

Hierarchical assignment-based adaptive density peak clustering

Liu Yongli, Fan Rongzhen (✉), Chao Hao

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China

Abstract

The density peak clustering (DPC) algorithm is sensitive to the selection of the truncation distance and prone to chaining effects during point assignment. To address these issues, a hierarchical assignment-based adaptive density peak clustering (HADPC) algorithm was proposed in this paper. This algorithm first introduces the concept of natural neighbors to adaptively compute local density, thereby reducing reliance on global parameters. Next, it reveals potential cluster structures by distinguishing between core points and boundary points, forming core clusters within a localized scope based on data distribution. Finally, boundary points are assigned through a weighted voting mechanism, resulting in complete clusters. To evaluate the effectiveness of the HADPC algorithm, comparative experiments are conducted against five baseline algorithms on the synthetic and University of California, Irvine (UCI) datasets. Experimental results indicate that the HADPC algorithm can achieve higher clustering accuracy than the comparison algorithms.

Keywords density peak clustering (DPC), natural neighbors, adaptive density, hierarchical allocation

1 Introduction

In the era of big data, effectively analyzing and processing the vast datasets generated across various sectors has become a significant challenge. Clustering analysis, as a key technique in data mining and machine learning, can group similar data points^[1], thereby simplifying data complexity and revealing inherent patterns. This approach is widely applied in such fields as scientific research^[2], cybersecurity^[3], and bioinformatics^[4].

Various clustering algorithms are proposed to date,

among which the K -means^[5], hierarchical clustering^[6], and density-based spatial clustering of applications with noise (DBSCAN)^[7] are the most representative. These algorithms demonstrate effectiveness in practical applications; however, they also exhibit certain limitations. For instance, the K -means algorithm requires the pre-specification of the number of clusters and is sensitive to the selection of initial centroid points^[8]. Furthermore, it is unsuitable for clustering nonconvex shapes. Hierarchical clustering methods do not require a predefined number of clusters. However, they exhibit high computational complexity^[9], making them challenging to apply to large-scale datasets. While the DBSCAN algorithm can identify clusters of arbitrary shapes, it encounters challenges in assigning boundary points and parameter selection^[10]. As an emerging clustering algorithm, DPC^[11] gains increasing attention due to its simple

principle and strong clustering performance. This algorithm automatically identifies cluster centers by calculating the local density of each data point and the distance to high-density points, assigning the remaining data points to their respective clusters. While the DPC algorithm is robust to outliers and capable of identifying clusters of arbitrary shapes, it also exhibits certain limitations in practical applications.

The DPC algorithm is sensitive to the selection of the truncation distance^[12]. For different datasets, appropriate values must be selected according to their characteristics. This not only increases the algorithm's complexity but also limits its applicability.

The DPC algorithm suffers from a chaining allocation problem^[13]. After identifying the cluster centers, the DPC algorithm assigns non-cluster-center points to the nearest category with a higher local density in a single iteration. This method does not consider both the proximity of the samples to the cluster centers and the local neighborhood of the samples, making it prone to misallocation.

To mitigate the sensitivity of the DPC algorithm's density computation to the truncation distance, Du et al.^[14] proposed a study on DPC based on K -nearest neighbors (KNN) and principal component analysis (DPC-KNN) algorithm, which computes local density using the average KNN distance, thereby decreasing reliance on the truncation distance and enhancing the accuracy of local density estimation. Liu et al.^[15] proposed the shared-nearest-neighbor-based clustering by fast search and find of density peaks (SNNDPC) algorithm, which introduces shared nearest neighbor (SNN) similarity, a new definition of local density, and the distance to the nearest point of higher density, thereby enhancing the representation of the local structure of the data. Li et al.^[16] proposed the fuzzy neighborhood-based DPC algorithm (F-DPC), which adaptively selects the truncation distance based on data field theory to compute local density, thereby more accurately reflecting the distribution characteristics of the data. Zhao et al.^[17] introduced the improved DPC based on the firefly algorithm IDPC-FA, which employs the firefly algorithm to automatically determine the

truncation distance using the Rand index as the objective function, thereby enhancing the algorithm's adaptability. Jiang et al.^[18] proposed the adaptive DPC based on the KNN and Gini coefficient (G-KNN-DPC) algorithm, which determines the optimal truncation distance based on the Gini coefficient and employs KNN to identify cluster centers, thereby enhancing the accuracy of cluster center identification.

To mitigate the chaining allocation problem, Yu et al.^[19] developed a three-way DPC algorithm based on evidence theory (3W-DPET), which assigns ungrouped objects to the most appropriate clusters by integrating multi-source information, thereby partially alleviating the 'Domino effect'. Guo et al.^[20] proposed the DPC with connectivity estimation (DPC-CE) algorithm, which integrates Euclidean distance with graph-based connectivity information to optimize data point allocation by penalizing distances between local centers, thereby enhancing the utilization of the data's topological structure. Abbas et al.^[21] introduced the density peak based clustering using the mutual nearest neighbors (DenMune) algorithm, which selects at least KNN from other points as seed points and utilizes a voting system to determine point allocation, thereby enhancing allocation efficiency and robustness. Guo et al.^[22] proposed the hybrid clustering algorithm based on improved DPC (DPC-MS), which employs mean shift to identify convergence points as pseudo-cluster centers during the iterative process and merges similar pseudo-cluster centers, thereby enhancing point allocation strategies and improving adaptability to cluster shapes. The algorithm introduced by Shi et al.^[23] constructs a minimum spanning tree to gradually merge initial distance-based clusters, optimizing the point assignment process and demonstrating strong performance when handling chain-structured data. Han et al.^[24] introduced the graph-based density peak merging for identifying multi-peak clusters (CPM) algorithm, which utilizes a graph-based peak merging approach to construct directed subgraphs using valley and local density information, thereby making point allocation among multiple density peaks more reasonable and effective.

While the aforementioned algorithms demonstrate enhancements in clustering performance compared to the DPC algorithm, they still exhibit certain limitations. For example, DPC-KNN computes local density based on a fixed number of KNN, yet determining the optimal value of necessitates further investigation. Both SNNDPC and F-DPC escalate the computational complexity of the algorithms, thereby diminishing their efficiency in large-scale datasets. Furthermore, certain algorithms depend on complex models or optimization techniques, such as IDPC-FA, which employs the firefly algorithm, introduces additional parameters and increases computational burdens, ultimately lacking true adaptability and practicality. The 3W-DPET algorithm, grounded in evidence theory, may produce evidence conflicts when handling uncertain data, resulting in unstable allocation outcomes. While the DPC-CE algorithm employs graph structures and considers the topological relationships of the data, the graph structure may prove unreliable in sparse or noisy datasets, thereby affecting the accuracy of allocations. Other methods, including the DenMune algorithm, are sensitive to seed point selection, complicating the effective handling of datasets with significant density variations. The iterative process of the DPC-MS algorithm can incur substantial computational costs. Additionally, the minimum spanning tree method may not accurately capture the internal structure of complex-shaped clusters. The CPM algorithm is particularly sensitive to noise and outliers.

Unlike existing methods, HADPC initially adaptively identifies the neighbors of each data point through natural neighbor search, utilizing this information to compute local density, thereby circumventing the challenges associated with parameter selection. By employing weighted normalization, HADPC accurately identifies cluster centers within multi-peak datasets. Subsequently, HADPC identifies core points and boundary points based on local density and natural neighborhood characteristics. Finally, HADPC adopts a hierarchical allocation strategy: core points are assigned via breadth-first search (BFS), ensuring the stability of the clustering structure, while boundary

points are allocated through a weighted voting mechanism, with voting weights determined by neighborhood density and distance. This study's main contributions are summarized as follows.

1) HADPC algorithm is proposed. HADPC algorithm mitigates the influence of preset parameters often encountered in traditional algorithms, thereby enhancing the algorithm's robustness and adaptability.

2) Hierarchical allocation strategy based on core and boundary points is designed. This strategy effectively tackles the chaining allocation problem and shows strong robustness against noise and outliers.

3) K -dimensional tree (KD-Tree) is introduced to accelerate the neighbor search process, and this enhancement significantly boosts the algorithm's runtime efficiency for large-scale datasets.

The rest of the paper is organized as follows: Sect. 2 reviews related work and foundational concepts, Sect. 3 offers a comprehensive overview of the proposed HADPC algorithm. Sect. 4 presents experiments performed on diverse datasets. Finally, conclusions are drawn in Sect. 5.

2 Related work

2.1 DPC algorithm

The DPC algorithm is a density- and distance-based clustering method, which is defined by the following two fundamental concepts.

Definition 1 Local density

The local density of a data point x_i is defined as the number of data points within a radius d_c centered at x_i . For a dataset $X = \{x_1, x_2, \dots, x_i, \dots, x_I\}$ containing I samples, the local density ρ_i of each data point x_i is defined as

$$\rho_i = \sum_{x_j \in X, j \neq i} \chi(d_{ij} - d_c) \quad (1)$$

where the indicator function $\chi(\cdot)$ is defined as

$$\chi(x) = \begin{cases} 1; & x < 0 \\ 0; & x \geq 0 \end{cases} \quad (2)$$

In Eq. (1), d_{ij} is the Euclidean distance between the sample points x_i and x_j , which is defined in an M -dimensional space as

$$d_{ij} = \sqrt{\sum_{m=1}^M (x_{im} - x_{jm})^2} \quad (3)$$

The recommended choice for the truncation distance d_c is based on the average number of neighbors of the samples, typically between 1% and 2% of the total number of samples. For smaller datasets, Rodriguez et al.^[11] suggested using a Gaussian kernel function instead of a truncation kernel function to estimate the local density of the samples, where the Gaussian kernel function is defined as

$$\rho_i = \sum_{j=1, j \neq i}^I e^{-\left(\frac{d_{ij}}{d_c}\right)^2} \quad (4)$$

Definition 2 Relative distance

The relative distance of a data point x_i is defined as the minimum distance from x_i to the data point x_j with a higher local density. Specifically, δ_i represents the Euclidean distance from x_i to its nearest neighbor with higher density, which is defined as

$$\delta_i = \begin{cases} \min_j d_{ij}; & \exists j, \rho_j > \rho_i \\ \max_j d_{ij}; & \nexists j, \rho_j > \rho_i \end{cases} \quad (5)$$

After calculating the local density ρ_i and the relative distance δ_i , a two-dimensional (2D) ρ - δ decision diagram is constructed for all data points, and points with relatively high local density and relative distance are manually selected as cluster centers. Additionally, Rodriguez et al.^[11] introduced a decision value θ_i to comprehensively consider both local density ρ_i and relative distance δ_i , which is defined as

$$\theta_i = \rho_i \delta_i \quad (6)$$

Finally, the samples are sorted in descending order according to their local density values. Beginning with the sample exhibiting the highest density, the remaining samples are sequentially assigned to the clusters of their respective nearest neighbors with higher density, leading to the final clustering outcome.

As illustrated in Fig. 1, first, the Euclidean distance between any two points in the dataset is calculated according to Eq. (3). Subsequently, the truncation distance d_c is computed using the Euclidean distance matrix. Finally, the local density ρ_i and relative distance δ_i of each data point in the dataset are calculated based on Eq. (4) and Eq. (5), and a 2D

ρ - δ clustering decision diagram is constructed.

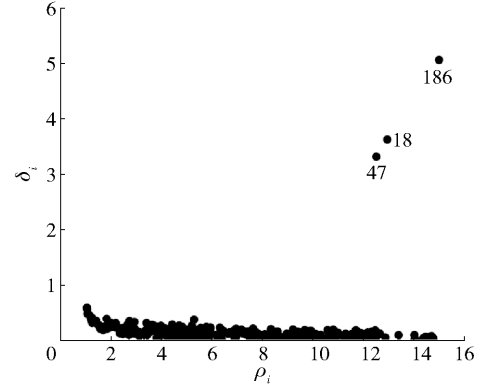


Fig. 1 Clustering decision diagram of the Three Blobs Dataset

In Fig. 1, taking the Three Blobs Dataset as an example, the complete clustering process of the DPC algorithm is demonstrated. In the clustering result figure, Feature 1 and Feature 2 represent the values of the first and second features of each sample, respectively, and are used to describe the positions of samples in the feature space. Based on these features, the algorithm further calculates the distances between samples and local densities, and then completes cluster center selection and sample assignment. As illustrated in Fig. 2, points 186, 18, and 47, characterized by relatively high decision values θ_i , are selected as cluster centers. Subsequently, the remaining data points are assigned to their corresponding clusters based on the nearest neighbors with higher density, leading to the final clustering outcome.

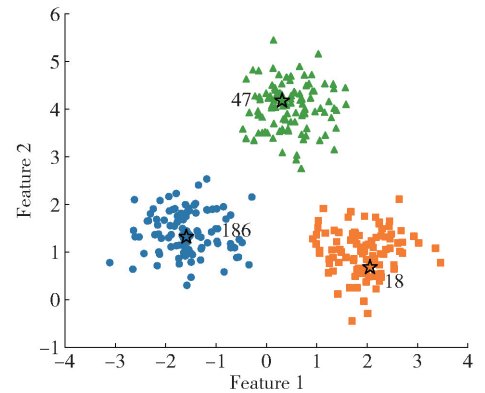


Fig. 2 Clustering results of the Three Blobs Dataset

The process of the DPC algorithm is shown in Algorithm 1.

Algorithm 1 DPC algorithm

Input: Dataset $X = \{x_1, x_2, \dots, x_I\}$.
Output: Clustering results.
Step 1: Standardize the dataset.
Step 2: Calculate the Euclidean distance d_{ij} between any two points in the dataset using Eq. (3), and construct the distance matrix $\mathbf{D} = (d_{ij})$.
Step 3: Compute the truncation distance d_c from the Euclidean distance matrix $\mathbf{D}$.
Step 4: Calculate the local density ρ_i and relative distance δ_i for each data point using the truncation distance d_c .
Step 5: Plot the clustering decision diagram with local density ρ_i along the horizontal direction and relative distance δ_i along the vertical direction, and determine the number of cluster centers N_c based on the decision diagram.
Step 6: Calculate the decision values θ_i for all data points using Eq. (6), and select the top N_c data points with the highest values as cluster centers.
Step 7: Assign category labels to the remaining data points based on their nearest neighbors with higher density in the dataset.
Step 8: Output the final clustering results.

2.2 Natural neighbor search

In KNN classification, selecting an appropriate value of k , $k = 1, 2, \dots, K$ remains a challenging issue^[25]. Unlike conventional KNN algorithms, the natural neighbor search algorithm adopts an adaptive search strategy, enabling data points in dense regions to have more natural neighbors, while those in sparse regions have fewer. This algorithm eliminates the need for manual presetting of the k value and more accurately reflects the local characteristics of the data.

The natural neighbor search algorithm utilizes adaptive searching, enabling each data point to have a distinct number of neighbors. Once all data points have identified their natural neighbors, the neighborhood relationships stabilize, and the search process concludes. The core principles of this algorithm are grounded in the following definitions.

Definition 3 Natural stable state

For a dataset $X = \{x_1, x_2, \dots, x_I\}$ containing I sample points, the number of searches r is initialized to 1 and incremented by 1 in each iteration. A KNN search is conducted on X with $k = r$ until each sample

point in dataset X has at least one other point as its mutual KNN. The search state at this point is termed the natural stable state. The natural stable state is attained when the following conditions are satisfied.

$$\forall x_i \exists x_j (r \in \mathbb{N}) \wedge (x_i \neq x_j) \rightarrow (x_i \in K_r(x_j) \wedge x_j \in K_r(x_i)) \quad (7)$$

where $K_r(\cdot)$ denotes the set of r -nearest neighbors of a sample.

Definition 4 Natural neighborhood feature value

The natural neighborhood feature value of dataset X is defined as the number of search rounds r at which the natural stable state is reached. That is, k^* is the minimum value of r that satisfies the natural stable state.

Definition 5 Natural neighbor

When dataset X is in a natural stable state, sample points that are neighbors are classified as natural neighbors. Specifically, for a sample point x_i , if another sample point x_j forms a mutual KNN relationship with x_i , then x_j is regarded as a natural neighbor of x_i . This is defined as

$$x_j \in N(x_i) \Leftrightarrow (x_i \in K_r(x_j) \wedge x_j \in K_r(x_i)) \quad (8)$$

The natural neighbor search algorithm initializes the search iteration $r = 1$ and establishes a natural neighbor set $N(x_i)$ and natural neighbor count s_{x_i} for each data point x_i . Subsequently, for the dataset $X = \{x_1, x_2, \dots, x_I\}$, a KNN search with $k = r$ is conducted to compute the r -nearest neighbor set $K_r(x_i)$ for each data point x_i . Then, the algorithm updates $N(x_i)$, adding x_i to the natural neighbor set $N(x_j)$ of its neighbor x_j and incrementing s_{x_j} . The search iteration r is incremented, and the KNN search is repeated until all data points satisfy the natural stable state, i. e., each sample point in dataset X has at least one other sample point with which it forms a mutual KNN relationship. Finally, the algorithm outputs the natural neighbor set $N(x_i)$, natural neighbor count s_{x_i} , and natural neighborhood feature value k^* . The detailed process is shown in Algorithm 2.

Algorithm 2 Natural neighbor search algorithm

Input: Dataset $X = \{x_1, x_2, \dots, x_I\}$
Output: Natural neighbor set $N(x_i)$, natural neighborhood

feature value k^* , and number of natural neighbors s_{x_i}

Step 1: Initialize the number of searches $r = 1$, natural neighbor count $s_{x_i} = 0$, and natural neighbor set $N(x_i) = \emptyset$.

Step 2: Perform KNN search on dataset X to compute the r -nearest neighbor set $K_r(x_i)$ for each sample point x_i .

Step 3: For each sample point x_i , update the natural neighbor set $N(x_i)$ and count s_{x_i} : if $x_j \in K_r(x_i)$ and $x_i \in K_r(x_j)$, add x_j to $N(x_i)$ and set $s_{x_i} = s_{x_i} + 1$.

Step 4: Increment search iteration $r = r + 1$.

Step 5: Check all sample points x_i ; if any x_i has $s_{x_i} = 0$, return to Step 2.

Step 6: When all x_i satisfy $s_{x_i} \geq 1$, set the natural neighborhood feature value $k^* = r$.

Step 7: Output the natural neighbor set $N(x_i)$, the natural neighbor feature value k^* , and the number of natural neighbors s_{x_i} .

3 HADPC algorithm

To address the issues related to the DPC algorithm's dependence on global parameters and its inadequate allocation strategy, the natural neighbor search algorithm is integrated into the HADPC algorithm. The HADPC algorithm overcomes the limitations of the DPC algorithm through two key improvements. First, it introduces an adaptive density calculation method that eliminates reliance on the truncation distance d_c . Second, it implements a hierarchical allocation strategy based on core and boundary points, thereby improving the stability and reliability of the clustering results.

3.1 Adaptive density estimation

The improved density calculation is adaptively defined based on the number of natural neighbors, eliminating the need for a global truncation distance. Instead, each point adaptively calculates its density based on its local environment, which enables reliable density estimation across regions with varying densities.

Definition 6 Improved local density

For a dataset $X = \{x_1, x_2, \dots, x_I\}$ containing I samples, the local density of each sample x_i is defined as

$$\rho_i = \frac{1}{s_{x_i}} \sum_{j \in N(x_i)} e^{-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}} \quad (9)$$

where $N(x_i)$ is the natural neighbor set, s_{x_i} is the number of natural neighbors of sample x_i , and σ_i is the local bandwidth parameter adjusted based on the sample's environment.

The bandwidth parameter σ_i can be adjusted based on the number of neighbors of the sample point, where $\bar{s}_X$ is the average number of natural neighbors of all sample points in dataset X .

$$\sigma_i = \frac{\bar{s}_X}{s_{x_i}} \quad (10)$$

As defined in Eq. (10), points in high-density regions are assigned smaller bandwidth parameter σ_i , while points in sparse regions will use a larger bandwidth, thus adapting to the density variations across different areas.

By calculating the local density of sample points using Eq. (9), this method avoids reliance on the truncation distance d_c and increases the likelihood of clustering centers emerging in relatively sparse areas, thus preventing their concentration in high-density regions. This approach facilitates more reliable clustering center selection, particularly for datasets with uneven density distributions.

To verify the accuracy of the adaptive density method in selecting cluster centers for datasets with significant density differences, this study uses the Jain dataset as an example to compare the performance of the local density defined by DPC with that of the improved adaptive local density. As shown in Fig. 3(a), when $d_c = 4$, the local density defined by the DPC algorithm produces multiple density peaks in dense regions because it ignores the intra-cluster density distribution. Consequently, traditional methods are prone to misjudging cluster centers in datasets with significant density differences. Fig. 3(b) illustrates the introduction of a natural neighbor weighted adaptive calculation of local density, which considers the relative density of samples within their neighborhoods, effectively balancing density differences across various areas. As a result, the distribution of local density becomes smoother and more coherent. Therefore, the adaptive density facilitates the accurate identification of cluster centers.

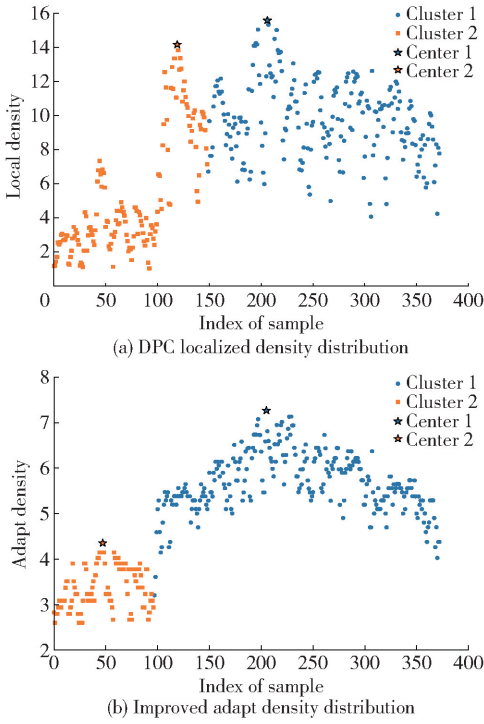


Fig. 3 Density distribution map of the Jain dataset

As shown in Fig. 4, the traditional DPC algorithm leads to incorrect cluster center selection due to limitations in density calculation, resulting in erroneous clustering outcomes.

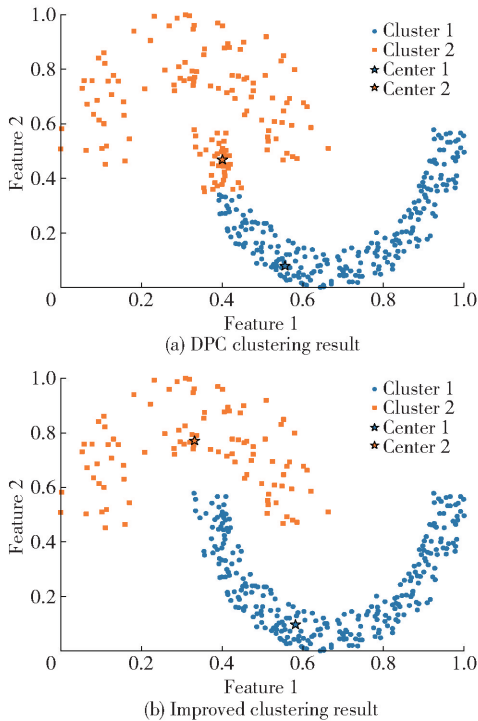


Fig. 4 Clustering results of the Jain dataset

3.2 Hierarchical allocation strategy

When identifying patterns in complex structures, human intuition typically first identifies prominent clusters in dense areas before processing objects on the boundaries^[26]. Based on this observation, a hierarchical assignment strategy that differentiates between core points and boundary points is introduced. This method categorizes data points into core points and boundary points based on the local environments of the sample points and their natural neighborhood feature values. Core points are assigned using a BFS method, whereas boundary points are assigned through a weighted voting mechanism.

In the hierarchical structure of data, certain points exhibit distinctly different attributes compared to others. These points occupy central positions within their local neighborhoods and shape the fundamental morphology of data clusters through their close connections with surrounding points. These points are termed core points. The identification of core points is based not only on absolute density but also on their significance within the relative local environment. Thus, the following definition is presented.

Definition 7 Second-order neighbor density

Accurately identifying core points is crucial in the hierarchical allocation process. To accurately identify core points, the concept of second-order neighbor density is introduced. For a dataset $X = \{x_1, x_2, \dots, x_l\}$ containing l data points, the second-order neighbor density $\eta(x_i)$ of each point x_i is defined as

$$\eta(x_i) = \frac{1}{s_{x_i} - 1} \sum_{x_j \in N(x_i)} s_{x_j} \quad (11)$$

where $N(x_i)$ is the set of natural neighbors of point x_i , and s_{x_i} is the number of neighbors of point x_i .

The second-order neighbor density $\eta(x_i)$ reflects the density distribution around point x_i . By calculating the average number of neighbors of the natural neighbors of point x_i , $\eta(x_i)$ captures density variations within the local area, providing more comprehensive information for identifying core points.

Definition 8 Core point

To ensure a core point is situated in a dense neighborhood and has a significant density advantage over its neighbors, the identification criteria for core points are

$$C(x_i) = \begin{cases} 1; & s_{x_i} > k^*, s_{x_i} \geq \eta(x_i) \\ 1; & s_{x_i} \leq k^*, s_{x_i} \geq 0.85\eta(x_i) \\ 0; & \text{otherwise} \end{cases} \quad (12)$$

where k^* denotes the natural neighborhood characteristic value, s_{x_i} is the number of natural neighbors of point x_i , and $\eta(x_i)$ is the second-order neighbor density.

In identifying core points, the number of natural neighbors for each sample point is initially compared with the natural neighborhood characteristic value, k^* , to differentiate between high-density and low-density regions in the dataset. Points with a natural neighbor count exceeding k^* are generally found in high-density areas and are regarded as potential core points. Conversely, points with a natural neighbor count below k^* are evaluated using their second-order neighbor density. They are classified as core points if the corresponding conditions are satisfied. This method ensures that the identified core points not only exhibit high-density but also maintain a density advantage within their neighborhood.

Definition 9 Boundary point

In contrast to core points, a sample point that is neither a cluster center nor a core point is defined as a boundary point. The definition of a boundary point is

$$B(x_i) = 1 - C(x_i) \quad (13)$$

The data points selected using Eq. (12) are considered core points. Core point assignment starts from the cluster center, employing a BFS to sequentially assign adjacent core points to the same cluster. At this stage, boundary points can be temporarily disregarded, clarifying the boundaries between clusters. Fig. 5 illustrates the distribution of core and boundary points in the synthetic dataset DS6, with red points representing core points. The identified core points effectively characterize the overall structure

and shape of the dataset.

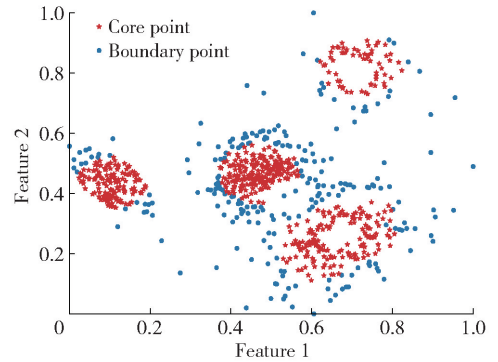


Fig. 5 Core points of the DS6 dataset

Definition 10 Weight calculation

The voting weight of boundary points is determined by their local density. Let the neighbor set be $N(x_i) = \{p_1, p_2, \dots, p_s, \dots, p_S\}$, where neighbors with higher local density exert greater influence on the assignment. With the local density of each neighbor p_s denoted by ρ_s , the voting weight w_s of neighbor p_s for boundary point x_i is defined as

$$w_s = \frac{\rho_s}{\sum_{s=1}^S \rho_s} \quad (14)$$

where $p_s \in N(x_i)$ denotes the s th neighbor of boundary point x_i , ρ_s denotes its local density, and S represents the total number of neighbors.

Definition 11 Weighted voting

After calculating the voting weight of each neighbor, the cluster label of a boundary point is determined by weighted voting among its neighbors. Let $G = \{G_1, G_2, \dots, G_l, \dots, G_L\}$ denote the set of cluster labels, where G_s represents the cluster label of neighbor p_s . For each candidate cluster $G_l \in G$, the voting weights of neighbors whose cluster label is G_l are summed. The boundary point x_i is assigned to the cluster with the maximum total voting weight, and defined as

$$G_{x_i} = \arg \max_{G_l \in G_{p_s \in N(x_i)}} \sum w_s f(G_s = G_l) \quad (15)$$

where $N(x_i)$ is the set of neighbors of the boundary point x_i , w_s is the voting weight of neighbor p_s , G_s is the cluster label of neighbor p_s , and $f(G_s = G_l)$ is an indicator function that equals 1 if $G_s = G_l$ and 0 otherwise.

By calculating the total voting weight for each cluster from the neighbors of the boundary point using Eqs. (14) and (15), the boundary point is assigned to the cluster with the highest voting weight. Fig. 6 illustrates the final clustering results for the DS6 dataset.

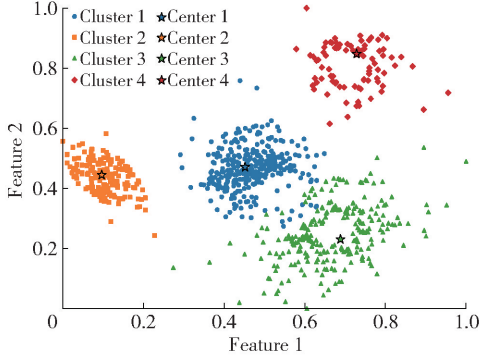


Fig. 6 Clustering results of the DS6 dataset

3.3 Algorithm flow and complexity analysis

The HADPC algorithm begins by applying Min-Max normalization to the dataset, ensuring consistent scaling across various features. Next, it employs the natural neighbor search algorithm to identify the natural neighbor set $N(x_i)$ and the count s_{x_i} of natural neighbors for each data point, calculating the adaptive local density ρ_i based on these neighbors using a weighted approach. Then, it computes the relative distance δ_i between each point and its nearest high-density point, plotting a clustering decision graph based on the local density ρ_i and relative distance δ_i to identify the clustering centers. Finally, it distinguishes between core points and boundary points by integrating each point's local density with neighborhood structure information. Core points are assigned to their corresponding clusters using a BFS, while boundary points are allocated to the appropriate clusters through a weighted voting mechanism based on the density and distance weights of their neighbors. The specific process of the algorithm is detailed in Algorithm 3.

Algorithm 3 HADPC algorithm

Input: Dataset $X = \{x_1, x_2, \dots, x_I\}$

Output: Clustering results

- Step 1: Perform Min-Max normalization on dataset X .
- Step 2: Use the natural neighbor search algorithm to determine each data point's natural neighbors $N(x_i)$ and count s_{x_i} .
- Step 3: Calculate the adaptive local density ρ_i for each data point based on its natural neighbors.
- Step 4: Compute the distance δ_i from each sample to the nearest high-density point.
- Step 5: Plot the clustering decision graph based on local density ρ_i and relative distance δ_i , and identify the clustering centers.
- Step 6: Identify core points and boundary points in the dataset based on data point density and local environment.
- Step 7: Allocate core points using BFS and allocate boundary points using a weighted voting mechanism.
- Step 8: Output the final clustering results.

The time complexity of the DPC algorithm is $O(I^2)$. The time complexity of the HADPC algorithm comprises the following components: data normalization is $O(I)$; natural neighbor search is $O(I^2)$; density calculation is $O(ISM)$; relative distance calculation is $O(I^2M)$; identification of core and boundary points is $O(I^2)$; core point allocation is $O(IS)$; boundary point allocation is $O(ISM)$, where S is the neighborhood size involved in natural-neighbor-based operations and M is the dimensionality of the sample points, both of which are much smaller than the sample size I . Therefore, the time complexity of the HADPC algorithm is $O(I^2)$, the same as that of the DPC algorithm. However, the time complexity of the HADPC algorithm results from multiple $O(I^2)$ functions, which may lead to increased computational time on large-scale datasets.

4 Experimental result and analysis

4.1 Experimental setup

This section presents an experimental evaluation of the HADPC algorithm by comparing with five representative clustering methods on synthetic and UCI datasets. The algorithms compared include DPC^[11], DPC-KNN^[14], SNNDPC^[15], DPC based on weighted local density sequence and nearest neighbor assignment (DPCSA)^[27], and DPC-CE^[20]. Except for DPCSA and DPC-CE, which do not require additional

parameter tuning, the other three algorithms underwent parameter optimization. For DPC, the parameter d_c is selected from the range 1% – 5% with a step size of 0.1%, with 50 trials to identify the optimal value. For DPC-KNN and SNNDPC, the parameter k is chosen in the range [1, 50] with a step size of 1, iterating 50 times to determine the best value. This study selects nine synthetic datasets, namely Jain, Flame, Aggregation, DS5, DS6, D31, S2, SpiralSquare, and Pathbased, and six UCI datasets, including Iris, Wine, Zoo, Seeds, Segment, and Landsat, for evaluation, as detailed in Tables 1 and 2. Here, I , M , and N_c denote the number of samples, data dimensionality, and the number of clusters, respectively. These datasets exhibit variations in overall distribution, as well as in the number of points and clusters, enabling the simulation of different scenarios to compare the performance of various clustering algorithms in diverse contexts.

Table 1 Synthetic dataset				
Dataset	I	M	N_c	Description
Jain	372	2	2	Containing one high-density cluster and one curved cluster, with a small inter-cluster distance
Flame	240	2	2	Containing a flame-shaped cluster and one spherical cluster
Aggregation	788	2	7	Containing seven clusters with different shapes and densities, with complex boundaries
DS5	500	2	5	Containing five irregularly shaped clusters with significant density variations
DS6	800	2	4	Containing four clusters with different density distributions
D31	3 100	2	31	Containing 31 randomly generated clusters, used to evaluate multi-cluster identification capability
S2	5 000	2	15	Containing 15 Gaussian-distributed clusters with relatively large scale
SpiralSquare	1 500	2	6	Containing a combination of spiral structures and square clusters
Pathbased	300	2	3	Containing a combination of path-shaped clusters and spherical clusters

Table 2 UCI dataset				
Dataset	I	M	N_c	Description
Iris	150	4	3	Iris dataset containing sepal and petal features of three iris species
Wine	178	13	3	Wine dataset containing 13 chemical attributes
Zoo	101	16	7	Animal classification dataset with various attribute features
Seeds	210	7	3	Wheat seed dataset with seven morphological features
Segment	2 310	18	7	Image segmentation dataset containing regional statistical features
Landsat	2 000	36	3	Satellite image dataset containing multispectral pixel features

The experimental environment for this study consists of an Intel i5-13490F central processing unit (CPU), 32 GB of random access memory (RAM), and a 64-bit Windows 11 operating system, developed using PyCharm 2023 (Professional Edition) and Python 3.10.

This study employs several clustering evaluation metrics, including adjusted mutual information (AMI), adjusted Rand index (ARI), and Fowlkes-Mallows index (FMI). Among these metrics, AMI assesses the similarity between clustering results and true categories, accounting for the effects of random clustering. The range of AMI is [0, 1], where a value of 1 indicates perfect alignment with true categories, and a value of 0 signifies complete disagreement with them. Its definition is provided as

$$V_{AMI} = \frac{V_{MI}(R_1, R_2) - E\{V_{MI}(R_1, R_2)\}}{\max(V_{MI}(R_1, R_2)) - E\{V_{MI}(R_1, R_2)\}} \quad (16)$$

where $V_{MI}(R_1, R_2)$ represents the mutual information between the clustering results R_1 and the true categories R_2 . $E\{V_{MI}(R_1, R_2)\}$ denotes the expected value of the mutual information, and $\max(V_{MI}(R_1, R_2))$ signifies the maximum mutual information value in the case of complete agreement.

The formula for calculating mutual information V_{MI} is shown as

$$V_{MI} = \sum_{a=1}^{N_c} \sum_{b=1}^{N_r} \frac{n_{ab}}{I} \ln \frac{n_{ab} I}{n_a n_b} \quad (17)$$

where N_c represents the number of clusters in the clustering results, N_r denotes the number of clusters in the true clustering results, n_a indicates the number of samples in the a th cluster, n_b signifies the number of

samples in the b th true cluster, n_{ab} refers to the number of samples in the intersection of the a th cluster and the b th true cluster, and I represents the total number of samples.

The ARI is a metric that measures the similarity between two clustering results. Its value lies in the range $[-1, 1]$, where values closer to 1 indicate better clustering results and values closer to -1 indicate poorer clustering results. Its definition is given by

$$V_{\text{ARI}} = \frac{V_{\text{RI}} - E\{V_{\text{RI}}\}}{\max V_{\text{RI}} - E\{V_{\text{RI}}\}} \quad (18)$$

where V_{RI} denotes the Rand index, $E\{V_{\text{RI}}\}$ represents the expected value of the Rand index under random clustering, and $\max V_{\text{RI}}$ denotes the maximum value of the Rand index when the clustering result is completely consistent with the true categories. The detailed formula for calculating the V_{ARI} is

$$V_{\text{ARI}} = \left\{ \sum_{a=1}^{N_c} \sum_{b=1}^{N_r} \binom{n_{ab}}{2} - \left[\sum_{a=1}^{N_c} \binom{n_a}{2} \sum_{b=1}^{N_r} \binom{n_b}{2} \right] \cdot \left(\frac{I}{2} \right)^{-1} \right\} \left\{ \frac{1}{2} \left[\sum_{a=1}^{N_c} \binom{n_a}{2} + \sum_{b=1}^{N_r} \binom{n_b}{2} \right] - \left[\sum_{a=1}^{N_c} \binom{n_a}{2} \sum_{b=1}^{N_r} \binom{n_b}{2} \right] \left(\frac{I}{2} \right)^{-1} \right\}^{-1} \quad (19)$$

where a and b represent the indices of clusters in the clustering results and true categories, respectively, n_a indicates the number of samples in the a th cluster, n_b signifies the number of samples in the b th true cluster, n_{ab} refers to the number of samples in the intersection of the a th cluster and the b th true cluster, and I represents the total number of samples.

The FMI is based on pairwise comparisons, evaluating whether pairs of data points are assigned to the same cluster in both clustering results, thereby measuring the similarity between them. Its value lies in the range $[0, 1]$, where 1 indicates complete agreement between the two clustering results, and 0 indicates complete disagreement. Its definition is given by

$$V_{\text{FMI}} = \sqrt{\frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}}} \quad (20)$$

where N_{TP} denotes the number of pairs of data points assigned to the same cluster in both clusterings, N_{FP} indicates the number of pairs assigned to the same cluster in one clustering but not in the other, and N_{FN}

represents the number of pairs not assigned to the same cluster in one clustering but assigned to the same cluster in the other.

4.2 Experiments on a synthetic dataset

The performance of the HADPC algorithm is compared with that of other five clustering algorithms across nine synthetic datasets. Table 3 presents the clustering results of the HADPC algorithm in comparison to other algorithms across multiple synthetic datasets. The experimental results indicate that the HADPC algorithm performs well on most datasets, particularly the Jain, DS5, DS6, SpiralSquare, and Pathbased datasets, where it achieves optimal values for all evaluation metrics. For the Flame, Aggregation, D31, and S2 datasets, although some metrics do not reach optimal levels, the overall performance remains close to the best. For instance, in the D31 dataset, the evaluation metrics of the HADPC algorithm are slightly lower than those of the SNNDPC algorithm, which may be attributed to the uniform density and predominantly spherical clusters in the D31 dataset, as SNNDPC is better suited for such cluster structures. Nonetheless, the clustering results of the HADPC algorithm still outperform those of the other comparative algorithms. Overall, the experimental results across various synthetic datasets demonstrate that, in most cases, the clustering results of the HADPC algorithm are superior to, or at least comparable with, those of other algorithms. In comparison to traditional DPC and its improved versions, the HADPC algorithm exhibits higher accuracy and stability when handling clusters of varying densities, shapes, and complexities, while requiring fewer adjustable parameters.

Table 3 Comparison of evaluation metrics for different clustering algorithms on synthetic datasets

Dataset	Algorithm	V_{AMI}	V_{ARI}	V_{FMI}	Number of parameters
Jain	HADPC	1.000 0	1.000 0	1.000 0	–
	DPC	0.576 1	0.618 3	0.838 6	2
	DPC-KNN	0.643 9	0.705 5	0.877 9	7
	SNNDPC	0.558 1	0.593 5	0.827 2	39
	DPCSA	0.231 3	0.044 2	0.592 4	–
	DPC-CE	1.000 0	1.000 0	1.000 0	–

Table 3 Continued					
Dataset	Algorithm	V_{AMI}	V_{ARI}	V_{FMI}	Number of parameters
Flame	HADPC	0.887 9	0.933 7	0.969 6	–
	DPC	1.000 0	1.000 0	1.000 0	4
	DPC-KNN	0.935 7	0.966 7	0.984 5	5
	SNNDPC	0.899 0	0.950 2	0.976 8	5
	DPCSA	1.000 0	1.000 0	1.000 0	–
	DPC-CE	0.518 1	0.499 8	0.758 5	–
Aggregation	HADPC	0.979 3	0.985 8	0.988 8	–
	DPC	0.992 3	0.995 6	0.996 6	4
	DPC-KNN	0.992 3	0.995 6	0.996 6	7
	SNNDPC	0.979 2	0.984 7	0.988 0	35
	DPCSA	0.957 0	0.958 1	0.967 3	–
	DPC-CE	0.992 3	0.995 6	0.996 6	–
DS5	HADPC	1.000 0	1.000 0	1.000 0	–
	DPC	0.866 8	0.730 4	0.798 7	4
	DPC-KNN	0.863 9	0.718 2	0.788 4	26
	SNNDPC	0.858 9	0.719 8	0.789 8	14
	DPCSA	0.682 0	0.397 2	0.595 8	–
	DPC-CE	0.747 8	0.544 6	0.678 4	–
DS6	HADPC	0.975 9	0.984 5	0.989 0	–
	DPC	0.915 4	0.914 9	0.940 2	4
	DPC-KNN	0.925 7	0.920 6	0.944 2	29
	SNNDPC	0.863 7	0.717 6	0.787 9	23
	DPCSA	0.920 6	0.913 0	0.938 9	–
	DPC-CE	0.885 5	0.885 3	0.919 4	–
D31	HADPC	0.962 5	0.948 9	0.950 5	–
	DPC	0.955 3	0.935 2	0.937 3	2
	DPC-KNN	0.958 2	0.940 9	0.942 8	31
	SNNDPC	0.964 2	0.950 9	0.952 5	41
	DPCSA	0.955 2	0.935 3	0.937 4	–
	DPC-CE	0.957 1	0.938 9	0.940 9	–
S2	HADPC	0.940 9	0.925 4	0.930 3	–
	DPC	0.934 8	0.920 2	0.925 5	5
	DPC-KNN	0.944 5	0.935 9	0.940 2	28
	SNNDPC	0.961 2	0.946 4	0.948 1	35
	DPCSA	0.933 6	0.915 2	0.920 9	–
	DPC-CE	0.944 5	0.935 9	0.940 2	–
SpiralSquare	HADPC	0.997 5	0.999 4	0.999 6	–
	DPC	0.735 4	0.513 2	0.634 7	4
	DPC-KNN	0.993 1	0.998 2	0.998 7	5
	SNNDPC	0.993 1	0.998 2	0.998 7	19
	DPCSA	0.992 5	0.998 2	0.998 7	–
	DPC-CE	0.990 6	0.997 6	0.998 2	–
Pathbased	HADPC	0.969 9	0.979 8	0.986 6	–
	DPC	0.579 4	0.506 7	0.684 4	2
	DPC-KNN	0.557 8	0.479 7	0.670 3	7
	SNNDPC	0.900 7	0.929 4	0.952 9	9
	DPCSA	0.729 0	0.613 3	0.751 1	–
	DPC-CE	0.498 8	0.378 7	0.628 5	–

To visually illustrate the performance differences among the algorithms, some experimental results are shown using visual graphics, where different colors denote different clusters, and the center of each cluster is marked with an asterisk. A detailed analysis of the experimental results is presented below.

The Jain dataset displays a clear upper and lower branching structure, with the upper branch containing relatively sparse data points, while the lower branch is denser. This variation in density distribution significantly affects the selection of clustering centers. As illustrated in Fig. 7, the HADPC algorithm effectively captures the distribution characteristics of various density regions. Employing the adaptive density calculation method, HADPC algorithm balances the influence of high-density areas on low-density regions and accurately identifies the corresponding clustering centers. The DPC-CE algorithm also performs well on this dataset, successfully identifying the correct clustering centers.

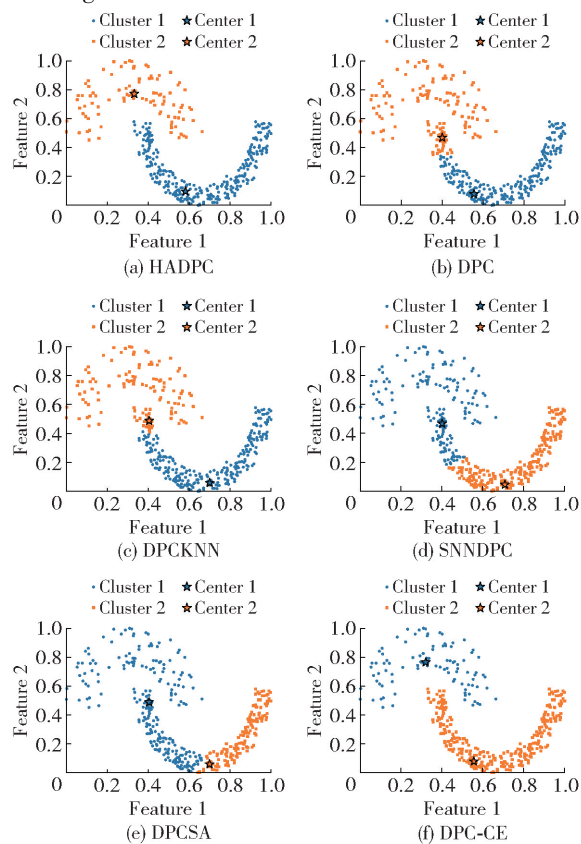


Fig. 7 Clustering results of six algorithms on the Jain dataset

In contrast, other algorithms do not fully account for the density variations within the dataset, often favoring regions with higher local density when selecting clustering centers, which leads to outcomes that deviate from the true cluster structure.

The Flame dataset comprises two uniformly dense clusters, whose centers are relatively separated, but there is a significant overlapping area between them, complicating the classification of boundary points. As illustrated in Fig. 8, the HADPC algorithm employs a hierarchical assignment strategy to first identify the core regions of the clusters. It then applies a weighted assignment method to classify the data points in the overlapping area based on their association with each core region. While the DPC, DPCSA, and DPC-CE algorithms can accurately identify the clustering centers, they struggle to partition the data points in the overlapping regions between clusters precisely. The DPC-KNN and SNNDPC algorithms exhibit poor adaptability to the overlapping areas, failing to capture the true distribution of the clusters, resulting in noticeable errors in the clustering outcomes.

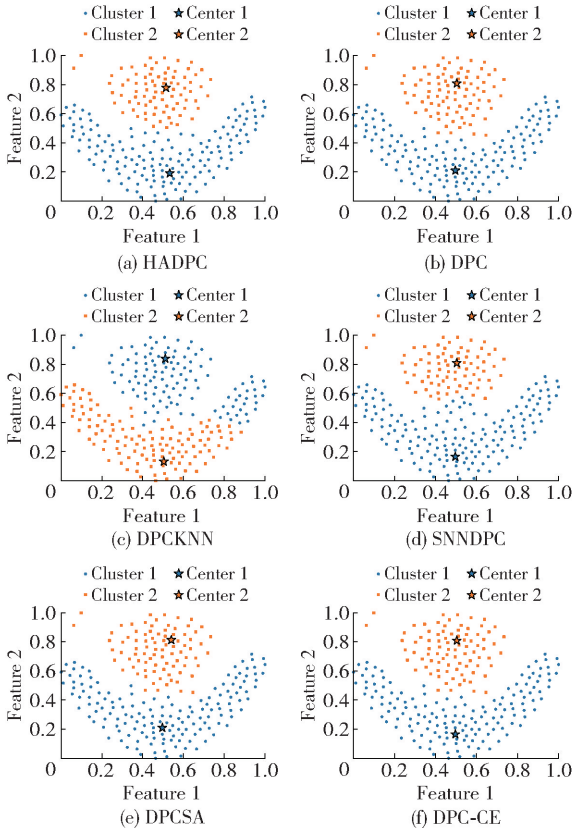


Fig. 8 Clustering results of six algorithms on the Flame dataset

The Aggregation dataset consists of seven clusters of varying sizes, with some clusters interconnected, complicating boundary identification during the clustering process. As illustrated in Fig. 9, the HADPC, DPC, and DPC-CE algorithms performed exceptionally well on this dataset, accurately identifying the clustering centers of all clusters and correctly delineating the boundaries between them. Although the DPC-KNN, SNNDPC, and DPCSA algorithms recognize the cluster centers and achieve good results when the clusters are completely separated, they struggle with the interconnected regions.

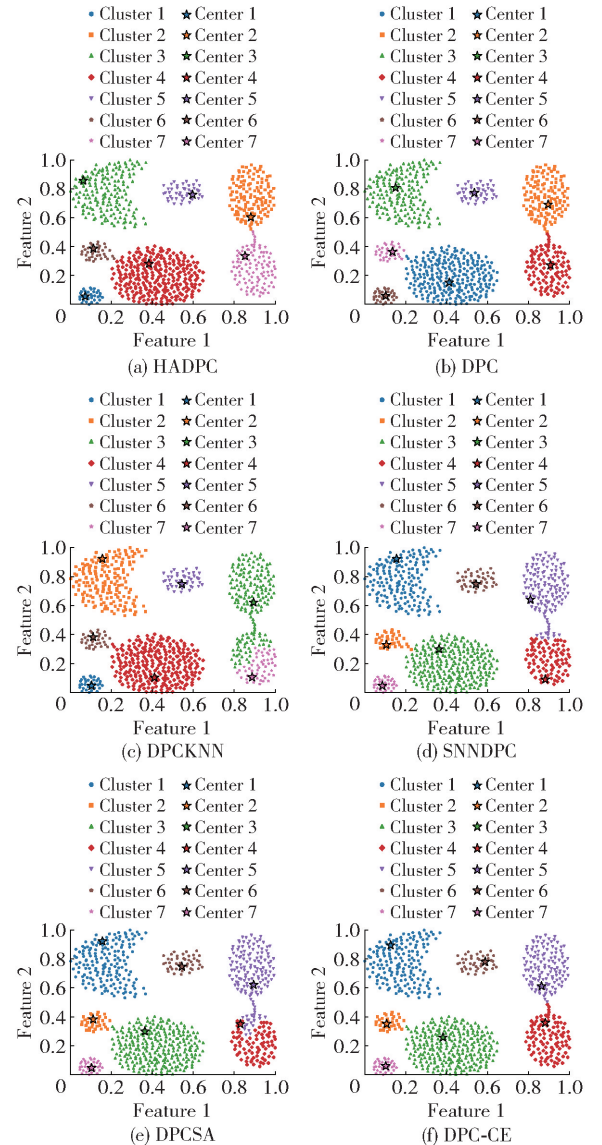


Fig. 9 Clustering results of six algorithms on the Aggregation dataset

Consequently, some data points are misclassified into adjacent clusters, resulting in reduced accuracy of the clustering outcomes.

The DS5 dataset comprises five clusters with varying shapes and densities, exhibiting a complex multi-peak density distribution that places high demands on the adaptability of clustering algorithms. As illustrated in Fig. 10, the HADPC algorithm successfully identifies the center of each cluster and accurately separates them using its adaptive local density calculation method and hierarchical allocation strategy. In contrast, the other comparative algorithms exhibit inadequate adaptability in handling this multi-peak density distribution dataset, failing to accurately locate all clustering centers and resulting in noticeable deviations from the true cluster structure.

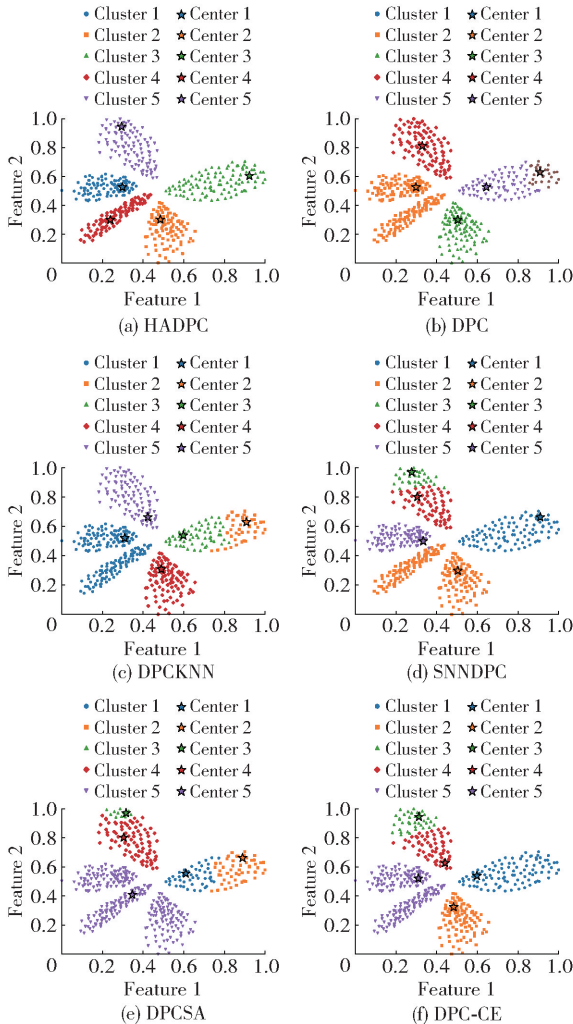


Fig. 10 Clustering results of six algorithms on the DS5 dataset

The SpiralSquare dataset features complex geometric shapes, including manifold-structured spiral clusters and square clusters with overlapping regions, posing challenges for clustering algorithms in shape recognition and boundary delineation. As illustrated in Fig. 11, the HADPC algorithm not only successfully identifies the manifold structure of the spiral clusters but also accurately partitions the overlapping regions of the square clusters. In contrast, the DPC algorithm misjudges the spiral clusters, failing to accurately capture their intricate geometric features. Although SNNDPC accurately identifies the spiral clusters, it struggles to handle the overlapping regions of the square clusters, leading to decreased clustering accuracy.

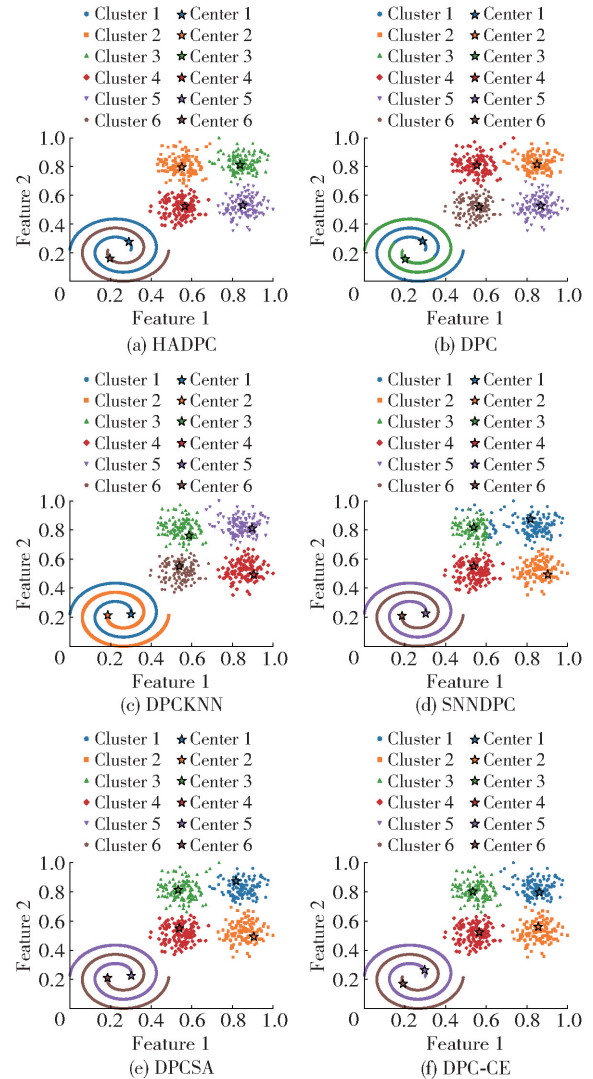


Fig. 11 Clustering results of six algorithms on the SpiralSquare dataset

The Pathbased dataset consists of data points distributed along a complex curved path, including intersections and overlaps between clusters, presenting high demands on the algorithm’s boundary handling capabilities and shape recognition. As shown in Fig. 12, the HADPC algorithm effectively identifies the path structure by partitioning core points and boundary points, accurately assigning boundary points using a weighted voting method. Although SNNDPC can handle data distributed along curved paths, its treatment of intersection areas lacks precision. While DPC, DPC-KNN, and DPCSA accurately identify central points, they exhibit limitations in recognizing the path structure. Furthermore, DPC-CE fails to accurately identify the clustering centers, leading to significant deviations between the clustering results and the true data distribution.

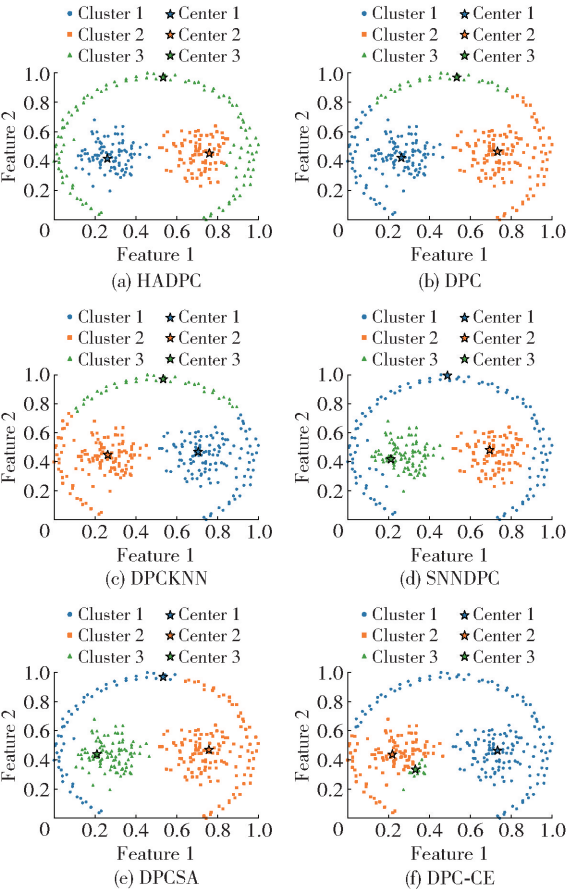


Fig. 12 Clustering results of six algorithms on the Pathbased dataset

4.3 Experiments on UCI dataset

To further evaluate the clustering performance of the

HADPC algorithm against other algorithms on UCI datasets, six representative UCI datasets are selected, with their basic information presented in Table 2. These datasets exhibit significant differences in dimensions, feature quantities, and data shapes, enabling a comprehensive assessment of the HADPC algorithm’s performance from multiple perspectives. Table 4 provides a comparative analysis of the evaluation metrics for various clustering algorithms on these UCI datasets.

Table 4 Comparison of evaluation metrics for different clustering algorithms on UCI dataset

Dataset	Algorithm	V_{AMI}	V_{ARI}	V_{FMI}	Number of parameters
Iris	HADPC	0.723 5	0.651 6	0.773 9	–
	DPC	0.781 0	0.719 6	0.815 9	2
	DPC-KNN	0.883 6	0.903 8	0.935 5	19
	SNNDPC	0.913 3	0.922 2	0.947 9	15
	DPCSA	0.883 6	0.903 8	0.935 5	–
	DPC-CE	0.736 3	0.635 7	0.767 3	–
Wine	HADPC	0.750 1	0.741 4	0.828 3	–
	DPC	0.707 4	0.672 4	0.783 5	2
	DPC-KNN	0.740 7	0.726 9	0.818 8	17
	SNNDPC	0.881 5	0.915 0	0.943 6	28
	DPCSA	0.750 1	0.741 4	0.828 3	–
	DPC-CE	0.779 8	0.769 2	0.846 9	–
Zoo	HADPC	0.851 2	0.884 9	0.911 6	–
	DPC	0.353 6	0.243 0	0.499 2	5
	DPC-KNN	0.283 9	0.111 1	0.342 5	8/15
	SNNDPC	0.842 8	0.876 8	0.907 0	11
	DPCSA	0.808 2	0.847 4	0.885 3	–
	DPC-CE	0.740 7	0.669 0	0.781 2	–
Seeds	HADPC	0.764 9	0.801 3	0.867 1	–
	DPC	0.676 8	0.707 6	0.805 1	4
	DPC-KNN	0.731 9	0.766 4	0.843 9	4
	SNNDPC	0.750 9	0.789 0	0.858 9	6
	DPCSA	0.664 5	0.687 3	0.791 8	–
	DPC-CE	0.718 9	0.762 9	0.841 3	–
Segment	HADPC	0.649 5	0.470 0	0.599 4	–
	DPC	0.570 4	0.362 4	0.526 7	4
	DPC-KNN	0.633 3	0.368 5	0.533 5	36
	SNNDPC	0.595 4	0.454 8	0.556 0	49
	DPCSA	0.688 7	0.392 6	0.558 9	–
	DPC-CE	0.541 8	0.425 9	0.515 3	–
Landsat	HADPC	0.605 1	0.549 5	0.653 9	–
	DPC	0.594 3	0.509 0	0.620 0	4
	DPC-KNN	0.592 7	0.520 9	0.613 6	36
	SNNDPC	0.604 1	0.468 0	0.578 7	45
	DPCSA	0.601 3	0.474 9	0.586 6	–
	DPC-CE	0.605 2	0.506 3	0.601 0	–

4.4 Runtime analysis

This section compares the runtime of DPC, DPC-KNN, SNNDPC, DPCSA, DPC-CE, and HADPC across 12 datasets. These 12 representative datasets are selected from the synthetic and UCI datasets, covering different sample sizes and feature dimensions, to evaluate the runtime efficiency of the algorithms. The algorithms are configured according to the parameters in Tables 3 and 4, with each algorithm independently executed 10 times. The time for center selection via decision graphs is excluded from the runtime calculations, and the average runtime per execution is computed, as shown in Table 5. Because the HADPC algorithm employs natural neighbor search and a multi-step allocation strategy, its time overhead exceeds that of DPC’s single-step sample allocation approach. Although both HADPC and DPC exhibit a time complexity of $O(I^2)$, their actual runtime differ due to algorithmic design variations. HADPC’s runtime on large-scale datasets is higher than that of DPC and DPC-KNN, primarily due to the additional computations of the natural neighbor search algorithm. However, its time consumption remains lower than that of SNNDPC and DPC-CE.

Table 5 Comparison of runtime of different clustering algorithms across combined datasets

Dataset	HADPC	DPC	DPC-KNN	SNNDPC	DPCSA	DPC-CE
Aggregation	0.066 1	0.059 6	0.044 4	2.011 9	0.271 8	2.248 5
Jain	0.030 7	0.014 9	0.011 6	0.494 2	0.052 0	0.557 3
Flame	0.016 1	0.005 3	0.006 4	0.221 5	0.066 0	0.287 5
S2	1.175 9	1.664 5	1.975 1	79.869 8	17.729 8	77.942 1
Pathbased	0.024 0	0.010 7	0.008 6	0.343 7	0.051 3	0.473 8
D31	0.735 8	0.886 1	0.760 5	30.274 2	6.378 9	29.543 5
Iris	0.017 3	0.003 7	0.004 1	0.091 0	0.015 9	0.106 5
Wine	0.041 1	0.008 6	0.005 0	0.130 6	0.023 9	0.183 6
Zoo	0.011 5	0.002 1	0.002 2	0.048 7	0.006 5	0.029 6
Seeds	0.023 7	0.007 6	0.006 5	0.176 3	0.028 6	0.262 2
Segment	7.395 8	1.574 1	0.459 1	17.742 8	4.163 8	17.569 0
Landsat	6.157 4	0.853 6	0.366 1	13.172 0	2.365 6	10.236 5

4.5 Parameter sensitivity analysis

In the HADPC algorithm, core point identification is a critical step in the hierarchical allocation strategy. For data points satisfying $s_{x_i} \leq k^*$, screening is performed by determining whether $s_{x_i} \geq \alpha \eta(x_i)$. This study adopts $\alpha = 0.85$ as the threshold. To evaluate the impact of this parameter on algorithm performance, the variation of α within the range $[0.35, 1.35]$ is tested at intervals of 0.05 across 12 datasets. These 12 representative datasets are selected from the synthetic and UCI datasets, covering different shapes, densities, and cluster structures, to analyze the effects of α on the clustering metrics AMI, ARI, and FMI. Fig. 13 illustrates the trends in clustering metrics under different α values across datasets.

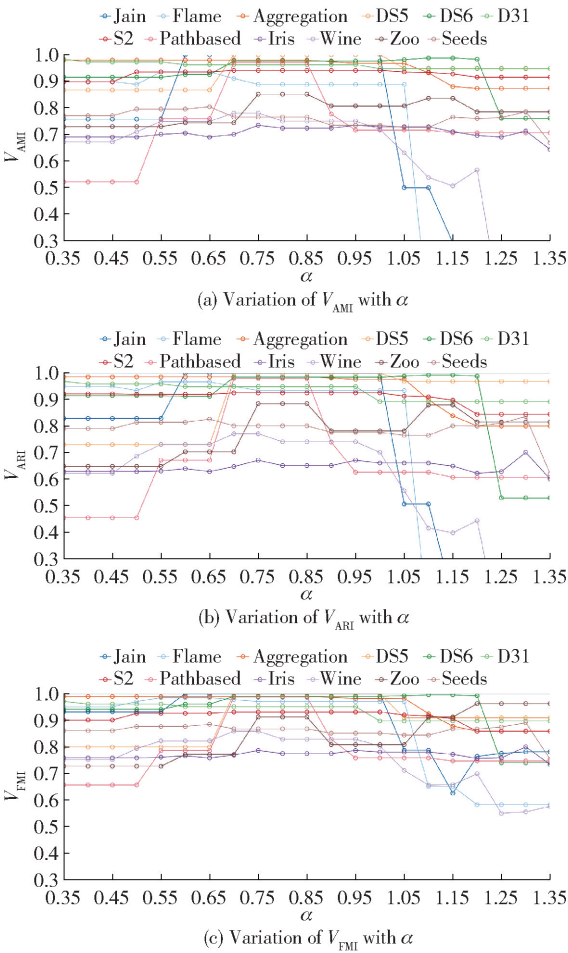


Fig. 13 Clustering metrics of HADPC at different density thresholds α

Experimental results indicate that on datasets with uneven density distributions, such as Jain and

Pathbased, smaller α values lead to unstable clustering results, with excessive core points causing boundary regions or noise points to be misidentified as core points. This affects the cluster structure formation during BFS and increases noise in the weighted voting of boundary points, resulting in lower and more volatile AMI, ARI, and FMI values. As α increases, the number of core points decreases, leading to clearer cluster structures and improved clustering metrics. However, excessively large α values result in too few core points, causing sparse regions to lack sufficient core point support, which leads to misallocation of some data points to high-density clusters, thereby reducing clustering accuracy. Comprehensive experimental results demonstrate that $\alpha = 0.85$ provides a reasonable balance between the number and representativeness of core points across most datasets, providing robust support for the stability and accuracy of the clustering structure.

To evaluate the impact of the parameter k^* on algorithm performance, this study tested variations of k^* within a $\pm 20\%$ range of the natural stable state on the Aggregation and Pathbased datasets, analyzing its effects on clustering metrics. Figs. 14 and 15 illustrate the core point distributions and final clustering results under different k^* values for each dataset, respectively. Experimental results demonstrate that HADPC algorithm exhibits strong robustness to variations in k^* . When k^* fluctuates within the $\pm 20\%$ range, clustering results remain stable, indicating that the natural neighbor search adaptively captures the topological structure of the dataset, rendering clustering outcomes insensitive to deviations in this parameter.

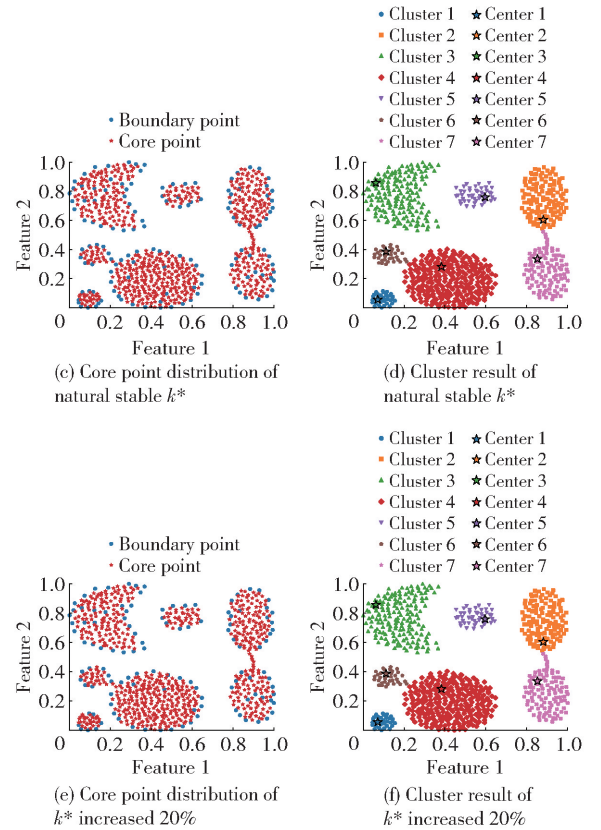
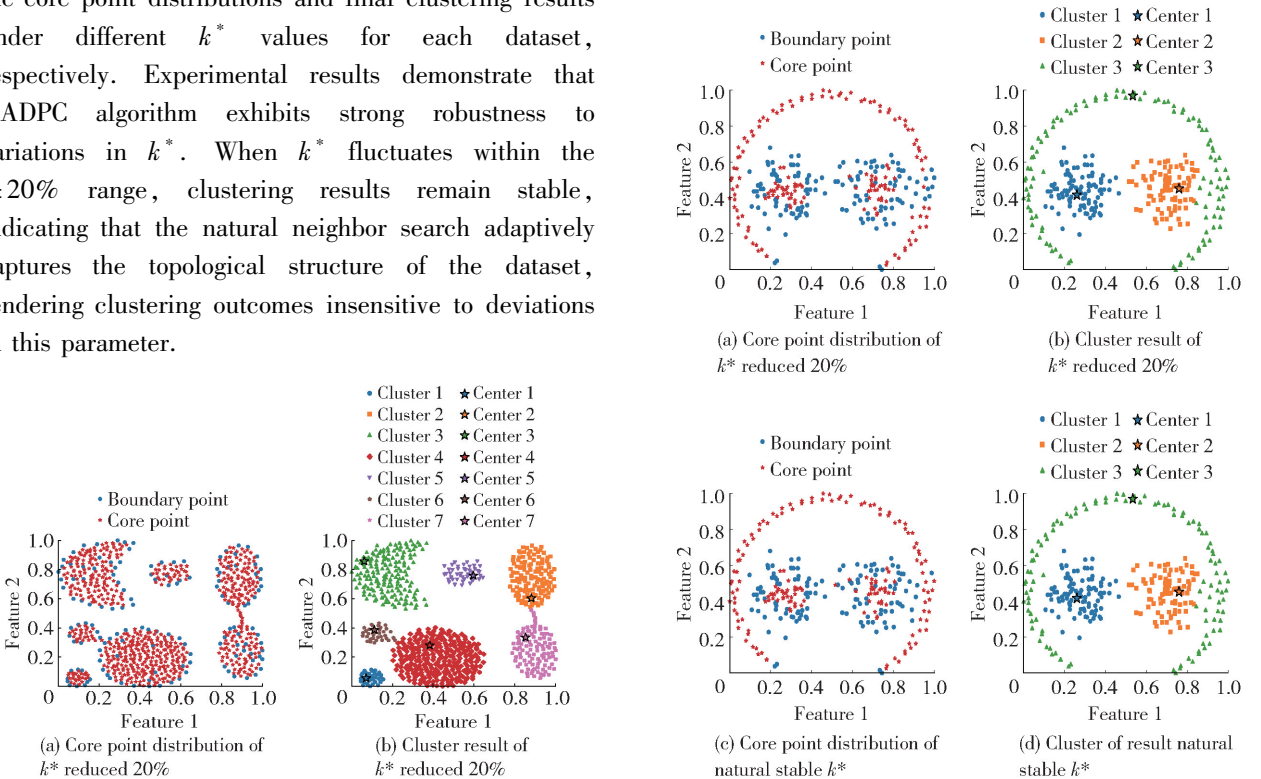


Fig. 14 Core point distribution and clustering results on the Aggregation dataset under different k^* values



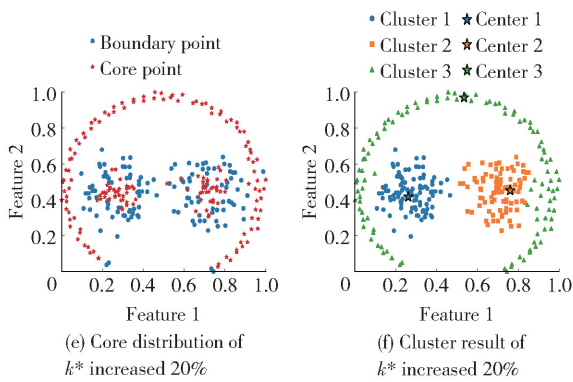


Fig. 15 Core point distribution and clustering results on the Pathbased dataset under different k^* values

5 Conclusions

HADPC algorithm were presented in this paper. The HADPC algorithm redefines local density and employs a two-step allocation method for non-cluster-center samples, thereby enhancing clustering accuracy. During the clustering process, the concept of natural neighbors is introduced to adaptively identify the nearest neighbors, avoiding reliance on global parameters. A weighted distance is used to model density relationships between samples, providing a more faithful representation of the underlying data distribution. Furthermore, a hierarchical allocation strategy based on core and boundary points is proposed. Core points are sequentially assigned to their corresponding clusters using a BFS method, while boundary points are allocated based on a weighted voting mechanism. This mechanism determines the belonging cluster according to the density and distance weights of neighboring points, resulting in more consistent and reasonable assignments of non-cluster-center samples. Experimental results from nine synthetic datasets and six UCI datasets indicate that the HADPC algorithm outperforms the comparison algorithms.

There is still room for improvement in the HADPC algorithm. For instance, the identification of core points currently relies on a fixed threshold which may limit the algorithm's adaptability. To overcome this limitation, an adaptive strategy for selecting this

threshold could be explored. In addition, the clustering accuracy of the algorithm could be further improved when dealing with high-dimensional or noisy datasets. These aspects remain open challenges and directions for future research.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (62273290).

References

- [1] WEI X X, PENG M S, HUANG H J, et al. An overview on density peaks clustering. *Neurocomputing*, 2023, 554: Article 126633.
- [2] LI T S, YANG L J, YANG J T, et al. Non-parameter clustering algorithm based on chain propagation and natural neighbor. *Information Sciences*, 2024, 672: Article 120663.
- [3] BUCZAK A L, GUVEN E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 2016, 18 (2): 1153 – 1176.
- [4] LIEW A W C, YAN H, YANG M S. Pattern recognition techniques for the emerging field of bioinformatics: a review. *Pattern Recognition*, 2005, 38(11): 2055 – 2073.
- [5] IKOTUN A M, EZUGWU A E, ABUALIGAH L, et al. K -means clustering algorithms: a comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 2023, 622: 178 – 210.
- [6] RAN X C, XI Y, LU Y G, et al. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 2023, 56(8): 8219 – 8264.
- [7] KULKARNI O S, BURHANPURWALA A. A survey of advancements in DBSCAN clustering algorithms for big data. *Proceedings of the 3rd International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC'24)*, 2024, Feb 23 – 24, Mathura, India. Piscataway, NJ, USA: IEEE, 2024; 106 – 111.
- [8] HEIDARI J, DANESHPOUR N, ZANGENEH A. A novel K -means and K -medoids algorithms for clustering non-spherical-shape clusters non-sensitive to outliers. *Pattern Recognition*, 2024, 155: Article 110639.
- [9] GUO W J, CHEN W, LIU X G. Density peak clustering by local centers and improved connectivity kernel. *Information Sciences*, 2024, 666: Article 120439.
- [10] CHOWDHURY S, HELIAN N, De AMORIM R C. Feature weighting in DBSCAN using reverse nearest neighbours. *Pattern Recognition*, 2023, 137: Article 109314.
- [11] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks. *Science*, 2014, 344(6191): 1492 – 1496.
- [12] YANG Y Q, CAI J H, YANG H F, et al. Density clustering with divergence distance and automatic center selection. *Information*

- Sciences, 2022, 596: 414 – 438.
- [13] GUAN J Y, LI S, HE X X, et al. Clustering by fast detection of main density peaks within a peak digraph. *Information Sciences*, 2023, 628: 504 – 521.
- [14] DU M J, DING S F, JIA H J. Study on density peaks clustering based on k -nearest neighbors and principal component analysis. *Knowledge-Based Systems*, 2016, 99: 135 – 145.
- [15] LIU R, WANG H, YU X M. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. *Information Sciences*, 2018, 450: 200 – 226.
- [16] LI Y, ZHOU W J, WANG H K. F-DPC: fuzzy neighborhood-based density peak algorithm. *IEEE Access*, 2020, 8: 165963 – 165972.
- [17] ZHAO J, TANG J J, SHI A Y, et al. Improved density peaks clustering based on firefly algorithm. *International Journal of Bio-Inspired Computation*, 2020, 15(1): 24 – 42.
- [18] JIANG D, ZANG W K, SUN R, et al. Adaptive density peaks clustering based on K -nearest neighbor and Gini coefficient. *IEEE Access*, 2020, 8: 113900 – 113917.
- [19] YU H, CHEN L Y, YAO J T. A three-way density peak clustering method based on evidence theory. *Knowledge-Based Systems*, 2021, 211: Article 106532.
- [20] GUO W J, WANG W H, ZHAO S P, et al. Density peak clustering with connectivity estimation. *Knowledge-Based Systems*, 2022, 243: Article 108501.
- [21] ABBAS M, EL-ZOGHABI A, SHOUKRY A. DenMune: density peak based clustering using mutual nearest neighbors. *Pattern Recognition*, 2021, 109: Article 107589.
- [22] GUO L M, QIN W J, CAI Z, et al. Hybrid clustering algorithm based on improved density peak clustering. *Applied Sciences*, 2024, 14(2): Article 715.
- [23] SHI Z C, GUO R Z, ZHAO Z G. An improved hierarchical clustering method based on the k -NN and density peak clustering. *Transactions in GIS*, 2023, 27: 2197 – 2212.
- [24] HAN M, LEE J S. Graph-based density peak merging for identifying multi-peak clusters. *Applied Soft Computing*, 2023, 146: Article 110657.
- [25] DING S F, DU W, XU X, et al. An improved density peaks clustering algorithm based on natural neighbor with a merging strategy. *Information Sciences*, 2023, 624: 252 – 276.
- [26] CHENG D D, ZHU Q S, HUANG J L, et al. A local cores-based hierarchical clustering algorithm for data sets with complex structures. *Neural Computing and Applications*, 2019, 31(11): 8051 – 8068.
- [27] YU D H, LIU G J, GUO M Z, et al. Density peaks clustering based on weighted local density sequence and nearest neighbor assignment. *IEEE Access*, 2019, 7: 34301 – 34317.

(Editor: Wang Xuying)