Neurocomputing 455 (2021) 401–418

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Fast hierarchical clustering of local density peaks via an association degree transfer method

Junyi Guan [a], Sheng Li [a,*], Xiongxiong He [a], Jinhui Zhu [b], Jiajia Chen [a]

[a] College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China
[b] Second Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou 310009, China

ARTICLE INFO

Article history:
Received 27 November 2020
Revised 17 May 2021
Accepted 21 May 2021
Available online 25 May 2021
Communicated by Zidong Wang

2010 MSC:
00-01
99-00

Keywords:
Clustering
Density peak
Hierarchical clustering
KNN

ABSTRACT

Density Peak clustering (DPC) as a novel algorithm can fast identify density peaks. But it comes along with two drawbacks: its allocation strategy may produce some non-adjacent associations that may lead to poor clustering results and even cause the malfunction of its cluster center selection method to mistakenly identify cluster centers; it may perform poorly with its high complex $O(n^2)$ when comes to large-scale data. Herein, a fast hierarchical clustering of local density peaks via an association degree transfer method (FHC-LDP) is proposed. To avoid DPC's drawbacks caused by non-adjacent associations, FHC-LDP only considers the association between neighbors and design an association degree transfer method to evaluate the association between points that are not neighbors. FHC-LDP can fast identify local density peaks as sub-cluster centers to generate sub-clusters automatically and evaluate the similarity between sub-clusters. Then, by analyzing the similarity of sub-cluster centers, a hierarchical structure of sub-clusters is built. FHC-LDP replaces DPC's cluster center selection method with a bottom-up hierarchical approach to ensure sub-clusters in each cluster are most similar. In FHC-LDP, only neighbor information of data is required, so by using a fast KNN algorithm, FHC-LDP can run about $O(nlog(n))$. Experimental results demonstrate FHC-LDP is remarkably superior to traditional clustering algorithms and other variants of DPC in recognizing cluster structure and running speed.

1. Introduction

Clustering, aiming to automatically classify similar points into groups (clusters), is an important data analysis tool for exploring knowledge in complex data of many fields including pattern recognition, data mining, image processing, etc [1,2].

Numerous clustering algorithms have been proposed based on specific assumptions regarding the nature of a 'cluster' [3,4]. For example, partitioning clustering K-means [5], one of the most well-studied clustering algorithms, defines a cluster as a group of data characterized by a small distance to a cluster center. It can effectively detect spherical clusters with a suitable initial setting but cannot well recognize clusters with arbitrary shapes. Hierarchical clustering method [6] that uses the similarity matrix of points in the dataset to build a dendrogram is helpful to understand the dataset structure but is sensitive to outliers.

Density-based methods are outstanding in detecting clusters with arbitrary shapes as well as eliminating outliers (noise). For example, classical DBSCAN [7] reconstructs arbitrary-shaped clusters and removes noise according to a density-connectivity criterion. However, setting a globally optimal density-connectivity threshold is difficult, especially for datasets consist of overlapping clusters. Density peak clustering method (DPC) [8] effectively separates highly overlapping clusters by initially searching density peaks. This allows DPC to be widely applied [9,10]. Nevertheless, drawbacks still exist, for example, its allocation strategy of remaining points is not robust; its identification method $(\gamma = \rho \times \delta)$ of cluster centers may mislead the selection of cluster centers; its high complexity of $O(n^2)$ is not suitable for large scale datasets.

Several methods have been developed to overcome DPC's drawbacks. SNN-DPC [11], FKNN-DPC [12], DPC-GD [13] and SSSP-DPC [14] were developed to reform the allocation strategy; ADPC-KNN [15], CDMC-IA [16], McDPC [17], and HCDP [18] were dedicated to the accurate identification of cluster centers. However, since optimization work always adds additional calculation, these methods all have high time complexities, for which a fast version of DPC (FastDPeak) was proposed [19], which lowered the time

* Corresponding author.
E-mail addresses: shengli@zjut.edu.cn (S. Li), hxx@zjut.edu.cn (X. He), 2512016@zju.edu.cn (J. Zhu).

https://doi.org/10.1016/j.neucom.2021.05.071
0925-2312/© 2021 Elsevier B.V. All rights reserved.

complexity to $O(nlog(n))$, and inherited all other performance of DPC, as well as the defects.

In this paper, we introduce a fast hierarchical clustering of local density peaks via an association degree transfer method (FHC-LDP) which first fast identifies local density peaks as sub-cluster centers to form sub-clusters and then uses a hierarchical clustering method to merge sub-clusters into final clusters optimally in global. The main contributions of this paper are: 1) FHC-LDP is a fast algorithm with a time complexity of $O(nlog(n))$ and it only calculates the KNN distance of points; 2) we design a method to fast evaluate the similarity between sub-clusters; 3) we use a hierarchical clustering method to ensure each sub-cluster is accurately merged into its most similar cluster, which makes up for the deficiencies of DPC's allocation strategy and center selection.

The rest paper is organized as follows: (1) Section 2 introduces DPC's improved works and the hierarchical clustering method. (2) Section 3 gives a detailed introduction to the DPC algorithm. (3) Section 4 mainly focuses on our proposed method. (4) Section 5 launches comparison experiments on the proposed method and some well-known traditional clustering methods and variants of DPC. (5) Section 6 ends the paper with an overall conclusion.

2. Related work

2.1. DPC's improved work

Different methods have been developed to overcome the drawbacks of DPC. The first group of methods attempted to reform the allocation strategy of DPC. In DPC-GD [13], geodesic distance was used to change the allocation strategy. SSSP-DPC [14] proposed a shortest-path-based allocation strategy. SNN-DPC [11] and FKNN-DPC [12] took labels of neighbors into consideration in the allocation of remaining points. These improved methods improved the allocation strategy of DPC while inheriting DPC's cluster center acquisition method ($\gamma = \rho \times \delta$).

The second group of methods tried to improve the cluster center recognition method of DPC. These methods first generated sub-clusters through DPC and then merged sub-clusters into final clusters according to a special merging criterion. For example, ADPC-KNN [15] designed an adaptive method based on KNN and density to judge the density reachability between sub-clusters; CDMC-IA [16] merged sub-clusters via independence and affinity; McDPC [17] first divided sub-clusters into different density levels, and then merged sub-clusters of similar density levels into final clusters. However, the complexity level of all the abovementioned methods was high, because the optimization work always added additional calculation.

FastDPeak, a fast version of DPC, proposed by Chen et al. [19] used KNN-density that based on cover-tree [20] to fast evaluate the density of each point, and designed two different strategies to compute δ of local density peaks and non-local density peaks to reduce the calculation of δ. FastDPeak reduced the complexity to $O(nlog(n))$ without affecting DPC's performance for large-scale dataset clustering. However, FastDPeak failed to improve other defects of DPC.

2.2. Hierarchical clustering

Hierarchical clustering [6] builds a cluster hierarchy (namely a tree of clusters). Each cluster node has its sub-cluster nodes (child nodes); all points covered by parent nodes are divided by its child nodes. Hierarchical clustering needs a final determination criterion of clusters compared with other clustering methods that directly give a clustering result. In hierarchical clustering methods, the distance (similarities or dissimilarities) between each pair of points is usually used as input, so the association information between points is well taken into account, which also leads to high computational complexity.

Single-linkage clustering [21], a common method of hierarchical clustering, is a bottom-up grouping method based on the similarity between each pair of elements. It is a step-by-step implementation, in which each step merges a pair of clusters that have the closest (most similar) pair of elements not yet belonging to the same cluster as each other.

Fig. 1 simply illustrates the clustering process of the single-linkage method, where ten points noted as $A \sim J$ form three clusters. Based on the closest distance of each pair of points, the single-linkage method builds a bottom-up tree of clusters, called a dendrogram. In the dendrogram, each branch represents a cluster. So, to get three clusters, we only need to cut the dendrogram into tree branches along the horizontal direction.

In the single-linkage method, the points cohesion in each cluster is well-considered, however, the computational complexity $O(n^2)$ is high. Moreover, its allocation strategy is irreversible and sensitive to noise (outlier).

3. DPC algorithm and its analysis

3.1. DPC algorithm

DPC is based on the assumption that cluster centers are characterized by a higher density ρ than their neighbors and by a relatively large distance δ from points with higher density [8].

For each point i, DPC calculates two quantities: the local density ρ_i as in Eq. (10) (or Eq. (10)), and the distance δ_i from its nearest point with a higher density as in Eq. (10), where d_{ij} is the Euclidean distance between point i and point j, while d_c is a cutoff distance. For the highest density point, its distance is conventionally defined as $\delta_i = \max_{i \neq j}(d_{ij})$. In addition, DPC stipulates that each point must inherit the label of its nearest point with a higher density. In other words, DPC builds a bottom-up tree structure for a specific dataset, where each node i (namely each point) regards its nearest point j with a higher density as its parent node, denoted $PN(i)$ as defined

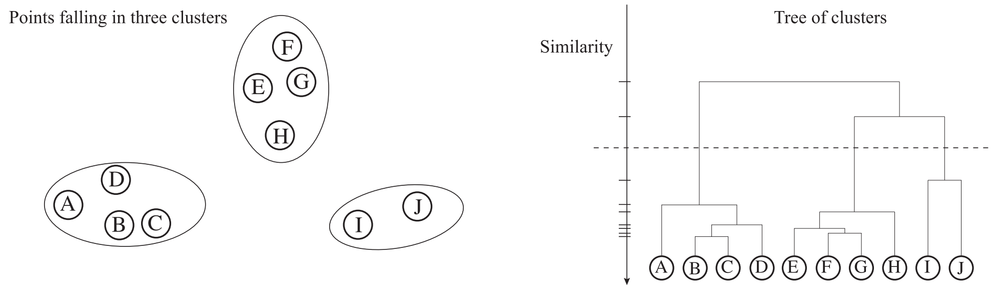


Fig. 1. The clustering idea of the single-linkage method.

in Eq. (10) (note: if there are multiple parent nodes that meet Eq. (10), DPC will conventionally select the parent node with the smallest density ranking order number in the reverse order of density values to ensure that the parent node of each point is unique).

Then, cluster centers can be fast found by selecting the first C number of points with the largest γ (i.e., $\gamma = \rho \times \delta$). After cluster centers are located and given unique labels, the remaining points inherit the labels of their parent nodes to complete the clustering. Fig. 2 further illustrates the core idea of DPC in detail.

$$\rho_i = \sum_{j \neq i} \chi(d_{ij} - d_c), \chi(x) = \begin{cases} 1, x < 0 \\ 0, x \geqslant 0 \end{cases} \tag{1}$$

$$\rho_i = \sum_{i \neq j} e^{-\left(\frac{d_{ij}}{d_c}\right)^2} \tag{2}$$

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{3}$$

$$PN(i) = \arg\min_{j:\rho_j > \rho_i} (d_{ij}) \tag{4}$$

Fig. 2 shows 24 points of a toy dataset *E1* embedded in a two-dimensional space (point number indicates density ranking order). After each point searching for its nearest point with a higher density as its parent node, a tree structure is built, where point 1 with the highest density is the root node. Based on the assumption that a cluster center should have a high density and a relatively large distance from its parent node, by using the γ method, DPC selects points 1 and 2 as cluster centers and gives each a unique label. Finally, each remaining point inherits the label of its parent node to complete clustering.

3.2. Analysis

DPC's contribution is remarkable, however, in addition to high complexity, its drawbacks are obvious.

3.2.1. Allocation strategy

DPC's original allocation idea is that each non-center point should be assigned to the same cluster as its nearest neighbor of higher density [8]. For neighbors, DPC usually considers that the average number of neighbors should around 1% to 2% of the total number of points in the dataset [8]. For a d-dimensional dataset, we define $S_r(i)$ as a d-dimensional sphere area with point i as the center point and radius r. Let ε_i be a neighborhood radius of point i, and let Δ_i be the maximum radius of point i that makes i be the density maximum in $S_{\Delta_i}(i)$. Then, we have:

Definition 1. $S_{\varepsilon_i}(i)$ is i's neighborhood space, if $j \in S_{\varepsilon_i}(i)$, then j is a neighbor of i, otherwise, j is a non-neighbor of i.

Property 1. $S_{\delta_i}(i) \supset S_{\Delta_i}(i)$, i.e., $\delta_i > \Delta_i$.

Discussion 1. Because δ_i is the minimum distance from i to the point with a higher density, therefore $\exists\, j \in S_{\delta_i}(i), \rho_j > \rho_i$. Because $\forall\, j \in S_{\Delta_i}(i), \rho_j < \rho_i$, therefore, $S_{\delta_i}(i) \supset S_{\Delta_i}(i)$, i.e., $\delta_i > \Delta_i$. *Property 1* implies that usually δ_i is only slightly larger than Δ_i (i.e., usually $\delta_i \approx \Delta_i$), because there is only one more point $j : \rho_j > \rho_i$ in $S_{\delta_i}(i)$ than in $S_{\Delta_i}(i)$ (i.e., $S_{\delta_i}(i)/S_{\Delta_i}(i)) = j$).

According to *Property 1*, we can have *Property 2*.

Property 2. if $\Delta_i > \varepsilon_i$, then $\delta_i > \varepsilon_i$.

Property 2 tells that when point i's $S_{\Delta_i}(i)$ is larger than its neighborhood space $S_{\varepsilon_i}(i)$, its parent node is no longer a neighbor of i and may even far beyond the neighborhood of i. This situation violates DPC's original allocation idea. In fact, DPC's allocation strategy never bothers whether a point and its parent node are neighbors, leading to the case where such distance δ_i between the point i and its parent node is larger than its neighborhood radius ε_i, i.e., $\delta_i > \varepsilon_i$, but DPC still regards them as most associated.

Distance is an important indicator in analyzing the association of two points, but DPC ignores it. As a result, DPC's allocation strategy may unadvisedly associate irrelevant points. Fig. 3 demonstrates the allocation strategy drawback of DPC in dealing with the classic *Jain* dataset [22].

In Fig. 3(a), *Jain* with 373 points is clearly composed of two crescent-shaped clusters, where points A and A' are not in the same cluster. Setting the number of neighbors as 8 (about 2% of the total number of *Jain*), we obtain point A's neighborhood as $S_{\varepsilon_A}(A)$ where $\varepsilon_A = d_{AE}$, and $A' \notin S_{\varepsilon_A}(A)$ is not a neighbor of A. However, DPC views point A' as the parent node of A (i.e., $PN(A) = A'$), that is, points A and A' are most associated. As a result, as shown in Fig. 3(b), with a correct selection of cluster centers, point A still has to inherit a wrong label from point A', which causes a "domino effect" [12] in point A's descendant nodes.

This is because point A's S_{Δ_A} is larger than its neighborhood $S_{\varepsilon_A}(A)$ (i.e., $\Delta_A > \varepsilon_A$), so it has to find its parent node A' far beyond its neighborhood (i.e., $\delta_A > \Delta_A > \varepsilon_A$, thus A' is a non-neighbor of A). This leads to an uncertain relationship between point A and A'. In this example, point A and A' are not even in the same cluster, which directly leads to the misclassification of A. Worse still, the large $S_{\Delta_A}(A)$ gives point A many descendant nodes, causing more misallocations of points.

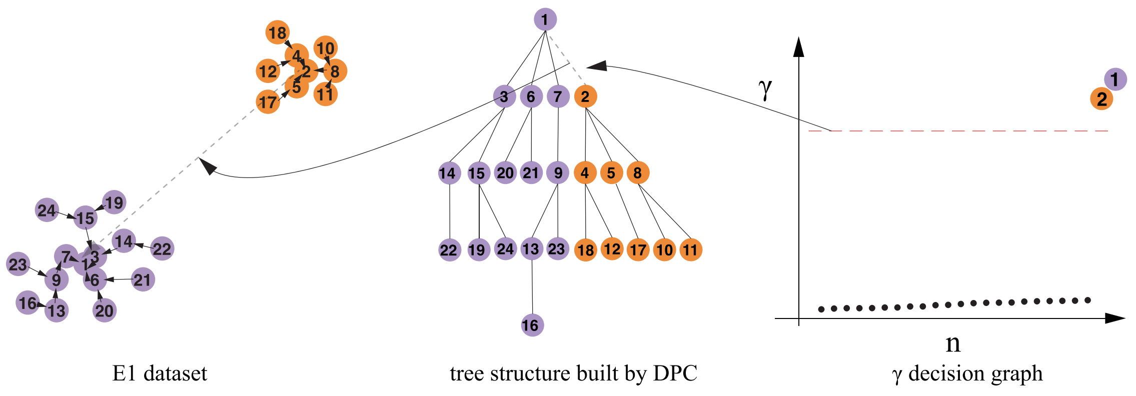


Fig. 2. DPC's clustering idea.

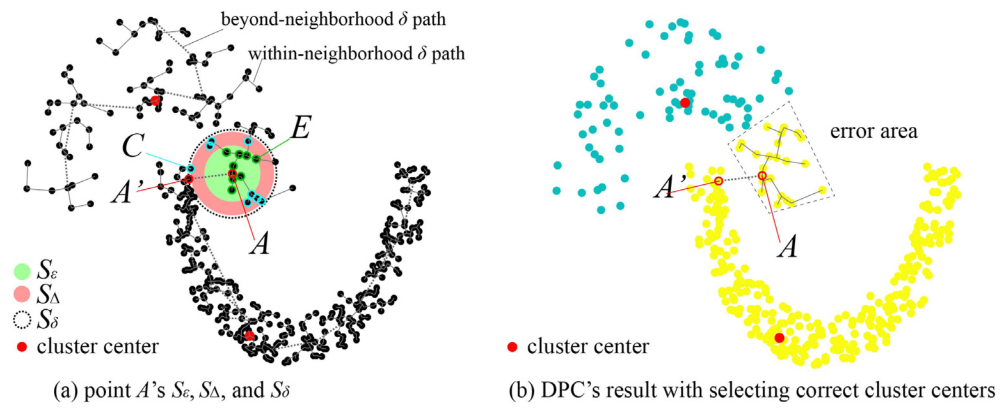

beyond-neighborhood δ path
within-neighborhood δ path
C
E
A'
A
Sε
SΔ
Sδ
cluster center
(a) point A's Sε, SΔ, and Sδ
error area
A'
A
cluster center
(b) DPC's result with selecting correct cluster centers

Fig. 3. The limitation of DPC's allocation strategy in dealing with the *Jain* dataset.

The above example shows that once $\Delta_i > \varepsilon_i$, DPC will unadvisedly associate point *i* with its non-neighbor points, which is unreliable and may lead to further misclassification.

3.2.2. γ method

In DPC, the selected cluster centers should have the largest γ (i.e., $\gamma = \rho \times \delta$) in global. For a point *i*, according to *Property 1*, δ_i is related to the radius Δ_i of its $S_{\Delta_i}(i)$ area. At the same time, point *i* is also density maximum in $S_{\Delta_i}(i)$. Therefore, it can be inferred that γ_i is related to $S_{\Delta_i}(i)$: a dense and large scale $S_{\Delta_i}(i)$ (i.e., a dense-large S_Δ) makes a big γ_i. Based on this, in DPC's γ method, if a cluster center is accurately identified, it had a dense-large S_Δ in global.

However, a cluster is a set of associated points in a certain local area of the dataset. Cluster center as density maximum in the cluster should have a relatively dense-large S_Δ in local, but this is not necessarily the case in global. In other words, if a cluster is not both dense and large in global, the S_Δ of its cluster center may also not dense-large (conspicuous) in global, which may in return cause a small γ. As a result, cluster centers with small γ may be misselected by DPC's γ method. Fig. 4 illustrates the limitation of DPC's γ method in dealing with the *Jain* dataset.

In Fig. 4(a), the *Jain* dataset has two clusters of different densities, where the density of the upper cluster is much lower than that of the bottom cluster, and the two cluster centers are denoted as *H* and *G*. Although these two clusters are similar in shape, their cluster centers differ completely in γ value (i.e., $\gamma_H \gg \gamma_G$) as in Fig. 4(c). Consequently, as shown in Fig. 4(d), DPC fails to accurately identify *H* and *G* as cluster centers, instead it recognizes *H* and *L* (a local density maximum in the bottom cluster) with the largest γ_H and γ_L as cluster centers of the *Jain* dataset.

The reason lies in the difference between S_Δ regions. In Fig. 4(a, b), S_Δ is marked with a color-coded density circle region (red color means high density, and yellow color means low density). We note that $S_{\Delta_L}(L)$ in the bottom cluster is dense, while $S_{\Delta_G}(G)$ in the sparse upper cluster is sparse, which leads to $\rho_L > \rho_G$; simultaneously, $S_{\Delta_L}(L)$ is also larger than $S_{\Delta_G}(G)$, which leads to $\delta_L > \delta_G$ (i.e., $d_{LL'} > d_{GG'}$). Thus, $\gamma_L > \gamma_G$. As a result, DPC selects *L* as cluster center rather than the real cluster center *G*, as in Fig. 4(c).

The above example verifies that a cluster center's γ is a local indicator that closely related to its cluster, but it is not suitable to use γ as a global indicator of cluster center selection, especially not for complex datasets.

4. The FHC-LDP algorithm

With the intention to fully and extensively improve DPC, we propose FHC-LDP, which not only avoids both the defects of DPC's allocation strategy and cluster center selection method, but also decreases the time complexity to $O(n\log(n))$. FHC-LDP performs clustering in three steps: 1) fast generate reliable sub-clusters; 2) fast evaluate similarities between sub-clusters; 3) hierarchical clustering on sub-clusters. In what follows, FHC-LDP will be fully introduced.

4.1. Fast generation of reliable sub-clusters

DPC's allocation strategy may unadvisedly associate with irrelevant non-adjacent points to cause some unpleasant chain errors. In order to ensure associated points are neighbors, we introduce a fixed concept of the neighborhood as a necessary condition to DPC's allocation strategy.

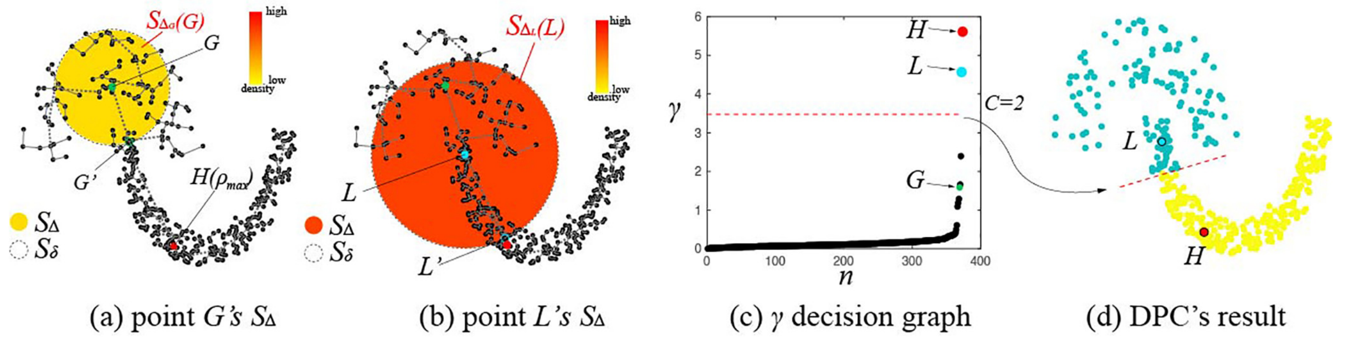


Fig. 4. the limitation of DPC's γ method in dealing with the *Jain* dataset.

In our method, for each point i, we define its k nearest neighbors as its fixed neighborhood $N_k(i) = \left\{i^1, i^2, \dots i^k\right\}$, where i^k indicates the kth nearest neighbor of point i. The neighborhood space of $N_k(i)$ is $S_{\varepsilon_i}(i)_{\varepsilon_i = d_{ii^k}}$, where d_{ii^k} is the Euclidean distance between point i and its kth nearest neighbor i^k. In order to ensure that associated points are real neighbors, each point i is only allowed to associate with its nearest neighbor of higher density within its neighborhood, we call this a direct association of point i. Consequently, points with the maximum density in their neighborhood (defined as local density peaks) having no neighbor of higher density within their neighborhood cannot complete the direct association.

To generate sub-clusters, we first consider all local density peaks as sub-cluster centers (denoted as *SC*) with unique labels and then let each of the remaining points (points that are not sub-cluster centers) inherit the label of its nearest neighbor with a higher density. After obtaining labels, points of the same label automatically form sub-clusters.

Since the direct association is executed within the neighborhood based on KNN, to avoid adding extra distance calculation, we choose the KNN-density method [19] to evaluate the local density as in Eq. (10). The generation of sub-clusters in our method only needs kNN distances of data points, which can be calculated by fast KNN search technology (such as cover-tree [20]).

$$\rho_i^k = \frac{1}{d_{ii^k}} \tag{5}$$

$$NPN(i) = \underset{j: j \in N_k(i), \rho_j^k > \rho_i^k}{\arg\min} (d_{ij}) \tag{6}$$

Similar to DPC's allocation strategy that essentially constructs a dataset into a tree structure, the direct associations of data points also construct a dataset into multiple tree structures with sub-cluster centers as root nodes, where each node i (namely each point) regards its nearest neighbor j with a higher density as its neighbor-parent node that denoted as $NPN(i)$, as defined in Eq. (10). Since point i is directly associated with $NPN(i)$, we define the path between them as a direct association path. If $NPN(i) = \varnothing$, then point i is a sub-cluster center.

For example, Fig. 5 illustrates the generation of sub-clusters by FHC-LDP on the *Jain* dataset. In Fig.5, FHC-LDP divides the dataset into 14 sub-clusters (Fig. 5(a)), where we zoom in on sub-cluster 4 to show the details of its generation process (Fig. 5(b)). In Fig. 5(b), 22 points are embedded in a two-dimensional space (point number indicates density ranking order), where each point searches for its *NPN* within its neighborhood, e.g., point 22 regards point 18 as its *NPN* within its neighborhood $N_k(22)$ (marked by red area). As a result, point 1 having no *NPN* within its neighborhood $N_k(1)$ (marked by gray area) is adaptively considered as a sub-cluster center and automatically labeled. Finally, after each non-center point inheriting the label of its *NPN*, these 22 points automatically form a sub-cluster with point 1 as the sub-cluster center, or a tree structure with point 1 as the root node, where each point (node) is the child node of its *NPN*.

In the following part, the reliability of the associated points will be analyzed in detail based on the idea of the mean-shift method [23].

4.1.1. Reliability analysis

According to the mean-shift method, a point should shift towards a dense region in its proximity. By defining the proximity of each point i as a range within its neighborhood $N_k(i)$, we have:

Definition 2. each point i should shift towards a dense region in its proximity within its neighborhood $N_k(i)$.

According to *Definition 2*, we can deduce *Property 3*:

Property 3. if $j \in N_k(i)$ is i's nearest neighbor of higher density (i.e., $NPN(i) = j$), then i shifts towards j.

Discussion 2. For point i, its neighbor-parent node j is within the nearest denser region to it, therefore, according to *Definition* 2, i should shift towards j. *Property* 3 implies that each point and its neighbor-parent node (if exist) are in the same cluster. This proves that the direct association between each point and its neighbor-parent node is reliable. In addition, it can be deduced that the sub-clusters formed by directly associated points in our allocation strategy are also reliable.

In DPC's allocation strategy, if a sub-cluster center (namely, a local density peak) is not selected as a cluster center, it will be associated with its closest point with a higher density. However, as analyzed in Section 3.2.1, since a sub-cluster center i is density maximum in its neighborhood, that is $\Delta_i > \varepsilon_i$, its closest point with a higher density is beyond its neighborhood and maybe irrelevant. Moreover, if a sub-cluster center is misclassified by an irrelevant association, total points in its sub-cluster will be misclassified. Hence, to avoid such a bad situation, we design a method to fast

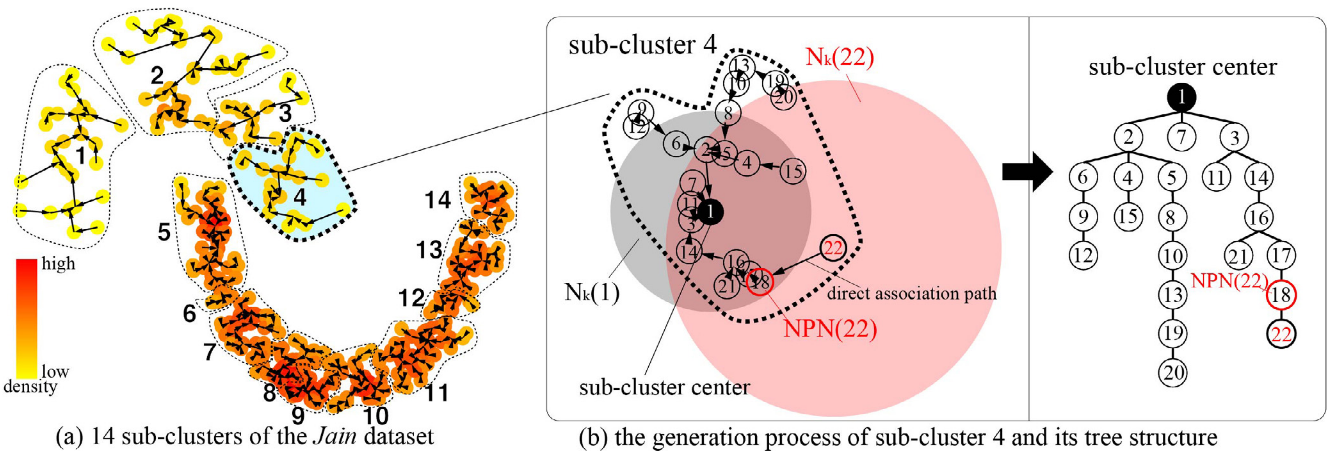


Fig. 5. The generation of sub-clusters by FHC-LDP ($k = 12$) on the *Jain* dataset.

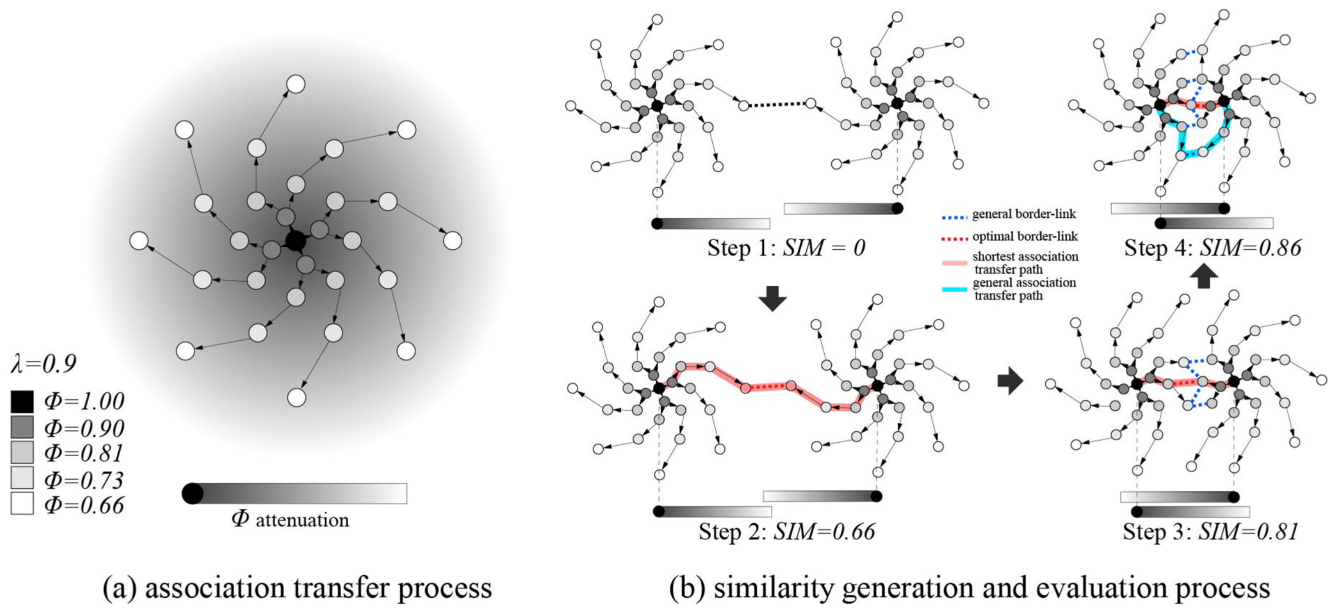


Fig. 6. The association-transfer method.

evaluate the similarity (cohesion) between sub-clusters and use a hierarchical clustering method to ensure that each sub-cluster is merged into the cluster that is most similar to it.

4.2. Fast evaluation of similarities between sub-clusters

Since a sub-cluster center is a representative instance [24] of its sub-cluster, we can analyze the similarity between two sub-clusters by evaluating the similarity between the sub-cluster centers. Herein, we design an association-transfer method to fast and reliably evaluate the similarity between sub-cluster centers.

4.2.1. Association-transfer method

The direct association between a point and its neighbor-parent node is reliable but not absolute, so an association degree ϕ is necessary (ϕ is very close to 1, 1= 100% associated). In our association-transfer method, association degree can transfer in the association path (i.e. a path composed of direct association paths) as in *Definition 3*:

Definition 3. if point i and its neighbor-parent node i' has an association degree of $\phi(i,i') = \alpha$, and i' and its neighbor-parent node i'' has an association degree of $\phi(i',i'') = \alpha'$, then i and i'' has an association degree of $\phi(i,i'') = \alpha \times \alpha'$.

According to *Definition 3*, we suppose the association degree $\phi(i, NPN(i))$ between each point i and its neighbor-parent node $NPN(i)$ is a fixed constant λ close to 1 (usually $\lambda = 0.9$), by using association-transfer method, we can evaluate the association degree $\phi(i,j)$ between a point i with any point j of its descendant or ancestor nodes as Eq. (10), where $\tau(i,j)$ is the total number of direct association paths between i and j.

$$\phi(i,j) = \lambda^{\tau(i,j)}, i,j \text{ are in the same association path.} \quad (7)$$

4.2.2. Similarity between sub-clusters

Following the principle of association within the neighborhood, we have the following definitions:

Definition 4. direct-similarity exists between i and j, if $i \in N_k(j), j \in N_k(i)$, i.e., point i and point j are mutual-neighbors [25].

Definition 5. point i and point j are mutual-border points of sub-cluster SC_p and sub-cluster SC_q, if $i \in SC_p$ and $j \in SC_q$ are mutual-neighbors, i.e., $i \in SC_p \cap N_k(j), j \in SC_q \cap N_k(i)$.

Definition 6. if sub-clusterSC_p and sub-cluster SC_q have mutual-border points, similarity exists between SC_p and SC_q, otherwise, SC_p and SC_q are not similar.

In addition, if two sub-clusters have similarity, we call them intersecting sub-clusters, otherwise, non-intersecting sub-clusters. Since a sub-cluster center is the density maximum in its neighborhood, according to *Definition 4*, we can deduce *Property 4*:

Property 4. if i and i' are different sub-cluster centers, then i and i' are not mutual-neighbors.

Proof 1. Because i is a sub-cluster center, therefore $\forall j \in N_k(i), \rho_i^k > \rho_j^k$. Suppose $i' \in N_k(i)$, since $i \neq i'$, thus $\rho_i^k > \rho_{i'}^k$. Because i' is also a sub-cluster center, therefore, $\forall j' \in N_k(i'), \rho_{i'}^k > \rho_{j'}^k$. Since, $\rho_i^k > \rho_{i'}^k$, so $i \notin N_k(i')$. This implies that i and i' are not mutual-neighbors. Similarly, suppose $i \in N_k(i'), i$ and i' are not yet mutual-neighbors. Summarizing, sub-cluster center i and i' are not mutual-neighbors. □

Property 4 tells that there is no direct-similarity between different sub-cluster centers. However, a sub-cluster center is related to all points in its sub-cluster, and points of different sub-clusters may have direct-similarities. Hence, we apply our association-transfer method to evaluate the similarity (indirect-similarity) between sub-cluster centers.

For sub-cluster SC with sc as center, based on the tree structure (see Fig. 5(b)) analysis of sub-cluster SC, we have *Property 5*:

Property 5. For a center sc in sub-cluster SC, if $i \in SC, \tau(sc,i) = \omega_i$, where ω_i (as in Eq. (10)) is the depth value of i in the tree structure of SC.

Discussion 3. Because *sc* is the root node of the tree structure *SC*, the depth value ω_i of $i_{\in SC}$ indicates the length of path from root node *sc* to *i*, which is exactly equal to the total number of the direct association paths from *sc* to *i*, therefore i.e., $\tau(sc, i) = \omega_i$.

$$\omega_i = \begin{cases} \omega_{NPN(i)} + 1, & NPN(i) \neq \varnothing \\ 0, & NPN(i) = \varnothing \end{cases} \tag{8}$$

$$\Phi(i) = \phi(sc, i) = \lambda^{\omega_i}, i \in SC \tag{9}$$

According to Property 5, we can use ω_i to fast calculate the association degree $\Phi(i) = \phi(sc, i)$ between sub-cluster center *sc* and point *i* as in Eq. (10), called the center-association degree of *i*.

By using the association-transfer method of sub-cluster centers, we can get a center-association degree Φ between each point and its sub-cluster center. Then, by analyzing the Φ values of mutual-border points, the similarity between two sub-clusters can be evaluated. In the association transfer process, Φ value attenuates as the transfer time increases, and since similarity is positively correlated with association degree, we fast evaluate the similarity (*SIM*) between sub-clusters by searching for their mutual-border points that have the max sum of Φ values, as in Eq. (10):

$$SIM(SC_p, SC_q) = \frac{1}{2}\max(\Phi(i) + \Phi(j)), i \in SC_p \cap N_k(j), j \in SC_q \cap N_k(i) \tag{10}$$

Fig. 6 intuitively demonstrates the performance of the association-transfer method and similarity evaluation method on toy datasets. Fig. 6(a) shows the association transfer process of a center, where Φ will attenuate exponentially with the number ω of transfer time (i.e., $\Phi = \lambda^{\omega}$) increases. In other words, the smaller Φ of a point, the less it can represent the center.

Fig. 6(b) demonstrates the generation and evaluation of similarity during the gradual intersection of two clusters in our method, where:

Step 1(non-intersecting): no mutual-border points pair, $SIM = 0$.

Step 2 (mild-intersecting): one pair of mutual-border points, $SIM = 0.66$.

Step 3 (moderate-intersecting): 5 pairs of mutual-border points, $SIM = 0.81$.

Step 4 (high-intersecting): 7 pairs of mutual-border points, $SIM = 0.86$.

It is noteworthy that in intersecting clusters, the mutual-border points with the max sum of Φ (red dotted line) exactly links the shortest associated transfer path (red line) between cluster centers, and the other pairs of mutual-border points (blue dotted line) draw a longer associated transfer path (blue line). Herein, we call the mutual-border points with the max sum of Φ as the optimal border-link and the remaining pairs of mutual-border points as the general border-links.

4.3. Hierarchical clustering of sub-clusters

After obtaining the similarity values, we use the single-linkage clustering method to merge sub-clusters into *C* final clusters from bottom to top. Unlike DPC to directly use the γ method to select *C* final clusters and apply the δ-based allocation strategy to assign the remaining points, our method merges sub-clusters that are both intersecting and similar into *C* final clusters.

Taking similarity values as an input of the single-linkage clustering method, we get a dendrogram of sub-clusters. By merely cutting the dendrogram into *C* clusters (branches) from top to bottom, the clustering is completed.

Fig. 7 illustrates the entire process of FHC-LDP in dealing with the *Jain* dataset [22]. As shown in Fig. 7(a), 14 sub-clusters are identified based on neighborhood information, and by fast searching for the optimal border-links, a quick evaluation of the similarity between intersecting sub-cluster is possible. Followed, by taking 14 sub-clusters and their similarity matrix as the input of the single-linkage clustering, we obtain a dendrogram of 14 sub-clusters as in Fig. 7(b), which clearly demonstrates the similarity (cohesion) relationship between sub-clusters. At last, based on similarity, FHC-LDP merges sub-clusters into two final clusters from bottom to top. As a result, the *Jain* dataset is successfully divided as shown in Fig. 7(c). Moreover, it is noteworthy that FHC-LDP draws the shortest associated transfer path (red line) among cluster centers.

The above example verifies that FHC-LDP's hierarchical method of merging sub-clusters into final clusters has higher fault tolerance than DPC's simple allocation of sub-clusters. This is because, firstly, sub-clusters formed based on neighbor information are reliable; secondly, FHC-LDP considers the similarity between sub-clusters, while DPC does not. Moreover, by using our fast similarity evaluation method, in addition to the KNN search, FHC-LDP does not require any additional distance calculation, which makes it run in about $O(nlog(n))$ that is much faster than DPC ($O(n^2)$).

4.4. The framework

Fig. 8 shows the framework of FHC-LDP. FHC-LDP fast calculates the KNN-density for each point, and then automatically builds reliable sub-clusters by each point finding its neighbor-parent node (meanwhile recording the depth value ω of each point). Next, FHC-LDP converts the depth value of each point to its Φ value

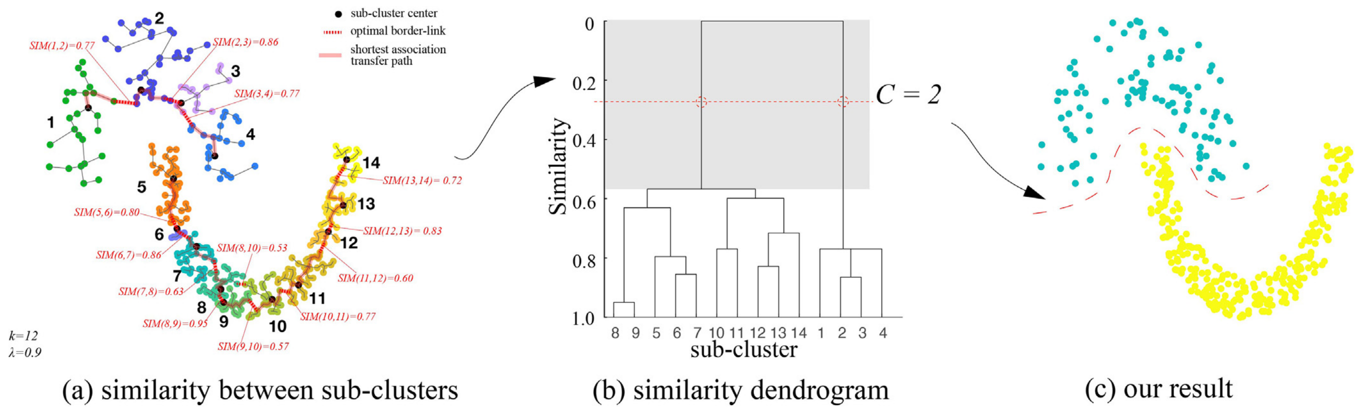


Fig. 7. The clustering process of FHC-LDP ($k = 12, C = 2$) on the *Jain* dataset.

and searches for mutual-border points between each pair of sub-clusters. Followed, FHC-LDP fast evaluates the similarity between sub-clusters based on their mutual-border points with the max sum of Φ. Subsequently, these similarity values are used as an input of the single-linkage clustering to build a dendrogram of sub-clusters. Finally, according to the specified number of clusters C, FHC-LDP cuts the dendrogram from top to bottom to obtain the final clusters. Algorithm 1 shows the overall steps of our FHC-LDP.

```
Algorithm 1: FHC-LDP
Input: Dataset $I \in \mathbb{R}^{n\times d}$ ($n$ is the total number of data ponits,
    and $d$ is the number of dimensions of the dataset), number
    of neighbors $k$, number of clusters $C$
Output: result of clustering $\{Cl_1, Cl_2, \dots, Cl_C\}$($C$ is the number
    of clusters).
    1. Normalization of dataset $I$.
    2. Fast searches for $k$ nearest neighbors and calculates local
    density $\rho^k$ according to Eq. (10).
    3. Searches for each point's neighbor-parent node $NPN$ by
    Eq. (10), at the same time, calculates its depth value $\omega$ by
    Eq. (10).
    4. Fast generates sub-clusters by each point inheriting the
    label of its neighbor-parent node.
    5. Calculates center-association degree $\Phi$ of each point in its
    sub-clusters according to Eq. (10).
    6. Calculates the similarity value between each pair of sub-
    clusters according to the optimal border-link by Eq. (10).
    7. Uses the single-linkage clustering method to build a
    bottom-up dendrogram of sub-clusters based on their
    similarity.
    8. Cuts the dendrogram of sub-clusters into $C$ final clusters
    to complete clustering.
```

4.5. Analysis of complexity

We set the total number of points to n, the number of neighbors to k, and the total number of sub-clusters to n_{sc} that is far less than n.

Time Complexity Analysis: the detailed time complexity analysis of Algorithm 1(FHC-LDP) is as follows:

Line 1: every point needs normalization, that is, approximately $O(n)$.

Line 2: fast searches for k nearest neighbors and calculates local density for each point, needing $O(nlog(n))$.

Line 3: the search of neighbor-parent node and the calculation of depth value can be performed simultaneously, needing $O(nlog(k))$.

Line 4: generates sub-clusters, needing $O(n)$.

Line 5: calculates center-association degree Φ value of each point, needing $O(n)$.

Line 6: calculates similarity value between sub-clusters, needing $O(nk)$ + $O(n_{sc}log(n_{sc}))$.

Line 7: single-linkage clustering of sub-clusters, needing $O(n_{sc}^2)$.

Line 8: divides sub-clusters into C final clusters, needing $O(n)$.

Comprehensively, the overall time complexity of FHC-LDP is about:
$O(n + nlog(n) + nlog(k) + n + n + nk + n_{sc}log(n_{sc}) + n_{sc}^2 + n) = O(nlog(n))$. Also, it is worth mentioning that, except for the KNN search of each point, FHC-LDP does not have any additional distance calculation. This makes the execution speed of FHC-LDP faster than FastDPeak, because, in addition to the KNN search, the latter also needs to calculate the δ of n_{sc} local density peaks.

Space Complexity: 1) FHC-LDP saves k nearest neighbors for every point, needs $O(nk)$ space. 2) FHC-LDP needs a similarity matrix of sub-clusters, needs $O(n_{sc}^2)$ space. Usually, $n_{sc}^2 \ll nk$, thus the overall space complexity of FHC-LDP is about: $O(nk + n_{sc}^2) = O(nk)$.

5. Experiments

5.1. Experimental set up

Algorithms: clustering performance comparisons are launched on the three well-known clustering algorithms (K-means [5], DBSCAN [7], and mean-shift [23]), naive DPC [8], and some state-of-the-art DPC-based algorithms (SNN-DPC [11], KNN-DPC [26], FastDPeak [19], and McDPC [17]), as well as the proposed FHC-LDP. For K-means, we run 100 times on each dataset and pick out the best result for comparison to avoid random factors as much as possible. For DPC, SNN-DPC, KNN-DPC, and FastDPeak, we directly use the γ method to obtain cluster centers according to the specified number C of clusters.

Datasets: we use 9 different types of 2-dimension synthetic datasets to evaluate the clustering performance of FHC-LDP in recognizing various complex shapes, and select 9 real-world datasets to further evaluate the performance of FHC-LDP on high-dimensional and large real-world datasets. The selected real-world datasets include 4 common UCI real-world datasets [27] (the *Iris*, *Wine*, *Movementlibras*, and *Breastcancer* datasets), 4 large datasets (the *YTF* (YouTube Faces) [28] dataset that composed of 10,000 samples of faces, the *REUTERS* [29] dataset that composed of 10,000 samples of English news stories, the *USPS* [30] dataset that composed of 11,000 samples of handwritten digits, the *MNIST* [31] dataset that composed of 10,000 samples of labeled images of handwritten digits, and the classic *OlivettiFace* dataset that consists 400 faces). The datasets are listed in Table 1.

Machine configuration: experiments are conducted by using Matlab (r2017b) on Mac-Book Pro with 2.9 GHz Intel Core i5, 8G RAM.

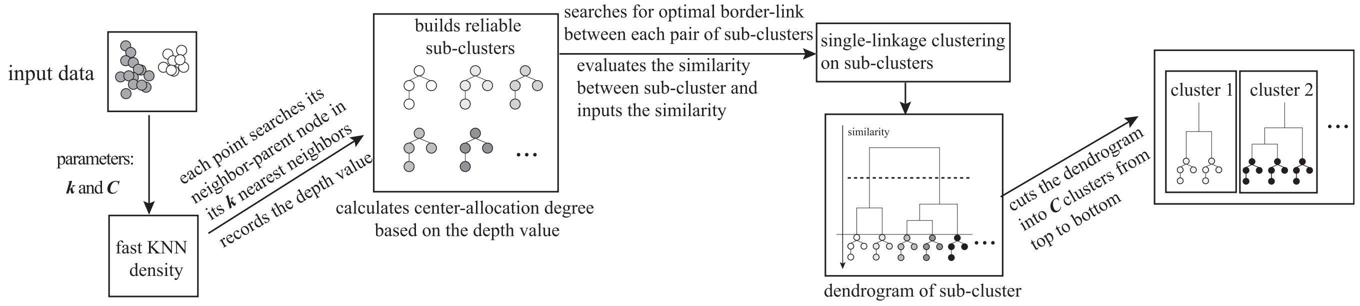

input data
parameters:
k and C
fast KNN density
each point searches its neighbor-parent node in its k nearest neighbors
records the depth value
builds reliable sub-clusters
calculates center-allocation degree based on the depth value
searches for optimal border-link between each pair of sub-clusters
evaluates the similarity between sub-cluster and inputs the similarity
single-linkage clustering on sub-clusters
similarity
dendrogram of sub-cluster
cuts the dendrogram into C clusters from top to bottom
cluster 1
cluster 2

Fig. 8. The framework of FHC-LDP.

Table 1
Datasets.

Synthetic					Synthetic					Real-world					Real-world				
Dataset	Instances	Attributes	Clusters	Source	Dataset	Instances	Attributes	Clusters	Source	Dataset	Instances	Attributes	Clusters	Source	Dataset	Instances	Attributes	Clusters	Source
Agg	788	2	7	[34]	Flame	240	2	3	[35]	Iris	150	4	3	[27]	Wine	178	13	3	[27]
Jain	373	2	2	[22]	Lineblobs	266	2	3	[36]	Movementlibras	360	90	15	[27]	Breastcancer	569	30	2	[27]
Threecircles	299	2	3	[36]	S3	5000	2	15	[32]	OlivettiFaces	400	92×112	40	[37]	REUTERS	10000	10	4	[29]
D31	3100	2	31	[38]	Spiral	312	2	3	[39]	YTF	10000	10	41	[28]	USPS	11000	10	10	[30]
Sitcks	512	2	4	[36]						MNIST	10000	500	10	[31]					

Data preprocessing: we use the min–max normalization method [32] to normalize each dataset to reduce the influence of different metrics in different dimensions.

Evaluation metric: we use the Adjusted Rand Index (ARI) [33] and Adjusted Mutual Information (AMI) [33] to evaluate the clustering performance of clustering algorithms.

Parameter requirements: FHC-LDP (k/C), DPC (p/C), SNN-DPC (k/C), KNN-DPC (k/C), FastDPeak (k/C), DBSCAN ($\varepsilon/MinPts$), mean-shift (*bandwidth*), K-means (C), and McDPC ($\gamma/\theta/\lambda/p$). These needed parameters are displayed as *PAR* in the subsequent experimental results tables.

5.2. Experiments on synthetic datasets

Herein, we demonstrate and discuss the quantitative comparisons on various types of synthetic datasets. In Fig. 9, we only present the clustering results of FHC-LDP, DPC, SNN-DPC (an excellent DPC variant algorithm designed to optimize the allocation

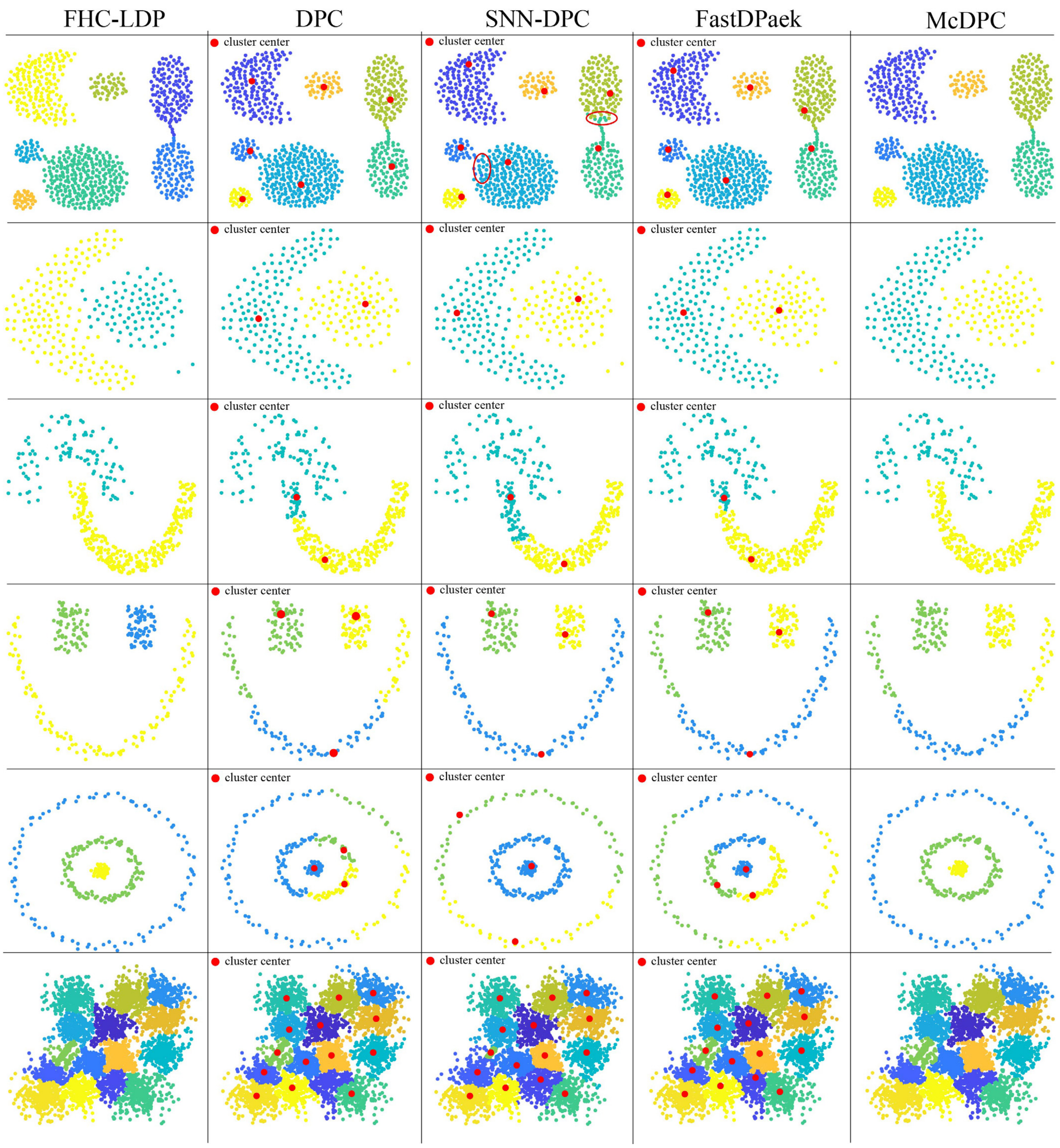


Fig. 9. The clustering results of different algorithms on 6 synthetic datasets.

Table 2
The comparison for 9 clustering algorithms on 9 synthetic datasets.

Algorithm	FHC-LDP			DPC			SNN-DPC			KNN-DPC			FastDPeak			DBSCAN			mean-shift			K-Means			McDPC		
Dataset	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR
Agg	**1.00**	**1.00**	15/7	0.99	0.99	2%/7	0.95	0.96	15/7	0.97	0.98	15/7	0.99	0.99	20/7	0.96	0.98	0.003/10	0.83	0.82	0.21	0.82	0.75	7	0.99	0.99	0.1/0.01/0.21/1
Flame	**1.00**	**1.00**	15/2	**1.00**	**1.00**	4%/2	0.91	0.95	5/2	**1.00**	**1.00**	5/2	**1.00**	**1.00**	21/2	0.87	0.95	0.009/8	0.88	0.93	0.25	0.45	0.48	2	**1.00**	**1.00**	0.1/0.01/0.4/5
Jain	**1.00**	**1.00**	12/2	0.54	0.62	2%/2	0.40	0.40	12/2	0.62	0.71	7/2	0.61	0.70	11/2	0.86	0.97	0.006/1	0.52	0.62	0.33	0.49	0.57	2	**1.00**	**1.00**	0.3/0.3/0.23/5
Lineblobs	**1.00**	**1.00**	15/3	0.58	0.49	2%/3	**1.00**	**1.00**	15/3	0.69	0.65	3/3	0.78	0.72	15/3	**1.00**	**1.00**	0.003/5	0.43	0.35	0.35	0.59	0.50	3	0.58	0.49	0.3/0.5/0.3/2
Threecircles	**1.00**	**1.00**	15/3	0.10	−0.0	2%/3	0.60	0.51	5/3	0.14	0.08	3/3	0.20	0.11	10/3	**1.00**	**1.00**	0.007/2	0.45	0.41	0.20	0.15	0.05	3	**1.00**	**1.00**	0.3/0.5/0.08/2
S3	0.94	0.93	47/15	0.94	0.92	2%/15	0.88	0.83	35/15	**0.96**	**0.95**	50/15	0.95	0.94	38/15	0.66	0.30	0.001/50	0.87	0.83	0.08	0.90	0.87	15	0.94	0.93	0.05/0.05/0.08/2
D31	**0.96**	**0.94**	40/31	0.95	0.93	2%/31	**0.96**	**0.94**	50/31	**0.96**	**0.94**	62/31	0.95	0.93	50/31	0.66	0.21	0.0005/20	0.95	0.94	0.07	0.91	0.82	31	0.94	0.90	0.05/0.01/0.06/2
Spiral	**1.00**	**1.00**	9/3	**1.00**	**1.00**	2%/3	**1.00**	**1.00**	6/3	**1.00**	**1.00**	31/3	**1.00**	**1.00**	7/3	**1.00**	**1.00**	0.01/5	0.28	0.14	0.1	−0.01	−0.01	3	**1.00**	**1.00**	0.3/0.3/0.1/5
Sticks	**1.00**	**1.00**	15/4	0.63	0.54	2%/4	**1.00**	**1.00**	11/4	0.82	0.77	10/4	0.57	0.40	15/4	**1.00**	**1.00**	0.003/1	0.79	0.79	0.25	0.81	0.70	4	**1.00**	**1.00**	0.3/0.3/0.2/2

Table 3
The comparison of 9 clustering algorithms on 9 real-world datasets.

Algorithm	FHC-LDP			DPC			SNN-DPC			KNN-DPC			FastDPeak			DBSCAN			mean-shift			K-Means			McDPC		
Dataset	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR	AMI	ARI	PAR
Iris	0.88	0.90	12/3	0.76	0.72	3%/3	**0.91**	**0.92**	15/3	0.86	0.88	3/3	0.88	0.90	10/3	0.58	0.57	0.10/6	0.71	0.64	0.24	0.73	0.71	3	0.77	0.72	0.1/0.1/0.34/1
Wine	0.74	0.73	20/3	0.71	0.67	2%/3	**0.83**	**0.90**	18/3	0.74	0.73	18/3	0.73	0.71	20/3	0.52	0.54	0.2/4	0.60	0.74	0.55	0.83	0.84	3	0.50	0.44	0.2/0.3/0.6/1
Movementlibras	0.56	0.34	8/15	0.48	0.26	2%/15	**0.58**	**0.39**	11/15	0.53	0.31	18/15	0.54	0.33	6/15	0.26	0.09	1.6/6	0.50	0.32	1.20	0.54	0.32	15	0.30	0.10	0.1/0.15/0.8/1
Breastcancer	0.64	0.75	11/2	0.01	- 0.0	2%/2	**0.75**	**0.85**	12/2	0.45	0.51	17/2	0.46	0.53	11/2	0.37	0.50	0.24/51	0.20	0.38	0.50	0.61	0.73	2	0.31	0.43	0.5/0.20/0.4/1
REUTERS	0.36	0.37	165/4	0.24	0.26	1%/4	0.39	0.36	50/4	0.30	0.33	100/4	0.24	0.25	200/4	0.12	0.09	0.06/40	0.05	0.01	0.39	**0.51**	**0.57**	4	0.02	0.01	0.1/1.5/0.38/2
YTF	**0.79**	**0.57**	65/41	0.73	0.50	2%/41	0.69	0.40	50/41	0.76	0.51	100/41	0.78	**0.57**	150/41	0.63	0.20	0.09/10	0.67	0.37	0.414	0.74	0.51	41	0.74	0.48	0.1/1.5/0.35/2
USPS	**0.68**	**0.57**	70/10	0.34	0.24	2%/10	0.56	0.38	50/10	0.51	0.30	110/10	0.65	0.51	80/10	0.45	0.15	0.04/20	0.59	0.39	0.35	0.59	0.48	10	0.29	0.11	0.1/1.5/0.32/2
MNIST	**0.93**	**0.93**	20/10	0.43	0.29	1%/10	0.77	0.60	50/10	0.79	0.69	200/10	0.84	0.76	55/10	0.56	0.23	4/6	0.59	0.39	0.30	0.81	0.77	10	0.36	0.25	2/10/2.35/2
OlivettiFaces	**0.86**	**0.77**	5/40	0.76	0.62	0.4%/40	0.81	0.68	6/40	0.83	0.71	16/40	0.82	0.70	6/40	0.73	0.59	0.5/2	0.63	0.25	0.80	0.74	0.59	40	0.61	0.42	0.1/0.25/0.75/1

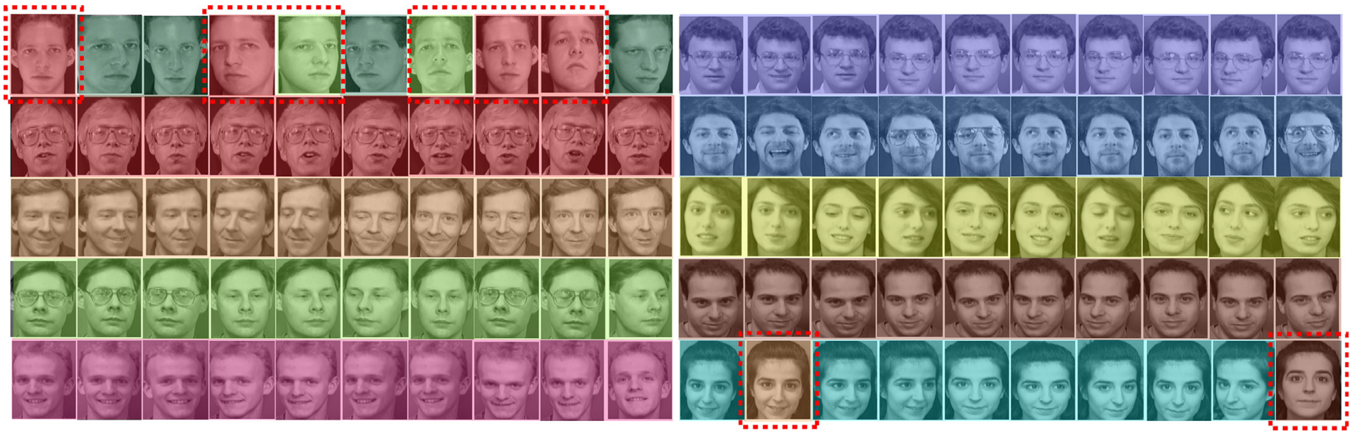


Fig. 10. The clustering result of FHC-LDP on the first 100 faces of the *OlivettiFaces* dataset.

strategy), FastDPeak (an outstanding DPC speed-up algorithm), and McDPC (a state-of-the-art multi-center density peak clustering) on six different synthetic datasets. In addition, Table 2 displays the performance score comparison of all algorithms.

In Fig. 9, FHC-LDP almost perfectly recognizes all the synthetic datasets, by ensuring each cluster consists of the most similar reliable sub-clusters, which guarantees that points in each cluster are related to each other. McDPC successfully identifies most of the synthetic datasets, except for the half-arc cluster in the *Lineblobs* dataset, because the three approximate-density clusters of *Lineblobs* are difficult to be divided into different density levels, which makes McDPC fail to reconstruct the half-arc cluster. Although *Jain* and *Threecircles* also consist of complex-shaped clusters, these clusters can be easily divided into different density levels, thus McDPC can reconstruct them successfully. SNN-DPC, DPC, and FastDPeak fail to divide the clusters of *Jain* and *Threecircles* datasets because their γ method incorrectly locates cluster centers. DPC and FastDPeak fail to completely reconstruct the half-arc cluster of *Linblobs* since their allocation strategies assign non-adjacent points into a cluster. In addition, for the *Agg*, *Flame*, and *S3* datasets, except for some small flaws (marked by red circles) of SNN-DPC on *Agg*, the clustering results of all algorithms are almost perfect.

The above experimental analysis from Fig. 9 and the comparison of AMI and ARI scores in Table 2 verify that FHC-LDP is outstanding in identifying different types of clusters.

5.3. Experiments on real-world datasets

5.3.1. Evaluation on small-scale UCI real-world datasets

Table 3 shows the AMI and ARI scores of all algorithms on 9 real-world datasets. As shown, for 4 small-scale UCI real-world datasets (*Iris*, *Wine*, *Movementlibras*, and *Breastcancer*), the overall clustering performance of FHC-LDP is outstanding and is only second to SNN-DPC, but the latter has an extremely high complexity of $O((k+C)n^2)$. This verifies the effectiveness of FHC-LDP in multi-dimensional and small-scale real-world datasets.

5.3.2. Evaluation on large-scale real-world datasets

Among four tested large-scale datasets, although the dimensionality of the *YTF*, *REUTERS*, and *USPS* datasets are not too high to be challenging, the large data-size of these datasets adds extra difficulties to clustering. As shown in Table 3, among naive DPC and the variants of DPC, FHC-LDP owns the highest AMI and ARI scores on *YTF*, *REUTERS*, and *USPS* datasets.

Unlike the above three low-dimensional datasets, the *MNIST* dataset, a 500-dimensional dataset of 10,000 points, is more challenging. Usually, dimensionality reduction techniques are used when encountering datasets with overly-high dimensional space that are uneasy to cluster. However, to further verify the feasibility of FHC-LDP when coming across overly-high dimensional large-scale datasets, we conduct comparative experiments on the *MNIST* dataset without dimensionality reduction. As shown in Table 3, among algorithms that can identify the number of clusters, our method obtains extremely high scores on *MNIST* ($AMI = 0.93, ARI = 0.93$), while others obtain scores no higher than 0.85.

The intuitive high AMI and ARI scores that FHC-LDP acquire in all the abovementioned comparative experiments on high-dimensional datasets and large-scale real-world datasets verify the feasibility and wide applicability of FHC-LDP in various real-world datasets regardless of their space dimension or data-size.

5.3.3. Evaluation on the OlivettiFace dataset

OlivettiFace dataset [37], a widespread benchmark for machine learning algorithms, is an image dataset that contains 10 different face-angle images of 40 people. It is challenging in analyzing this dataset since its number of clusters is comparable with its total number of elements (that is, each cluster has only ten elements). As shown in Table 3, by evaluating the scores of AMI and ARI, the conclusion can be made that our algorithm has the highest recognition accuracy.

Fig. 10 demonstrates the performance of FHC-LDP in recognizing the first 100 face images in *OlivettiFace*, where different colors represent different clusters. It can be noted the recognition accuracy of the 100 faces is up to 92% (8 faces marked by red dotted rectangles are not successfully recognized), which is competitive to other state-of-the-art approaches.

5.4. The speed of FHC-LDP

5.4.1. Speed comparison of all DPC-based algorithms

For clustering large datasets, running speed is an important performance indicator that cannot be ignored. To show the fast speed of FHC-LDP, we conduct several comparison experiments with other DPC-based algorithms, as in Table 4 and Fig. 11.

It can be noted that FHC-LDP($O(nlog(n))$) and FastDPeak ($O(nlog(n))$) [19] based on the KNN distance of data points are significantly faster than the other four algorithms that based on the distance between data points. For the other four algorithms, DPC ($O(n^2)$) and KNN-DPC ($O(n^2)$) are generally about the same speed, because KNN-DPC does not focus on speeding up DPC; McDPC ($O(n^2)$) is slower than DPC on small datasets, but faster on large datasets; SNN-DPC ($O((k+C)n^2)$) is the most time-consuming

Table 4
The runtime of 6 DPC-based clustering algorithms on all tested datasets (unit: second).

Dataset	Iris	Wine	Flame	Lineblobs	Threecircles	Spiral	Movementlibras	Jain	OlivettiFaces	Sticks	Breastcancer	Agg	D31	S3	MNIST	YTF	REUTERS	USPS
n	150	178	240	266	299	312	360	373	400	512	569	788	3100	5000	10000	10000	10000	11000
FHC-LDP	**0.0029**	**0.0031**	**0.0045**	**0.0033**	**0.0062**	**0.0039**	**0.0053**	**0.0036**	**0.0085**	**0.0075**	**0.0094**	**0.0091**	**0.0249**	**0.0580**	**0.7332**	**0.5862**	**1.5039**	**1.0599**
FastDPeak	0.0036	0.0055	0.0051	0.0038	0.0089	0.0126	0.0117	0.0098	0.0099	0.0085	0.0136	0.0230	0.0394	0.0923	0.8789	0.9569	1.5674	1.1154
McDPC	0.3105	0.2554	0.2614	0.2575	0.3132	0.5253	0.2669	0.2775	0.4256	0.2631	0.3015	0.3390	1.2759	2.4270	22.0311	8.3823	9.1931	10.3127
KNN-DPC	0.0059	0.0138	0.0197	0.0153	0.0093	0.0447	0.0478	0.0457	0.0390	0.1099	0.0620	0.1923	1.1967	4.4080	33.8067	17.8734	20.9620	24.8013
DPC	0.0064	0.0051	0.0138	0.0161	0.0281	0.0905	0.0221	0.0682	0.0248	0.0223	0.0846	0.1434	1.4412	4.5626	36.0860	22.3275	23.8793	32.6120
SNN-DPC	0.1043	0.1066	0.0987	0.1145	0.1155	0.2028	0.2848	0.3116	0.2329	0.2753	0.5397	1.0112	11.7149	28.3429	129.6524	139.8833	120.2540	157.1375

The best values are highlighted.

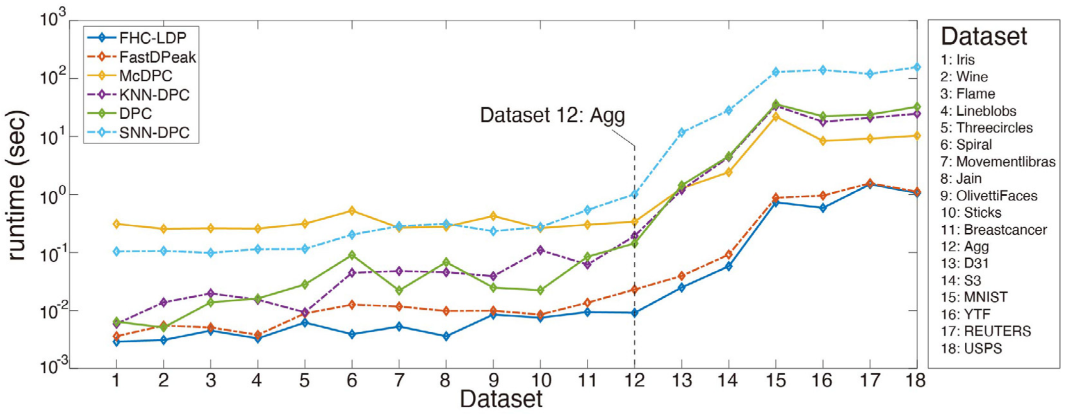


Fig. 11. The speed comparison of different algorithms on all tested datasets.

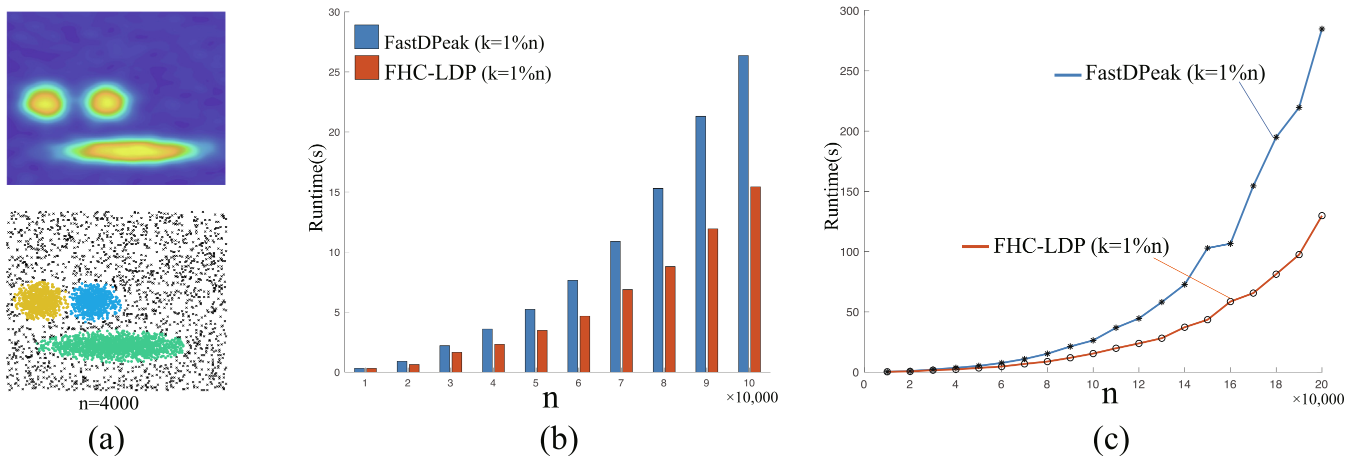


Fig. 12. Experiments of FHC-LDP and FastDPeak on the tested dataset (a) with n increasing (b,c).

due to its complex allocation strategy. Moreover, it is worth noting that as the dataset size becomes larger, the speed advantage of FHC-LDP and FastDPeak becomes more obvious.

5.4.2. Speed comparison between FHC-LDP and FastDPeak

As shown in Table 4 and Fig. 11, FHC-LDP and FastDPeak run at a similar speed in datasets of no more than 10,000 data, since the time complexities of FHC-LDP and FastDPeak are both mainly on the calculation of KNN distance of data points. Although the overall computational complexities of FHC-LDP and FastDPeak are both $O(nlogn)$, there is a small difference in the total distance calculation. FHC-LDP only needs to calculate the KNN distance of points, while FastDPeak needs to calculate the δ values of local density peaks [19], which is non-negligible as datasets become larger. Therefore, to further compare the running speed of FHC-LDP and FastDPeak, we conduct a comparison experiment on different-size tested datasets drawn from a synthetic probability distribution [14], as shown in Fig. 12.

In Fig. 12, under the same k value setting (the time-consuming of KNN search is roughly the same), FHC-LDP is faster than FastDPeak because of the latter's extra calculation of the δ values for a large number of local density peaks, which verifies that FHC-LDP is faster than FastDPeak.

In summary, FHC-LDP with fast running speed is suitable for large datasets.

5.5. The dendrogram of FHC-LDP

Similar to DPC's decision graph that can assist cluster center selection without prior knowledge, FHC-LDP's dendrogram can also assist the determination of cluster number for its superiority in demonstrating the dataset structure. To evaluate the performance of FHC-LDP's dendrogram, we compare FHC-LDP's dendrogram with the decision graphs of other DPC-based algorithms.

Fig. 13 presents FHC-LDP's dendrogram and the decision graphs of DPC, SNN-DPC, and FastDPeak on the *Agg*, *Flame*, *Jain*, *Lineblobs*, *Threecircles*, and *S3* datasets. The clustering results in Fig. 9 tell that decision graphs help to accurately locate cluster centers of the *Agg*, *Flame*, *Lineblobs*, and *S3* datasets, while misleading the choice of cluster centers of the *Threecircles* and *Jain* datasets. Because the first C number of points with largest γ can not necessarily represent the real cluster centers (as analyzed in Section 3.2.2).

Unlike other decision graphs that only provide unreliable γ-values, FHC-LDP's dendrogram of sub-clusters can clearly demonstrate the dataset structure. As in Fig. 13, FHC-LDP's dendrogram on the *Agg* dataset clearly displays that *Agg*'s 7 clusters are located in 5 non-intersecting regions, among which two non-intersecting regions are separately composed of two clusters. To divide the *Agg* dataset into 7 clusters, we just need to cut the dendrogram into 7 branches horizontally (marked by a red dotted line).

Besides, benefited from such superiority of FHC-LDP's dendrogram, FHC-LDP can accurately divide datasets into non-intersecting clusters by directly cutting out branches with no

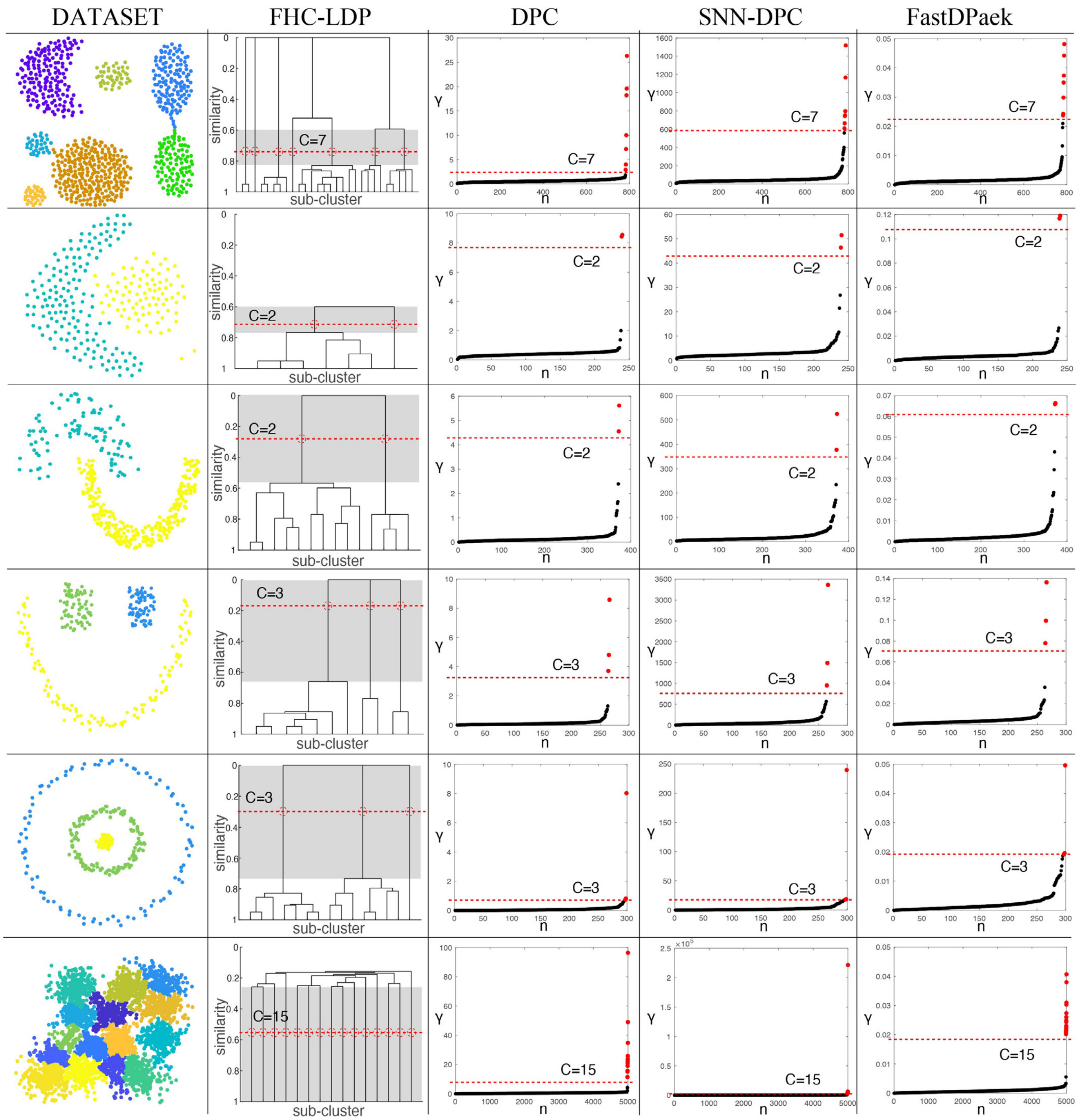


Fig. 13. The comparison between FHC-LDP's dendrogram and the decision graphs of DPC, SNN-DPC, and FastDPeak.

similarity in the dendrogram. For example, for the *Jain*, *Lineblobs*, and *Threecircles* datasets, no matter how complex the cluster shapes and density differences are, as long as clusters are not intersecting, FHC-LDP's dendrogram can accurately determine the number of clusters by finding the number of branches with no similarity. In addition, since FHC-LDP's dendrogram is based on the similarity between sub-clusters, it also presents the overlapping degree between clusters. For example, for the *Flame* and *S3* datasets, FHC-LDP's dendrograms clearly show the consistency of a single non-intersecting region that with multiple overlapping clusters: *Flame* is composed of 2 high-overlap clusters and *S3* is composed of 15 low-overlap clusters. In contrast, the decision graphs provide no other information except γ-values.

The above performance comparison experiments on FHC-LDP's dendrogram and the decision graphs demonstrate that FHC-LDP's dendrogram not only provides some important structure information of datasets, but also helps to determine the number of clusters.

5.6. The setting of parameter k

According to the size difference, we divide datasets into: general datasets (i.e., datasets with $500 \leqslant n < 10,000$.), small datasets (i.e., datasets with $n < 500$.), and large datasets (i.e., datasets with $n \geqslant 10,000$). Correspondingly, we have three different experience-based ways to set k:

(i) for general datasets, we generally set $1\%n \leqslant k \leqslant 3\%n$;

(ii) for small datasets, since n is too small and the KNN-density evaluation method requires a sufficiently large value of k, we generally set $5 \leqslant k \leqslant 20$;

(iii) for large datasets, since n is large enough, we generally set $20 \leqslant k \leqslant 2\%n$.

To demonstrate the robustness of parameter k setting, we draw the k-AMI plots of some tested dataset and present in Fig. 14. It can be noted that as long as the k value reaches a certain value, our clustering performance will tend to be stable. The setting range of parameter k (marked by green area) basically covers the optimal parameter range (i.e., the parameter range that can obtain *AMI* scores that differs from the highest *AMI* score by less than 0.1, marked by light blue area). The above verifies the insensitivity of FHC-LDP to its parameter and the effectiveness of the setting range of parameter k.

6. Conclusion

Herein, a fast hierarchical clustering of local density peaks via an association degree transfer method named FHC-LDP is proposed. It only needs to find the k nearest neighbors of each point for KNN-density evaluation, and then let each point finds its neighbor-parent node in its k nearest neighbors to quickly form sub-clusters without calculating the distance δ. Besides, we also design a fast association degree transfer method for the similarity evaluation between sub-clusters, and a hierarchical clustering method for the mergence of sub-clusters into final clusters, which effectively overcome the defects of DPC's allocation strategy and increase the accuracy of cluster center recognition. Numerous experiments have verified that FHC-LDP is fast and can recognize arbitrary-shaped clusters regardless of their spatial dimensions and data-size.

In future work, we will focus on exploring other alternative methods for fast evaluating density values. In addition, we hope to develop a fully automatic and robust method for selecting the optimal grouping scheme.

CRediT authorship contribution statement

Junyi Guan: Conceptualization, Methodology, Software. **Sheng Li:** Conceptualization, Validation. **Xiongxiong He:** Supervision. **Jinhui Zhu:** Data curation. **Jiajia Chen:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Science Foundation of P.R. China (Grant: 61873239) and Zhejiang Science Foundation (Grant:2020C03074).

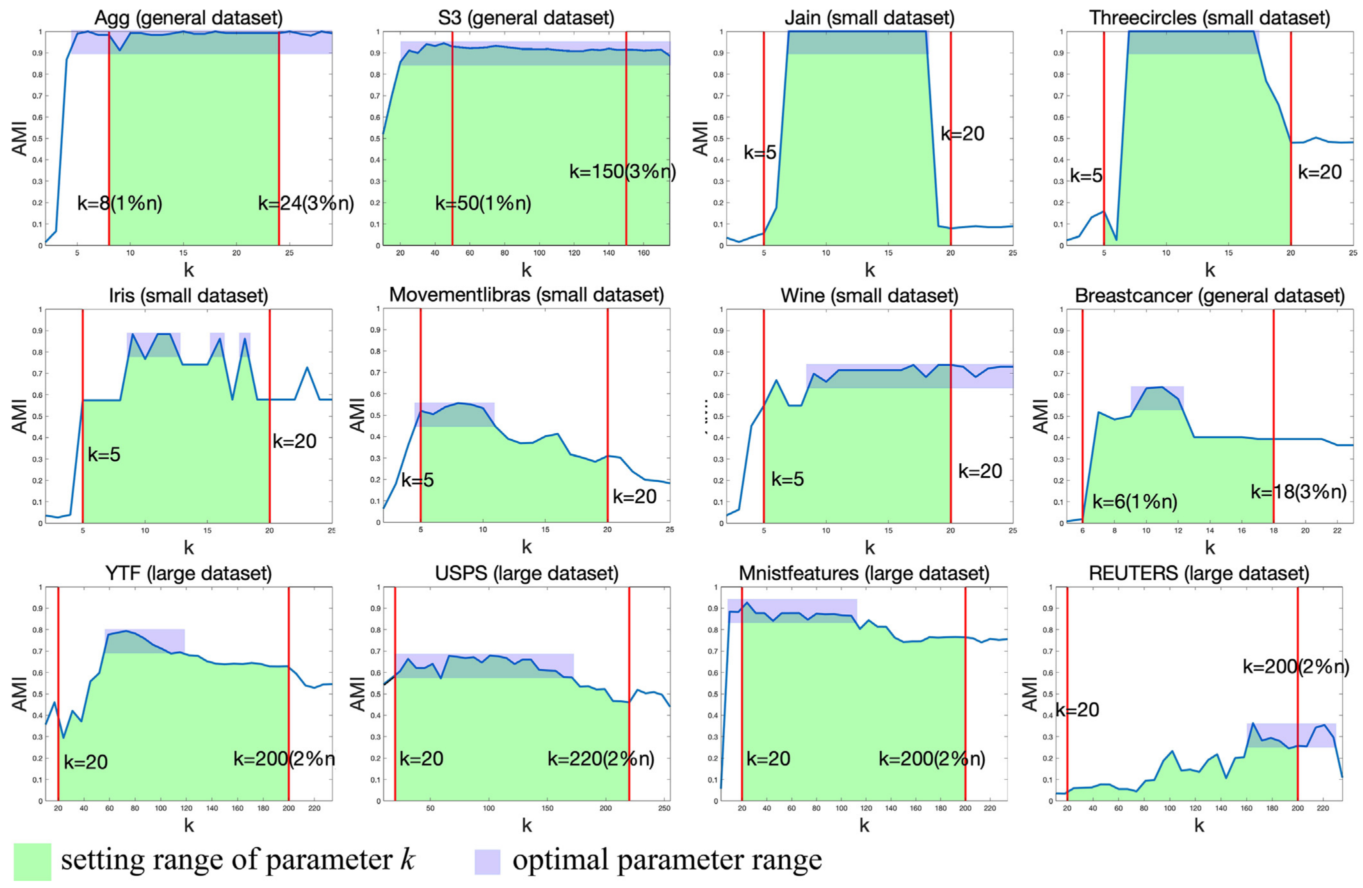


Fig. 14. The k-AMI plots of FHC-LDP on several tested datasets.

References

[1] P. Berkhin, A survey of clustering data mining techniques, Grouping Multidimensional Data. Springer, Berlin, Heidelberg (2006) 25–71 doi:10.1007/3-540-28349-8_2 .

[2] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys (CSUR) 31 (3) (1999) 264–323, https://doi.org/10.1145/331499.331504.

[3] M.I. Jordan, T.M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260, https://doi.org/10.1126/science.aaa8415.

[4] C. Wiwie, J. Baumbach, R. Röttger, Comparing the performance of biomedical clustering methods, Nature Methods 12 (11) (2015) 1033–1038, https://doi.org/10.1038/nmeth.3583.

[5] J. MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability 1 (14) (1967) 281–297.

[6] S.C. Johnson, Hierarchical clustering schemes, Psychometrika 32 (3) (1967) 241–254, https://doi.org/10.1007/BF02289588.

[7] M. Ester et al., A density-based algorithm for discovering clusters in large spatial databases with noise, Kdd 96 (34) (1996) 226–231.

[8] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496, https://doi.org/10.1126/science.1242072.

[9] Y. Shi et al., A novel clustering-based image segmentation via density peaks algorithm with mid-level feature, Neural Computing and Applications 28 (1) (2017) 29–39, https://doi.org/10.1007/s00521-016-2300-1.

[10] B. Wang et al., Density peaks clustering based integrate framework for multi-document summarization, CAAI Transactions on Intelligence Technology 2 (1) (2017) 26–30, https://doi.org/10.1016/j.trit.2016.12.005.

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

[12] X. Juanying et al., Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors, Information Sciences 354 (2016) 19–40, https://doi.org/10.1016/j.ins.2016.03.011.

[13] D. Mingjing et al., Density peaks clustering using geodesic distances, International Journal of Machine Learning and Cybernetics 9 (8) (2018) 1335–1349, https://doi.org/10.1007/s13042-017-0648-x.

[14] D.U. Pizzagalli, S.F. Gonzalez, R. Krause, A trainable clustering algorithm based on shortest paths from density peaks, Science, Advances 5 (10) (2019) eaax3770, https://doi.org/10.1126/sciadv.aax3770.

[15] Y. Liu, M. Zhengming, Y. Fang, Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy, Knowledge-Based Systems 133 (2017) 208–220, https://doi.org/10.1016/j.knosys.2017.07.010.

[16] G. Wang, Y. Wei, P. Tse, Clustering by defining and merging candidates of cluster centers via independence and affinity, Neurocomputing 315 (2018) 486–495, https://doi.org/10.1016/j.neucom.2018.07.043.

[17] Y. Wang et al., Mcdpc: multi-center density peak clustering, Neural Computing and Applications (2020) 1–14, https://doi.org/10.1007/s00521-020-04754-5.

[18] Z. Rong et al., A novel hierarchical clustering algorithm based on density peaks for complex datasets, Complexity (2018), https://doi.org/10.1155/2018/2032461.

[19] C. Yewang et al., Fast density peak clustering for large scale data based on knn, Knowledge-Based Systems 187 (2020) , https://doi.org/10.1016/j.knosys.2019.06.032 104824.

[20] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 97–104, https://doi.org/10.1145/1143844.1143857.

[21] J.C. Gower, G.J.S. Ross, Minimum spanning trees and single linkage cluster analysis, Journal of the Royal Statistical Society: Series C (Applied Statistics) 18 (1) (1969) 54–64, https://doi.org/10.2307/2346439.

[22] A.K. Jain, M.H.C. Law, Data clustering: A user's dilemma, in: International conference on pattern recognition and machine intelligence, Springer, Berlin, Heidelberg, 2005, pp. 1–10.

[23] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (8) (1995) 790–799, https://doi.org/10.1109/34.400568.

[24] S.-J. Huang, R. Jin, Z.-H. Zhou, Active learning by querying informative and representative examples, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (10) (2014) 1936–1949, https://doi.org/10.1109/TPAMI.2014.2307881.

[25] B.M.R., et al, Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection, Statistics & Probability Letters 35 (1) (1997) 33–42. doi:10.1016/S0167-7152(96)00213-1 .

[26] M. Du, S. Ding, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, Knowledge-Based Systems 99 (1) (2016) 135–145, https://doi.org/10.1016/j.knosys.2016.02.001.

[27] K. Bache, M. Lichman, Uci machine learning repository, http://archive.ics.uci.edu/ml. .

[28] L. Wolf, T. Hassner, I. Maoz, Face recognition in unconstrained videos with matched background similarity, Computer Vision and Pattern Recognition (CVPR). .

[29] D.D. Lewis, Y. Yang, T. Rose, F. Li, Rcv1: A new benchmark collection for text categorization research, Journal of Machine Learning Research 5 (2004) 361–397, https://doi.org/10.1023/B:JODS.0000024125.05337.9e.

[30] D. Keysers, T. Deselaers, C. Gollan, H. Ney, Deformation models for image recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (8) (2007) 1422–1435, https://doi.org/10.1109/TPAMI.2007.1153.

[31] Y. LeCun, C. Cortes, Mnist handwritten digit database, http://yann.lecun.com/exdb/mnist/. .

[32] P. Franti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recognition 39 (5) (2006) 761–775, https://doi.org/10.1016/j.patcog.2005.09.012.

[33] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, Journal of Machine Learning Research 11 (2010) 2837–2854.

[34] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM ttransactions on Knowledge Discovery from Data (tkdd), 1, 2007, p. 4, 10.1145/1217299.1217303.

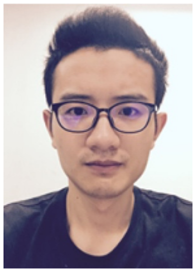
[35] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transactions on Computers 100 (1) (1971) 68–86, https://doi.org/10.1109/T-C.1971.223083.

[36] L. Zelnik-manor, P. Perona, Self-tuning spectral clustering, Neural Information processing Systems (2004) 1601–1608. .

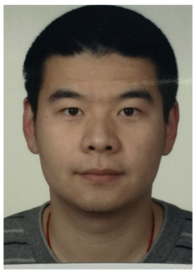
[37] F. Samaria, A. Harter, Parameterisation of a stochastic model for human face identification, Proceedings of 1994 IEEE Workshop on Applications of Computer Vision, IEEE (1994) 138–142 doi:10.1109/ACV.1994.341300. .

[38] C. Veenman, M. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (9) (2002) 1273–1280, https://doi.org/10.1109/TPAMI.2002.1033218.

[39] H. Chang, Dit-YanYeung, Robust path-based spectral clustering, Pattern Recognition 41 (1) (2008) 191–203, https://doi.org/10.1016/j.patcog.2007.04.010.



Junyi Guan currently pursuing the Ph.D. degree in the College of Information Engineering, Zhejiang University of Technology(ZJUT), Hangzhou, China. His current research interests include data mining, pattern recognition, and machine learning.



Sheng Li Ph.D. in electronic engineering, University of York, York, U.K. Associate professor of ZJUT. His research interests include signal processing, machine learning, and pattern recognition.

Xiongxiong He received Ph.D. in Zhejiang University, Hangzhou, China. Professor of ZJUT. His research areas include nonlinear control, signal processing, and pattern recognition.

Jinhui Zhu Ph.D. in surgery, Zhejiang Chinese Medical University, Hangzhou, China. Chief Physician of Second Affiliated Hospital, Zhejiang University School of Medicine. His research interests include bioinformatics and pattern recognition.

Jiajia Chen received M.A. in East China Normal University, Shanghai, China. Her current research interests include data mining and pattern recognition.