science.1242072

15. Y. Wang, D. Wang, Y. Zhou, X. Zhang, C. Quek, VDPC: Variational density peak clustering algorithm, *Inf. Sci.*, **621** (2023), 627–651. https://doi.org/10.1016/j.ins.2022.11.091

16. Z. Liang, P. Chen, Delta-density based clustering with a divide-and-conquer strategy: 3DC clustering, *Pattern Recogn. Lett.*, **73** (2016), 52–59. https://doi.org/10.1016/j.patrec.2016.01.009

17. Y. Wang, D. Wang, X. Zhang, W. Pang, C. Miao, A. H. Tan, et al., McDPC: Multi-center density peak clustering, *Neural Comput. Applic.*, **32** (2020), 13465–13478. https://doi.org/10.1007/s00521-020-04754-5

18. Y. Wang, J. Qian, M. Hassan, X. Zhang, T. Zhang, C. Yang, et al., Density peak clustering algorithms: A review on the decade 2014–2023, *Expert Syst. Appl.*, **238** (2024), 121860. https://doi.org/10.1016/j.eswa.2023.121860

19. W. Tong, S. Liu, X. Z. Gao, A density-peak-based clustering algorithm of automatically determining the number of clusters, *Neurocomputing*, **458** (2021), 655–666. https://doi.org/10.1016/j.neucom.2020.03.125

20. Y. Li, L. Sun, Y. Tang, DPC-FSC: An approach of fuzzy semantic cells to density peaks clustering, *Inf. Sci.*, **616** (2022), 88–107. https://doi.org/10.1016/j.ins.2022.10.041

21. Y. Chen, J. Zhou, X. He, X. Luo, An improved density peaks clustering based on sparrow search algorithm, *Cluster Comput.*, **27** (2024), 11017–11037. https://doi.org/10.1007/s10586-024-04384-9

22. R. Zhang, Z. Miao, Y. Tian, H. Wang, A novel density peaks clustering algorithm based on Hopkins statistic, *Expert Syst. Applic.*, **201** (2022), 116892. https://doi.org/10.1016/j.eswa.2022.116892

23. T. Gao, D. Chen, Y. Tang, B. Du, R. Ranjan, A. Y. Zomaya, et al., Adaptive density peaks clustering: Towards exploratory EEG analysis, *Knowledge-Based Syst.*, **240** (2022), 108123. https://doi.org/10.1016/j.knosys.2022.108123

24. X. Yang, Z. Cai, R. Li, W. Zhu, GDPC: Generalized density peaks clustering algorithm based on order similarity, *Int. J. Mach. Learn. Cyber.*, **12** (2021), 719–731. https://doi.org/10.1007/s13042-020-01198-0

25. W. Guo, W. Wang, S. Zhao, Y. Niu, Z. Zhang, X. Liu, Density peak clustering with connectivity estimation, *Knowledge-Based Syst.*, **243** (2022), 108501. https://doi.org/10.1016/j.knosys.2022.108501

26. X. Qin, X. Han, J. Chu, Y. Zhang, X. Xu, J. Xie, et al., Density peaks clustering based on jaccard similarity and label propagation, *Cogn. Comput.*, **13** (2021), 1609–1626. https://doi.org/10.1007/s12559-021-09906-w

27. H. Yu, L. Y. Chen, J. T. Yao, A three-way density peak clustering method based on evidence theory, *Knowledge-Based Syst.*, **211** (2021), 106532. https://doi.org/10.1016/j.knosys.2020.106532

28. X. Yang, F. Xiao, An improved density peaks clustering algorithm based on the generalized neighbors similarity, *Eng. Applic. Artif. Intell.*, **136** (2024), 108883. https://doi.org/10.1016/j.engappai.2024.108883

29. L. Fu, E. Medico, FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data, *BMC Bioinf.*, **8** (2007), 3. https://doi.org/10.1186/1471-2105-8-3

30. H. Chang, D. Y. Yeung, Robust path-based spectral clustering, *Pattern Recogn.*, **41** (2008), 191–203. https://doi.org/10.1016/j.patcog.2007.04.010

31. C. J. Veenman, M. J. T. Reinders, E. Backer, A maximum variance cluster algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.*, **24** (2002), 1273–1280. https://doi.org/10.1109/TPAMI.2002.1033218

32. A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, *ACM Trans. Knowl. Discov. Data*, **1** (2007), 4-es. https://doi.org/10.1145/1217299.1217303

33. P. Franti, O. Virmajoki, V. Hautamaki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE *IEEE Trans. Pattern Anal. Mach. Intell.*, **28** (2006), 1875–1881. https://doi.org/10.1109/TPAMI.2006.227

34. P. Fränti, O. Virmajoki, Iterative shrinking method for clustering problems, *Pattern Recogn.*, **39** (2006), 761–775. https://doi.org/10.1016/j.patcog.2005.09.012

35. A. Asuncion, D. Newman, UCI machine learning repository, 2007. Available from: `https://ergodicity.net/2013/07/`.

36. M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, S. Żak, Complete gradient clustering algorithm for features analysis of x-ray images, In: *Advances in Intelligent and Soft Computing*, Berlin: Springer, 2010, 15–24. https://doi.org/10.1007/978-3-642-13105-9_2

37. V. G. Sigillito, S. P. Wing, L. V. Hutton, K. B. Baker, Classification of radar returns from the ionosphere using neural networks, *Johns Hopkins APL Technical Digest*, **10** (1989), 262–266.

38. L. Breiman, J. Friedman, R. A. Olshen, C. J. Stone, *Classification and regression trees*, London: Routledge, 2017.

39. D. Cai, X. He, J. Han, Document clustering using locality preserving indexing, *IEEE Trans. Knowl. Data Eng.*, **17** (2005), 1624–1637. https://doi.org/10.1109/TKDE.2005.198

40. N. X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, *J. Mach. Learning Res.*, **11** (2010), 2837–2854.

41. E. B. Fowlkes, C. L. Mallows, A method for comparing two hierarchical clusterings, *J. Amer. Stat. Assoc.*, **78** (1983), 553–569.

42. J. MacQueen, Some methods for classification and analysis of multivariate observations, In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California press*, **5** (1967), 281–298.

43. J. C. Bezdek, R. Ehrlich, W. Full, FCM: The fuzzy c-means clustering algorithm, *Comput. Geosc.*, **10** (1984), 191–203. https://doi.org/10.1016/0098-3004(84)90020-7

44. R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, *Inf. Sci.*, **450** (2018), 200–226. https://doi.org/10.1016/j.ins.2018.03.031

45. D. Yu, G. Liu, M. Guo, X. Liu, S. Yao, Density peaks clustering based on weighted local density sequence and nearest neighbor assignment, *IEEE Access*, **7** (2019), 34301–34317. https://doi.org/10.1109/ACCESS.2019.2904254

46. S. Ding, C. Li, X. Xu, L. Guo, L. Ding, X. Wu, Horizontal federated density peaks clustering, *IEEE Trans. Neural Networks Learning Syst.*, **36** (2023), 820–829. https://doi.org/10.1109/TNNLS.2023.3329720

47. J. Guan, S. Li, J. Zhu, X. He, J. Chen, Fast main density peak clustering within relevant regions via a robust decision graph, *Pattern Recogn.*, **152** (2024), 110458. https://doi.org/10.1016/j.patcog.2024.110458

AIMS Press