Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.052114

ARTICLE

A Shared Natural Neighbors Based-Hierarchical Clustering Algorithm for Discovering Arbitrary-Shaped Clusters

Zhongshang Chen, Ji Feng*, Fapeng Cai and Degang Yang

College of Computer and Information Science, Chongqing Normal University, Chongqing, 400030, China

*Corresponding Author: Ji Feng. Email: jifeng@cqnu.edu.cn

Received: 23 March 2024 Accepted: 14 June 2024 Published: 15 August 2024

ABSTRACT

In clustering algorithms, the selection of neighbors significantly affects the quality of the final clustering results. While various neighbor relationships exist, such as *K*-nearest neighbors, natural neighbors, and shared neighbors, most neighbor relationships can only handle single structural relationships, and the identification accuracy is low for datasets with multiple structures. In life, people's first instinct for complex things is to divide them into multiple parts to complete. Partitioning the dataset into more sub-graphs is a good idea approach to identifying complex structures. Taking inspiration from this, we propose a novel neighbor method: Shared Natural Neighbors (SNaN). To demonstrate the superiority of this neighbor method, we propose a shared natural neighbors-based hierarchical clustering algorithm for discovering arbitrary-shaped clusters (HC-SNaN). Our algorithm excels in identifying both spherical clusters and manifold clusters. Tested on synthetic datasets and real-world datasets, HC-SNaN demonstrates significant advantages over existing clustering algorithms, particularly when dealing with datasets containing arbitrary shapes.

KEYWORDS

Cluster analysis; shared natural neighbor; hierarchical clustering

1 Introduction

Clustering is one of the most important methods of data mining [1]. Intuitively, it groups a set of data in a way that maximizes the similarity within a cluster and minimizes the similarity between different clusters [2]. In recent years, cluster analysis has been widely used in many aspects, such as image segmentation [3], business decision-making [4], data integration and other fields. Clustering methods are typically categorized based on their distinct characteristics, including partition-based clustering [5], density-based clustering [6], and hierarchical clustering [7], among others.

Partitioned clustering algorithms are a kind of high-efficiency clustering algorithms. It uses an iteratively updating strategy to optimize the data until reaches the optimal result. Among the representative ones is the *K*-means algorithm [8]. However, *K*-means is highly sensitive in the initial choice of cluster centers, and it has limitations in identifying non-spherical clusters. Therefore, researchers have proposed numerous improvement algorithms. For example, Tzortzis et al. proposed the MinMax *K*-means clustering algorithm [9], which uses the variance weights to weigh initial points, and enhances

the quality of initial centers. To improve the performance of K-means in detecting arbitrary shape datasets, Cheng et al. proposed the K-means clustering with Natural Density Peaks algorithm (NDP-Kmeans) [10]. It uses the natural neighbor [11] to compute the density points ρ, and chooses the ρ of largest known as the natural representative. Iterative updates of these natural representatives find multiple natural density peaks (NDPs) as initial sub-clusters. In these NDPs, introduced the Graph Distance to iteratively update these NDPs, until favorable results. It improves two methods that center point selection and allocation strategy of K-means, and has superior performance in discovering arbitrary-shaped clusters.

To effectively discover arbitrary-shaped clusters, density-based clustering algorithms are proposed [12]. DBSCAN [13] is one of the typical representative algorithms among them. It identifies dense regions as clusters and filters objects in the sparse region. It defines the density of points by setting the radius ϵ and the minimum number of points *minpts*. Dcore [14] assumes that each cluster is composed of some loosely connected cores that roughly retain the shape of clusters. Daszykowski et al. [15] used *minpts* for estimating ϵ, and thus it sets only one parameter for DBSCAN. RECOME [16] defines the density as the relative K-nearest neighbor kernel density (RNKD), and then obtains the final result by merging the highest RNKD clusters. Rodriguez et al. [17] proposed a clustering algorithm based on based on density and distance (DPC). It selects points with larger densities of ρ and δ distances as clustering centers, known as density peaks. And then determines the clustering labels for each remaining points by the nearest point with higher densities. However, the DPC algorithm is unable to effectively identify the variable-density dataset, and the allocation strategy is sensitive and the fault tolerance is poor. Cheng et al. [18] introduced the concept of a Granular Ball [19–21] in unsupervised learning, improving the allocation strategy of DPC. Liu et al. proposed the SNNDPC algorithm [22], used the shared neighbors to redefine the ρ and δ, and used the two-step allocation method to enhance its performance in discovering arbitrary-shaped clusters.

Hierarchical clustering seeks to build a hierarchy of clusters [23]. Among them, Chameleon [24] is a representative algorithm in hierarchical clustering algorithms. It uses the K-nearest neighbor graph to express the distribution of datasets. Then, it partitions the nearest neighbor graph according to the hMetis algorithm [25]. Finally, based on the connectivity and compactness of sub-clusters, integrates clusters in a bottom-to-top manner. Although the Chameleon algorithm has good performance, the use environment of the hMetis algorithm is not easy to build, and it does not achieve ideal performance when processing arbitrary-shaped datasets. To improve the Chameleon algorithm's performance in processing datasets with arbitrary structures, Zhang et al. [26] used the new natural neighbor graph to replace the K-nearest neighbor graphs. Liang et al. [27] employed the DPC algorithm framework to replace both the K-nearest neighbor graphs and the hMetis algorithm. Zhang et al. [28] used the mutual K-nearest neighbors to directly generate sub-clusters, which omits the process of partitioning the graph.

Datasets with complex structures generally refer to those that contain clusters with diverse shapes (including spherical, non-spherical, and manifold), sizes, and densities. Traditional clustering algorithms (e.g., DBSCAN, K-means) are not particularly effective when dealing with such datasets [29]. Natural neighbor is a parameter-free algorithm that adaptively constructs neighbor relationships. This neighbor relationship is very good for the recognition of manifold clusters, but it is not sensitive to datasets with complex structures. The shared neighbor method [30] takes into consideration the association between sample points rather than solely relying on simple distance metrics. This similarity metric better reflects the relationships between data points, particularly when handling non-spherical or variable density data. However, it is relatively sensitive to the choice of the K parameter.

To show that the proposed neighbor method can effectively identify datasets of any structure, we propose the HC-SNaN algorithm. Inspired by natural neighbors and shared neighbors, we propose a new neighbor method-shared natural neighbor method. This method skillfully utilizes the advantages of natural neighbors and shared neighbors, and more effectively reflects the true distribution of data points in complex structures. The algorithm first constructs a shared natural neighbor graph by using shared natural neighbors and then cuts sub-clusters into multiple cut sub-graphs. Next, we merge the fuzzy points with the cut sub-graphs to form initial sub-clusters. Among them, fuzzy points are points not connected in the sub-graph while cutting the sub-graphs. Finally, based on our proposed new merging strategy, the initial clusters are merged. To show the process of our algorithm more clearly, the flow chart of our proposed HC-SNaN algorithm is shown in Fig. 1.

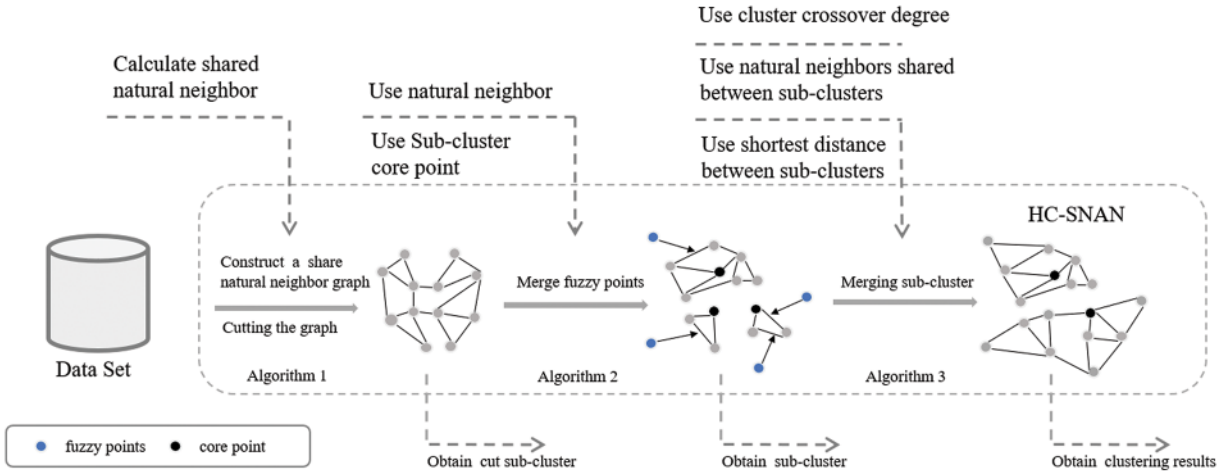

Figure 1: Flow chart of the proposed hierarchical clustering method HC-SNaN

The experimental results on synthetic datasets and real-world datasets show that our algorithm is more effective and efficient than existing methods when processing datasets with arbitrary shapes. Overall, the major contributions of this paper are:

1. We propose a new neighbor method. This method combines the characteristics of natural neighbors and shared neighbors, it shows excellent performance when dealing with variable-density and noise datasets.
2. We propose a new method to generate sub-clusters on shared natural neighbors. This method first uses the shared natural neighbor graph to divide the dataset into multiple sub-clusters. Then uses a new merging strategy to merge into pairs of sub-clusters. Thus, satisfactory clustering results can be obtained on multiple datasets.
3. Experimental results on synthetic and real-world datasets show that the HC-SNaN has more advantages in detecting arbitrary-shaped clusters than other excellent algorithms.

In Section 2, we review research related to natural neighbors and shared neighbors. Section 3 introduces the main principles and ideas of shared natural neighbors and describes the algorithm proposed in this paper in detail. Section 4 demonstrates the analysis of the synthetic experimental result datasets and the real-world datasets while evaluating the algorithm's performance. Finally, in the concluding section, we summarize the main points of this paper and highlight possible future challenges.

2 Related Works

2.1 Natural Neighbor

In contrast, the natural neighbor method automatically adapts to the dataset is distribution, eliminating the need for manual K parameter selection. The effectiveness of natural neighbors has been demonstrated in various applications, including cluster analysis, instance approximation, and outlier detection [31–33]. The core idea of natural neighbors is primarily manifested in three key aspects: neighborhood, search algorithm, and the number of neighbors.

Definition 1 (Stable searching state). A stable search state can be reached if and only if the natural neighbor processes satisfy the following conditions:

$$(\forall x_i)\left(\exists x_j\right) \wedge \left(x_i \neq x_j\right) \rightarrow \left(x_i \in KNN_\lambda\left(x_j\right)\right) \wedge \left(x_j \in KNN_\lambda\left(x_i\right)\right) \tag{1}$$

The parameter λ represents the natural eigenvalue obtained through the natural neighbor algorithm. Assuming this condition is satisfied for the first time in the initial λ (where $1 \leq \lambda \leq N$) rounds of search, the stable search state for any object is characterized by the existence of one or more objects that are each other's λ-nearest neighbors. The $\mathrm{KNN}_\lambda\left(x_i\right)$ represents the λ-nearest neighbor of x_i.

Definition 2 (Natural neighbor). In a naturally stable search algorithm, data points in the immediate neighborhood are considered natural neighbors of each other. Assuming that the search state stabilizes after the λ^{th} round of search. For any data points x_i and x_j, if they are each other's neighbors, they share the following relationship:

$$x_j \in NN_\lambda\left(x_i\right) \Leftrightarrow \left(\left(x_i \in KNN_\lambda\left(x_j\right)\right) \wedge \left(\left(x_j \in KNN_\lambda\left(x_i\right)\right)\right.\right. \tag{2}$$

The $NN_\lambda\left(x_i\right)$ denotes the set of natural neighbors of point $\mathrm{x_i}$, and $NN_\lambda\left(x_\mathrm{j}\right)$ represents the set of natural neighbors of point x_j. The natural neighbor is characterized by both invariance and stability. Invariance implies that if x_i is in the natural neighbor set of x_j during the algorithm's search, then when the algorithm reaches the steady state, it remains correct that x_i is still the natural neighbor of x_j. Stability signifies that, for the same datasets, the set of natural neighbors obtained by the natural neighbor search algorithm for each point remains constant regardless of how many times the algorithm is repeated.

2.2 Shared Neighbor

A sample and its neighboring data points usually belong to the same cluster category, allowing for a more accurate assessment of the sample's local density based on its neighboring information. Specifically, the shared region encompassing a sample and its neighbors provides rich local information, aiding in a more precise depiction of the sample's distribution.

Incorrectly determining similarity can result in misclassifications. The shared region among samples plays a pivotal role in determining sample similarity and local density. Therefore, establishing sample similarity and local density based on the relationships between a sample and its neighbors is crucial for improving clustering accuracy.

Definition 3 (Shared neighbor). For any points x_i and x_j in the dataset X, $KNN\left(x_i\right)$ represents the K-nearest neighbor of x_i and $KNN\left(x_j\right)$ represents the K-nearest neighbor of x_i. The shared neighbors of x_i and x_j, denoted as:

$$SNN\left(x_i, x_j\right) = KNN\left(x_i\right) \cap KNN\left(x_j\right) \tag{3}$$

Eq. (4) defines the shared neighbor similarity between x_i and x_j.

$$\begin{cases} \dfrac{|SNN\left(x_i, x_j\right)|^2}{\sum\limits_{x_p \in SNN\left(x_i, x_j\right)} \left(d_{\left(x_i, x_p\right)} + d_{\left(x_j, x_p\right)}\right)}, & \text{if } x_i, x_j \in SNN\left(x_i, x_j\right) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

For a sample x_i, its shared neighbor local density is defined as:

$$\rho_{x_i} = \sum_{j \in L\left(x_i\right)} Sim\left(x_i, x_j\right) \tag{5}$$

where $L\left(x_i\right)$ is the set of K points with the highest similarity to x_i.

3 Proposed Algorithm

Identifying any dataset requires excellent recognition ability for complex datasets. In order to identify datasets with complex structures, it is a good way to divide complex structures into multiple simple structures. HC-SNaN algorithm inspired by this has three steps: the first step is to construct a neighborhood graph representing the structure of a dataset, and then partition the neighborhood graph into several sub-graphs and fuzzy points. The second step is to assign fuzzy points to sub-graphs according to rules. The third step is to merge sub-graphs and obtain the final clustering result. In this section, we will detail the details of the algorithm.

3.1 Shared Natural Neighbor SNaN and Shared Natural Neighbor Graph G_{SNaN}

Shared natural neighbors have better adaptability for identifying complex datasets. In comparison to the K-nearest neighbor method, the natural neighbor method is more flexible and exhibits stronger robustness. This flexibility arises from the natural neighbor method's ability to dynamically acquire neighbors based on the distribution pattern of the environment where the data points are situated. Consequently, it can more effectively adapt to datasets with varying densities and distributions, particularly suited for streaming datasets.

The shared neighbor method establishes similarity between data points based on their shared K -nearest neighbors. To ensure the validity of the similarity measure, shared neighbors require that the data points themselves share a common neighbor. This property allows shared neighbors to avoid combining a small number of relatively isolated points with high density, making it particularly sensitive to datasets with variable density.

Inspired by the advantages of the above two neighbor methods, we propose a new neighbor method-shared natural neighbor. This method is not influenced by neighbor parameters and effectively prevents over-aggregation among high-density points. It demonstrates flexibility in adapting to datasets with different density distributions, making it valuable for discovering clusters with arbitrary structures. Its specific implementation is shown in Eq. (6).

Definition 4 (Shared natural neighbor). For any two natural neighbor points n_i and n_j, $NN_\lambda\left(n_i\right)$ denotes the set of natural neighbors of n_i, and $NN_\lambda\left(n_j\right)$ denotes the set of natural neighbors of n_j. Therefore, the shared natural neighbors of n_i and n_j are denoted as:

$$SNaN\left(n_i, n_j\right) = NN_\lambda\left(n_i\right) \cap NN_\lambda\left(n_i\right) \tag{6}$$

The quantity of shared natural neighbor for the pair n_i and n_j is denoted by $\left|SNaN\left(n_i, n_j\right)\right|$.

Fig. 2 illustrates the neighbor graphs of four types of neighbors on the variable-density dataset. After conducting ten experiments, we identified optimal results for both the K-nearest neighbor and shared neighbor graphs. As shown in Fig. 2a, regardless of the chosen K value, the K-nearest neighbor graph failed to express the dataset's distribution. While the shared neighbor graph in Fig. 2b could discern the variable-density characteristics, it resulted in numerous sub-graphs that may affect the final clustering. In the variable-density dataset, the performance limit of natural neighbor so, it fails to identify the distribution characteristics of accurately of the dataset in Fig. 2c. In contrast, the shared natural neighbor graph in Fig. 2d precisely divided the dataset into two sub-graphs, effectively representing the dataset's distribution, and without generating excess sub-graphs to affect the sequel clustering process.

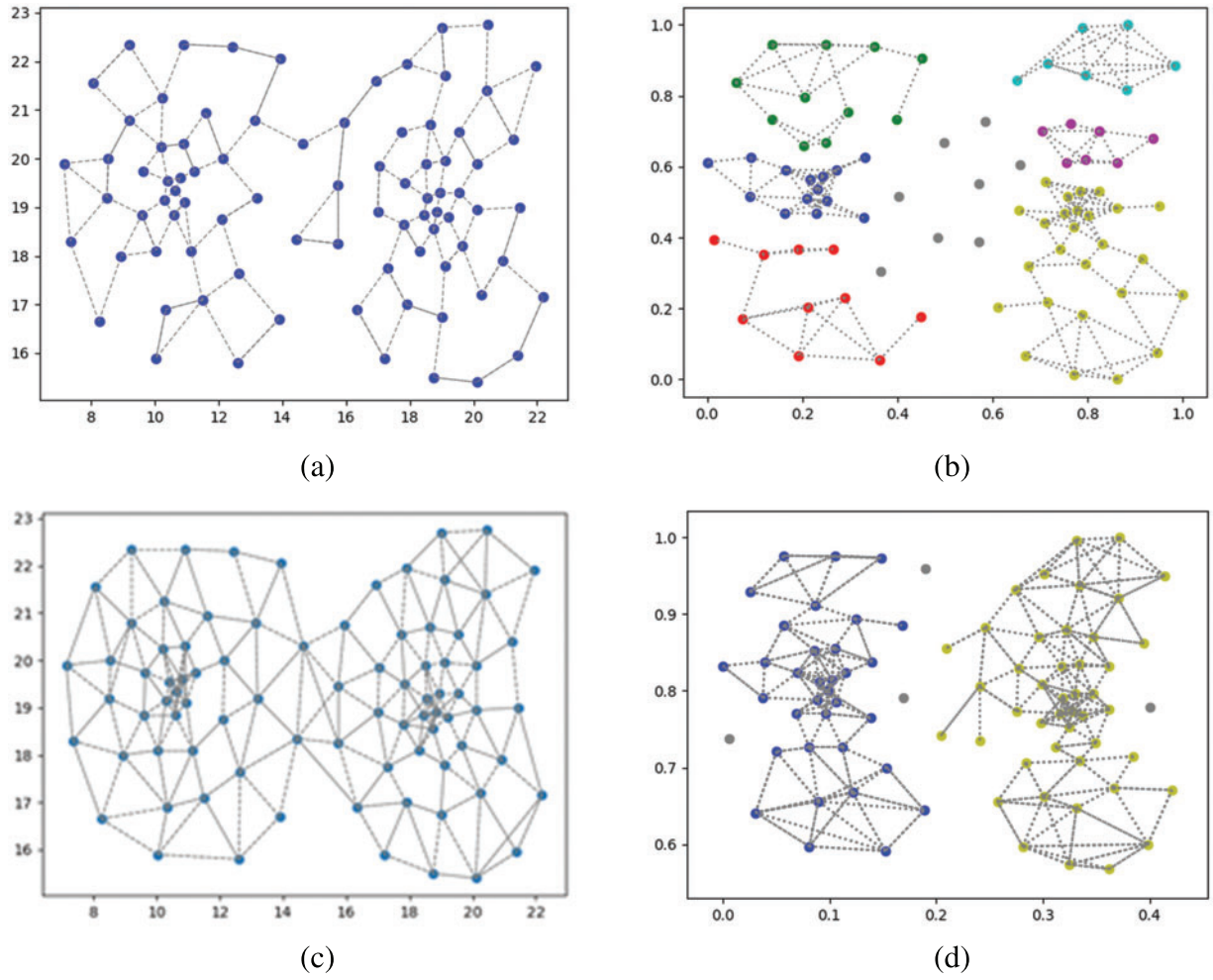

Figure 2: Neighbor graphs of the variable-density dataset. (a) K-nearest neighbor ($K = 3$) (b) shared neighbor ($K = 12$) (c) natural neighbor (d) shared natural neighbor. Gray points are fuzzy points

3.2 Hierarchical Clustering Algorithm Based on Shared Natural Neighbor

The G_{SNaN} partitions the dataset into multiple sub-clusters and fuzzy points, effectively illustrating the distribution relationship within the dataset. The subsequent steps involve assigning fuzzy points and ultimately merging sub-clusters to derive the final clustering result. In this section, we provide a detailed introduction to our proposed HC-SNaN algorithm, including a novel merging strategy for allocating fuzzy points and merging sub-clusters. A process diagram of our algorithm is presented in Fig. 3.

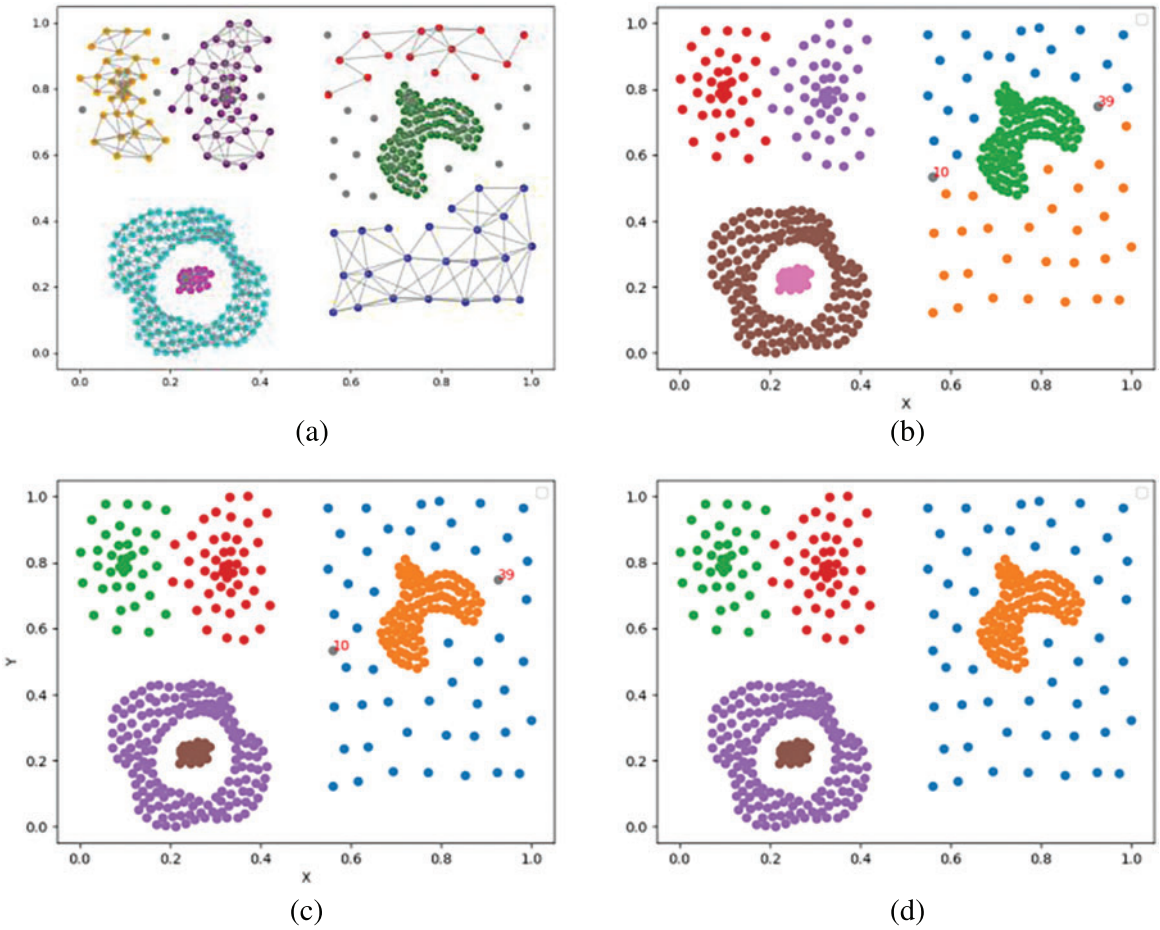

Figure 3: Assignment steps of HC-SNaN algorithm (a) Cutting the neighbor graph (b) merge the fuzzy points (c) merge the sub-clusters (d) after allocating local outliers, the final clustering results

Definition 5 (Shared natural neighbor similarity). For any two natural neighbor points n_i and n_j, $NN_\lambda(n_i)$ represents the λ natural neighbors of n_i, and the same for n_j. The shared natural neighbor similarity of n_i and n_j, denoted as:

$$\underset{(SNaN)}{SIM}\left(n_i, n_j\right) = \begin{cases} \dfrac{|SNaN\left(n_i, n_j\right)|^2}{\sum\limits_{p \in SNaN\left(n_i, n_j\right)} \left(d_{n_{ip}} + d_{n_{jp}}\right)}, & \text{if } n_i, n_j \in SNaN\left(n_i, n_j\right) \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Definition 6 (Sub-cluster core point). For natural neighbors n_i within sub-cluster C_i, the local density is obtained by summing the similarities of all shared natural neighbors of n_i. The point with the highest local density in the sub-cluster is then selected as the core point of the sub-cluster:

$$Cop\left(C_i\right) = \underset{n_i \in C_i}{argmax} \sum_{n_j \in C_i} \underset{(SNaN)}{SIM}\left(n_i, n_j\right)_{sort} \tag{8}$$

Definition 7 (Local density of natural neighbor). The local density of the natural neighbor is represented denoted as:

$$NaD\left(n_i\right) = \frac{Num\left(n_i\right)}{\underset{p \in NN_\lambda\left(n_i\right)}{max} d\left(n_i, p\right)} \tag{9}$$

where $Num\left(n_i\right)$ is the natural neighbor's number of n_i.

Definition 8 (Cluster crossover degree). For C_i and C_j, their sub-cluster crossover degree based on local density of natural neighbor is calculated, denoted as:

$$\mathrm{CCD}\left(C_i, C_j\right) = \frac{\sqrt{NaD_{Cop\left(C_i\right)} \cdot NaD_{Cop\left(C_j\right)}}}{NaD_{Cop\left(C_i\right)} + NaD_{Cop\left(C_j\right)}} \tag{10}$$

Definition 9 (Shortest distance between sub-clusters). All distances between points in sub-clusters C_i and C_j are sorted in ascending order at first. Then the shortest distance is calculated by the mean value of the first $cn_{\left(n_i, n_j\right)}$ distance, denoted as:

$$D_{\min}\left(C_i, C_j\right) = \frac{1}{cn_{\left(n_i, n_j\right)}} \sum_{1}^{cn_{\left(n_i, n_j\right)}} \sum_{\left(n_i \in c_i, n_j \in c_j\right)} \left(d_{\left(n_i, n_j\right)}\right)_{\mathrm{sort}} \tag{11}$$

where $cn_{\left(n_i, n_j\right)}$ is all points of C_i and C_j. To reduce the impact of outliers, we use the minimum 10% of the shortest distances in C_i and C_j to calculate the $cn_{\left(n_i, n_j\right)}$.

Definition 10 (Natural neighbors shared between sub-clusters). The natural neighbors shared between sub-clusters are represented as:

$$NNS\left(C_i, C_j\right) = \left(\underset{n_i \in C_i}{\cup} NN_\lambda\left(n_i\right)\right) \cap \left(\underset{n_j \in C_j}{\cup} NN_\lambda\left(n_j\right)\right) \tag{12}$$

where $\underset{n_i \in C_i}{\cup} NN_\lambda\left(n_i\right)$ is a union of all $NN_\lambda\left(n_i\right)$ of in C_i, the $\underset{n_j \in C_j}{\cup} NN_\lambda\left(n_j\right)$ is a union of all $NN_\lambda\left(n_j\right)$ of in C_j. When the intersection of natural neighbors between two clusters is not empty, it indicates a certain degree of connectivity.

In the segmentation stage of HC-SNaN, we first constructed the shared natural neighbor graph $\mathrm{G_{SNaN}}$. Subsequently, temporarily remove these points whose number of shared natural neighbors is lower than the cut threshold $C\mathrm{G_{Thr}}$ in the $\mathrm{G_{SNaN}}$. The removed points are defined as fuzzy points. This step makes the boundary between clusters clearer, it can help to discover arbitrary-shaped clusters. Among them, the $C\mathrm{G_{Thr}}$ is the user-specified number of shared natural neighbors. The $\mathrm{G_{SNaN}}$ combines the characteristics of natural neighbors and shared neighbors. It is more suitable for expressing the distribution structure of the datasets. Detailed steps are described in Algorithm 1.

Algorithm 1: Cut sub-graphs

Input: X (the dataset), CG_{Thr}(the cutting graph thresholds)
Output: C_{cut}(cut sub-clusters), f_p (fuzzy point)
Search for the λ-nearest neighbors for each data point;
Find a stable state based on the concept of natural neighbors;
Calculate $SNaN\left(n_i, n_j\right)$ according to Eq. (6);
For each pair of $SNaN\left(n_i, n_j\right)$ do
  If $\left|SNaN\left(n_i, n_j\right)\right| > CG_{Thr}$ then
    Connect this pair of $SNaN\left(n_i, n_j\right)$;
  else
    This pair of points is f_p;
  End
  The sub-graphs connected by edges are termed as C_{cut};
End

Moving to the merging phase, we calculate the distance of fuzzy points from the sub-cluster core point, as well as the average distance. Next, we determine the clusters to which the natural neighbors of the fuzzy point are subordinate. Finally, the fuzzy point is assigned to the cluster closest to the core point of the sub-cluster, provided it is smaller than the average distance to the core points of other sub-clusters. Fuzzy points that cannot be assigned are designated as local outlier points. Detailed steps are described in Algorithm 2.

Algorithm 2: Generate the initial sub-clusters

Input: X (the dataset)
Output: $C_{initial}$(the initial sub-clusters), l_o(the local outlier points)
Generate cut sub-clusters C_{cut} and fuzzy point f_p according to Algorithm 1;
For $C_{cut}\left(C_i, \ldots, C_n\right)$ do
  Calculate $Cop\left(\mathrm{C}_{\mathrm{cut}}\left(C_i\right)\right)$ based on Eq. (8);
End
For *each point* x_i *in* f_p do
  Calculate the distance d_{x} between x_i and $Cop\left(\mathrm{C}_{\mathrm{i}}, \ldots, \mathrm{C}_{\mathrm{n}}\right)$;
  Calculate the average distance $\overline{\mathrm{d}_{\mathrm{x}}}$ between x_i and $Cop\left(\mathrm{C}_{\mathrm{i}}, \ldots, \mathrm{C}_{\mathrm{n}}\right)$;
  Calculate the C_{cut} to which the natural neighbor of the x_i belongs;
  If $d_x > \overline{d_x}$ then
    Merge x_i to C_{cut} where their nearest natural neighbors belong;
  else
    x_i is defined as local outlier points l_o;
  End
End

Upon completing the assignment of fuzzy points, an initial cluster is formed. Subsequently, we calculate the cluster crossover degree, the shortest distance between sub-clusters, and the natural neighbors shared between sub-clusters. The merging process continues until the desired number of clusters is achieved. Finally, we assign the l_o to obtain the final clustering result. Detailed steps are described in Algorithm 3.

Algorithm 3: HC-SNaN

Input: X (the dataset), CG_{Thr} (the cutting graph thresholds), nc (the number of clusters)
Output: Clustering results
Generate cut sub-clusters C_{cut} and fuzzy point f_p according to Algorithm 1;
Generate sub-clusters and local outlier points l_o according to Algorithm 2;
Calculate the local density of natural neighbor $NaD(n_i)$ according to Eq. (9);
While length(sub-clusters) > nc **do**
  Calculate $CCD(C_i, C_j)$, $D_{min}(C_i, C_j)$ and $NNS(C_i, C_j)$ using Eqs. (10)–(12);
  Calculate the distance D_x between $Cop(C_i)$ and $Cop(C_j)$;
  Calculate the average distance $\overline{D_x}$ between $Cop(C_i)$ and $Cop(C_i, \ldots, C_n)$;
  Sort $CCD(C_i, C_j)$ in decreasing order;
  For *highest* $CCD(C_i, C_j)$ do
    If $|NNS(C_i, C_j)| > 0$ and $D_{min}(C_i, C_j) < ave\overline{D}_{min}(C_i, C_j)$ then
      Merge(C_i, C_j);
    End
  End
  For *highest* $CCD(C_i, C_j)$ do
    If $|D_{min}(C_i, C_j) < ave\overline{D}_{min}(C_i, C_j)$ and $D_x < \overline{D_x}$ then
      Merge(C_i, C_j);
    End
  End
End
For *each point* x_i *in* l_o do
  Calculate the natural density difference $\sum_{x_j \in KNN_\lambda(x_i)} \frac{NaD(x_i)}{NaD(x_j)}$;
  Assign x_i to the cluster whose λ-nearest neighbor has a natural density difference value close to 1;
End

The main improvement of the HC-SNaN algorithm: it constructs sparse graphs based on the shared natural neighbor construction, which not only sparse the data but also improves the rarefaction of the algorithm to different datasets. In the merging phase, the algorithm considers the interconnection and compactness within the data. This operation can better capture the features inside the cluster, which is highly effective for processing datasets with different shape densities.

3.3 Complexity Analysis

HC-SNaN mainly includes three steps: The first step is to divide the sub-graph. First, we search the natural neighbor $O(nlogn)$, construct the G_{SNaN}, and divide this graph $O(nlogn)$. The complexity of the first step is $O(nlogn)$. The second step is to assign fuzzy points. It is necessary to calculate the shared natural neighbor similarity $O(n^2)$, the sub-cluster core point $O(nlogn)$, the distance between the core points of the sub-clusters, and the average distance $O(n^2)$. The complexity of the second step is $O(nlogn + 2n^2)$. The third step is to merge sub-clusters and assign local outliers. It is necessary to calculate the degree of intersection of sub-clusters $O(n^2)$, the shortest distance of sub-clusters $O(n^2)$ and the shared natural neighbor between sub-clusters $O(n^2)$, and assign local outliers $O(n^2)$. The

complexity of the third step is $O\left(4n^2\right)$. Assuming that n is a large number, the time complexity of HC-SNaN can be regarded as $O\left(n^2\right)$.

4 Experimental Analysis

To illustrate the effectiveness of the HC-SNaN algorithm, we conducted experiments on synthetic datasets and real datasets. These datasets have different scales, different dimensions, and different flow structures. The information of these datasets is shown in Table 1.

Table 1: Details of datasets

Dataset	Samples	Attribute	Clusters	Dataset	Sample	Attribute	Clusters
Pathbased	299	2	3	Wine	178	13	3
Jain	372	2	2	Haberman	306	3	2
Compound	403	2	3	Ecoli	336	7	8
Dadom	499	2	6	Dermatology	358	34	6
Aggregation	787	2	7	BCW	683	10	2
T2-4T	4197	2	6	Mnist	810	500	7
Atom	499	3	2	Contraceptive	1473	9	3
Chainlink	499	3	2	Page-blocks	5473	10	5

4.1 Experimental Methods

To further assess the superiority of our proposed HC-SNaN algorithm, this section will present a comparison of our algorithm with seven others. These include recent algorithms: GB-DP algorithm [18], SNNDPC algorithm [22], NDP-Kmeans algorithm [10], and HC-LCCV algorithm [34]. Classical algorithms: DBSCAN algorithm [13], *K*-means algorithm [8], and DPC algorithm [17]. In Tables 2 and 3, the use of boldface indicates the optimal results obtained in the algorithm experiments. The corresponding parameter for achieving the optimal result on the dataset is denoted as Arg.

Table 2: Clustering results on synthetic datasets

Algorithm	AMI	ARI	FMI	Arg.	AMI	ARI	FMI	Arg.
	Pathbased				Jain			
DPC	0.5035	0.4196	0.6487	3.8	0.6365	0.6965	0.8739	0.9
GB-DP	0.4934	0.4338	0.6597	3	0.2193	0.0591	0.5947	2
NDP-Kmeans	0.5173	0.3092	0.3759	3/0	**1**	**1**	**1**	2/0
HC-LCCV	0.4699	0.2685	0.3730	∼	0.5061	0.3595	0.4698	∼
K-means	0.5452	0.4642	0.6629	3	0.3677	0.3241	0.7005	2
DBSCAN	0.6652	0.4573	0.7279	1.81/17	0.9713	0.9887	0.9956	2.6/16
SNNDPC	0.9005	0.9292	0.9528	9/3	0.4354	0.4060	0.7397	12/2
HC-SNaN	**0.9657**	**0.9798**	**0.9865**	1/3	**1**	**1**	**1**	0/2

(Continued)

Table 2 (continued)

Algorithm	AMI	ARI	FMI	Arg.	AMI	ARI	FMI	Arg.
	Compound				Dadom			
DPC	0.8383	0.6450	0.7288	9/6	0.6287	0.3710	0.5208	2.62/6
GB-DP	0.8003	0.7489	0.8222	6	0.7081	0.5704	0.6623	6
NDP-Kmeans	0.8211	0.4903	0.3464	0.7/6	0.7766	0.4290	0.6906	0.1/6
HC-LCCV	0.9145	0.7470	0.4422	∼	0.9057	0.7498	0.8677	∼
K-means	0.7381	0.5864	0.6798	6	0.7320	0.6305	0.7021	6
DBSCAN	0.9039	0.8220	0.9268	1.5/13	0.5729	0.5380	0.8017	0.21/2
SNNDPC	0.8471	0.8372	0.8764	16/6	0.8477	0.7354	0.7854	2/6
HC-SNaN	**0.9925**	**0.9973**	**0.9979**	1/6	**1**	**1**	**1**	0/6
	Aggregation				T2-4T			
DPC	0.9341	0.9199	0.9380	0.9	0.6933	0.4690	0.5945	3.59
GB-DP	0.6557	0.5362	0.6359	7	0.4470	0.2876	0.4473	6
NDP-Kmeans	0.9245	0.7778	0.8950	7/0.3	0.8982	0.7589	0.8965	0.1/6
HC-LCCV	0.8894	0.6792	0.4026	∼	0.0398	0.0060	0.4936	∼
K-means	0.8391	0.7112	0.7722	7	0.6547	0.4835	0.5952	6
DBSCAN	0.8166	0.6052	0.8062	0.951/4	0.8450	0.6613	0.8608	0.07/5
SNNDPC	0.9548	0.9593	0.9681	15/7	0.7064	0.5073	0.6244	16/6
HC-SNaN	**0.9794**	**0.9855**	**0.9886**	2/7	**0.9823**	**0.9784**	**0.9798**	8/6
	Atom				Chainlink			
DPC	0.1325	0.0318	0.6581	1.8/2	0.3114	0.2068	0.6593	2.29/2
GB-DP	0.2655	0.1495	0.6492	2	0.1254	0.1310	0.5979	2
NDP-Kmeans	0.0186	0.0002	0.6995	2/0.1	**1**	**1**	**1**	2/0.1
HC-LCCV	**1**	**1**	**1**	∼	**1**	**1**	**1**	∼
K-means	0.2964	0.1880	0.6559	2	0.0950	0.1269	0.5645	2
DBSCAN	0.9913	0.9653	0.9362	3.5/9	0	0	0.7064	0.51/1
SNNDPC	**1**	**1**	**1**	4/2	0.2891	0.1777	0.6531	3/2
HC-SNaN	**1**	**1**	**1**	0/2	**1**	**1**	**1**	0/2

Table 3: Clustering results on real-world datasets

Algorithm	AMI	ARI	FMI	Arg.	AMI	ARI	FMI	Arg.
	Wine				Haberman			
DPC	0.3965	0.2926	0.6192	0.6	0.0082	0.0115	0.7805	1.3
GB-DP	0.3318	0.2235	0.5832	3	−0.0039	0.0072	0.7769	2
NDP-Kmeans	0.4080	0.1671	0.3458	0.1/3	0.0034	−0.0021	0.7771	0/2
HC-LCCV	0.3925	0.2030	0.2942	∼	0.0130	0.0315	0.7390	∼
K-means	0.4288	0.2795	0.3392	3	−0.0018	−0.0039	0.5508	2
DBSCAN	0.0000	0.0000	0.5813	0.5/2	0.0000	0.0000	0.0000	1.97/15
SNNDPC	0.8763	0.8983	0.9324	18/3	0.0012	−0.0114	0.5596	7/2
HC-SNaN	**0.9465**	**0.9651**	**0.9769**	0/3	**0.0257**	**0.0415**	**0.7787**	4/2

(Continued)

Table 3 (continued)

Algorithm	AMI	ARI	FMI	Arg.	AMI	ARI	FMI	Arg.
	Ecoli				Dermatology			
DPC	0.4477	0.3750	0.6483	0.4	0.5947	0.3591	0.5665	0.4
GB-DP	0.5482	0.3661	0.5523	8	0.0165	0.0039	0.1991	6
NDP-Kmeans	0.0300	0.0024	0.5783	0.1/8	0.3798	0.0882	0.1845	0.1/6
HC-LCCV	0.4546	0.2430	0.3389	∼	0.2574	0.0472	0.2080	∼
K-means	0.5889	0.3006	0.2760	8	0.1846	0.0465	0.1104	6
DBSCAN	0.0761	0.0458	0.4140	0.8/4	0.5965	0.4147	0.5380	0.99/3
SNNDPC	0.5536	**0.5653**	0.7066	7/8	**0.8711**	0.7332	0.7876	9/6
HC-SNaN	**0.6060**	0.5544	**0.7290**	7/8	0.8420	**0.7822**	**0.8295**	6/6
	BCW				Mnist			
DPC	0.6971	0.8029	0.9118	0.2	0.8969	0.8636	0.9092	3.59
GB-DP	0.0009	0.0027	0.7346	2	0.0407	−0.0253	0.2874	6
NDP-Kmeans	0.0047	0.0170	0.4325	0.2/2	0.0184	−0.0006	0.5878	0.1/6
HC-LCCV	0.0168	0.0059	0.3032	∼	0	0	0	∼
K-means	0.0047	0.0170	0.4325	2	0.8722	0.8965	0.8965	6
DBSCAN	0.7630	0.8523	0.9318	0.6/8	−0.0743	0.0025	0.4043	0.07/5
SNNDPC	0.7910	0.8579	0.9340	10/2	0.9233	0.8755	0.9167	16/6
HC-SNaN	**0.8092**	**0.8909**	**0.9500**	9/2	**0.9295**	**0.8766**	**0.9175**	8/6
	Contraceptive				Page-blocks			
DPC	0.0103	0.0026	0.4336	0.3	0.0308	0.0273	**0.9015**	2.29/2
GB-DP	**0.2451**	**0.1497**	0.4780	3	0.0015	0.0076	0.8951	2
NDP-Kmeans	0.0150	0.0016	0.5847	0.2/3	0.0102	0.0044	0.8104	2/0.1
HC-LCCV	0.0293	0.0055	0.2979	∼	0.0336	0.047	0.7867	∼
K-means	0.0293	0.0170	0.3644	3	0.0487	−0.0105	0.6505	2
DBSCAN	−0.0017	0.0006	0.5908	1.3/4	0.0582	0.0304	0.8126	0.51/1
SNNDPC	0.0093	0.0012	0.4414	21/3	0.1272	0.053	0.6348	3/2
HC-SNaN	−0.0010	0.0003	**0.5930**	2/3	**0.1283**	**0.2048**	0.9002	0/2

For parameter settings, *K*-means and GB-DP algorithms only require specifying the number of clusters. However, due to *K*-means is instability, we conducted five experiments on each dataset to obtain optimal results. The SNNDPC algorithm requires setting the parameters *K* and the number of clusters. The NDP-Kmeans algorithm needs to set the proportion of noise points alpha and the number of clusters. HC-LCCV does not require any parameters. DBSCAN requires setting two parameters: ϵ and *minpts*.

In the experiment, we used AMI [35], ARI [36], and FMI [37] to evaluate the performance of clustering algorithms. The experiments were conducted on a desktop computer with a Core i5 3.40 GHz processor, windows 11 operating system, and 16 GB RAM, running on PyCharm 2022.

4.2 Clustering on Synthetic Datasets

Firstly, we conducted experiments on eight synthetic datasets, including two three-dimensional datasets and one noise dataset. The first five datasets are common and complex. The detailed

information on these datasets is listed in Table 1. Fig. 4 shows the clustering results of the HC-SNaN algorithm.

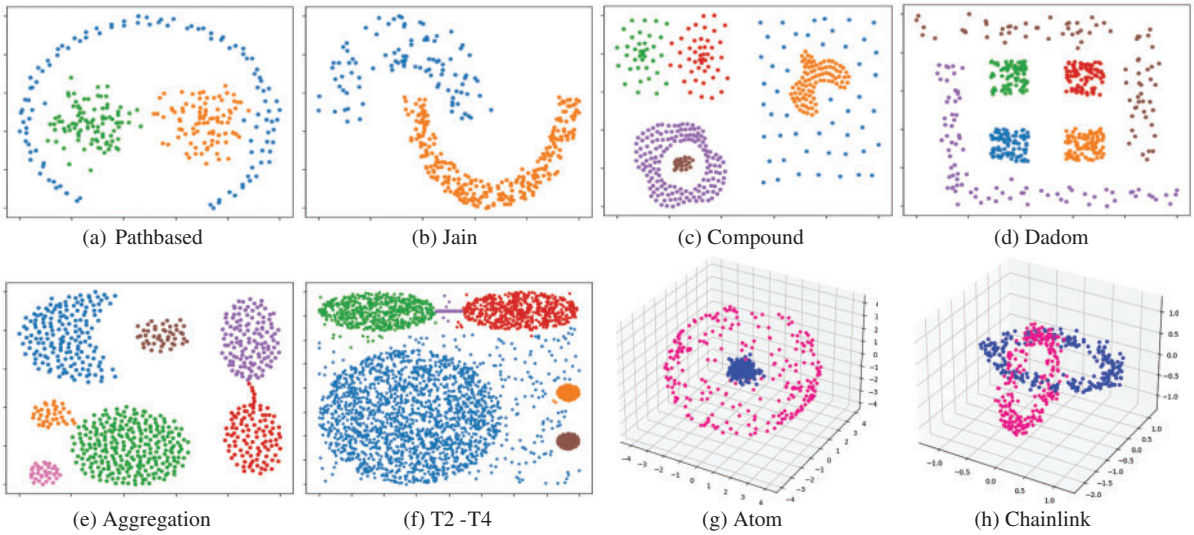

Figure 4: Clustering results of HC-SNaN on synthetic datasets (a–h)

For instance, as shown in Fig. 4a Pathbased dataset, three clusters are closely connected, yet HC-SNaN misallocated only one data point. However, in this dataset, besides the SNNDPC, the metrics of the other six algorithms indicate their failure to recognize even basic shapes. In the Jain dataset depicted in Fig. 4b, HC-SNaN perfectly identifies the dataset. Jain consists of two intertwined clusters shaped like new moons, representing a typical flow dataset with uneven densities. However, according to Table 1, the indicators of NDP-Kmeans are also 1, and SNNDPC's metrics are all above ninety-seven. Nevertheless, the other five algorithms fail to recognize this dataset. In the Dadom dataset shown in Fig. 4d, HC-SNaN metrics are all 1. Dadom comprises complex manifold clusters. As shown in Table 1, except for HC-SNaN, the other seven algorithms cannot effectively recognize this dataset. In the Aggregation dataset depicted in Fig. 4e, HC-SNaN ranks first in all metrics, while SNNDPC, DPC, and NDP-Kmeans also achieve metrics exceeding ninety. The Chainlink dataset in Fig. 4h is a three-dimension dataset. Although the structure of this dataset seems simple, but significantly different densities between two clusters. HC-SNaN, HC-LCCV, and NDP-Kmeans all yield the best results, while the remaining five algorithms fail to successfully cluster the data.

NDP-Kmeans and GB-DP are algorithms introduced in 2023, whereas SNNDPC and HC-LCCV are relatively new algorithms released within the past five years. Therefore, in this section of results analysis, we selected these four algorithms and presented the clustering outcome graphs for some datasets.

Fig. 5 illustrates the clustering results of GB-DP, HC-LCCV, SNNDPC, and NDP-Kmeans on Compound, T2-4T and Atom datasets. The first line's Compound dataset consists of six closely connected clusters merged at varying densities. Most algorithms are difficult to identify this dataset. As shown in Fig. 4c, HC-SNaN only misallocated one data point. However, none of these four algorithms could correctly identify this dataset. The second line's T2-4T is a noise dataset. Ordinary algorithms struggle to recognize noise datasets without appropriate processing, especially when the dataset's top features two ellipses and a stick-like cluster that are closely linked, easily mistaken for a single cluster.

The second line of Fig. 5 reveals that these four algorithms all fail to recognize this noise dataset. The HC-SNaN algorithm obtains sub-graphs and fuzzy points by dividing the shared neighbor graph. In general, the number of natural neighbors shared between fuzzy points is small. In the noise dataset, noise points have fewer neighbors, so they are easily detected as fuzzy points. In Fig. 4f, our algorithm exhibits excellent performance on noise datasets. The third column of Fig. 5 features a three-dimension dataset Atom, which appears simple with just two clusters, but them blends at different densities. While SNNDPC and HC-LCCV both accurately identified this dataset, GB-DP and NDP-Kmeans showed poorer clustering performance.

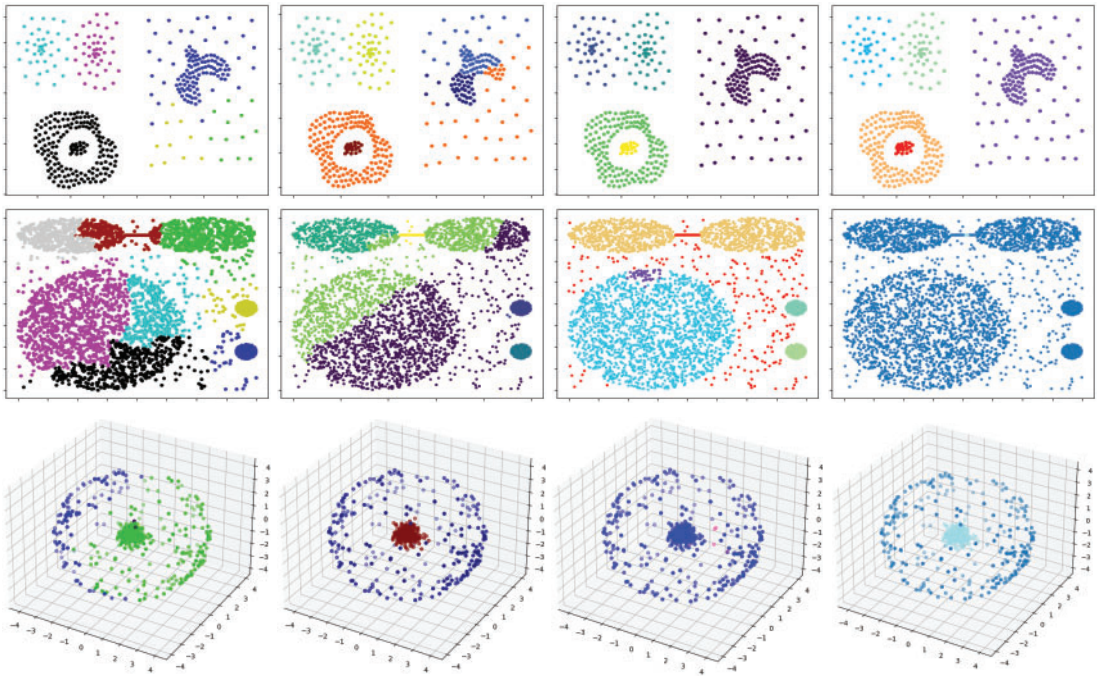

Figure 5: Results on the synthesized datasets. The first column is the GB-DP result, the second column is the SNNDPC result, the third column is the NDP-Kmeans result, and the last column is the HC-LCCV result

4.3 Clustering on Real-World Datasets

To assess the viability of the HC-SNaN algorithm for high-dimensional data, we conducted comparative experiments with seven other clustering algorithms on eight real-world datasets sourced from the UCI Machine Learning Repository. These real-world datasets vary in scale and dimensions. The clustering results, presented in Table 3 showcase the comparative results of HC-SNaN alongside the other algorithms.

The HC-SNaN consistently outperforms the other algorithms on the Wine, haberman, BCW, and mnist datasets, where metrics like AMI, ARI, and FMI rank first, with some metrics significantly surpassing other algorithms. In the Ecoli datasets, HC-SNaN achieves optimal levels of AMI and FMI metrics, with ARI ranking second. For the Dermatology dataset, HC-SNaN attains the highest ARI and FMI metrics, surpassing the other six algorithms. In the Contraceptive dataset, HC-SNaN exhibits the best FMI performance. The Page-blocks dataset shows the metrics rank first with the AMI and ARI, but the FMI compared to the DPC algorithm witch slightly inferior.

Overall, experimental verification confirms that HC-SNaN outperforms the other seven algorithms on most real-world datasets. The results underscore the algorithm's high clustering performance in discovering various shapes of clustering structures, particularly excelling in handling clustering tasks in dense regions.

5 Conclusion

In this paper, we introduce a novel neighbor method called shared natural neighbors (SNaN). The SNaN is derived by combining natural neighbors and shared neighbors, and then a graph G_{SNaN} is constructed based on SNaN. This graph accurately identifies the distribution of datasets with arbitrary shapes, providing valuable assistance for subsequent merging clustering. To showcase the superiority of our SNaN method, we propose the hierarchical clustering algorithm HC-SNaN. Experimental results on both synthetic and real-world datasets demonstrate that HC-SNaN has satisfactory clustering results across diverse datasets with different shapes and densities.

However, it is important to note that our algorithm for processing large-scale high-dimensional data may incur substantial time costs, which is an inherent limitation of hierarchical clustering. Therefore, further research is needed to explore the application of this algorithm to massive high-dimensional datasets.

Acknowledgement: The authors would like to thank the editors and reviewers for their professional guidance, as well as the team members for their unwavering support.

Funding Statement: This work was supported by Science and Technology Research Program of Chongqing Municipal Education Commission (KJZD-M202300502, KJQN201800539).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Zhongshang Chen and Ji Feng; Manuscript: Zhongshang Chen; Analysis of results: Degang Yang and Fapeng Cai. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Zheng and J. Zhao, "A new unsupervised data mining method based on the stacked autoencoder for chemical process fault diagnosis," *Comput. Chem. Eng.*, vol. 135, pp. 106755, Oct. 2020. doi: 10.1016/j.compchemeng.2020.106755.

[2] C. Fahy, S. Yang, and M. Gongora, "Ant colony stream clustering: A fast density clustering algorithm for dynamic data streams," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2215–2228, 2019. doi: 10.1109/TCYB.2018.2822552.

[3] B. Szalontai, M. Debreczeny, K. Fintor, and C. Bagyinka, "SVD-clustering, a general image-analyzing method explained and demonstrated on model and Raman micro-spectroscopic maps," *Sci. Rep.*, vol. 10, no. 1, pp. 4238, 2020. doi: 10.1038/s41598-020-61206-9.

[4] S. Cai and J. Zhang, "Exploration of credit risk of P2P platform based on data mining technology," *J. Comput. Appl. Math.*, vol. 372, no. 1, pp. 112718, 2020. doi: 10.1016/j.cam.2020.112718.

[5] P. Tavallali, P. Tavallali, and M. Singhal, "K-means tree: An optimal clustering tree for unsupervised learning," *J. Supercomput.*, vol. 77, no. 5, pp. 5239–5266, 2021. doi: 10.1007/s11227-020-03436-2.
[6] B. Bataineh and A. A. Alzahrani, "Fully automated density-based clustering method," *Comput. Mater. Contin.*, vol. 76, no. 2, pp. 1833–1851, 2023. doi: 10.32604/cmc.2023.039923.
[7] H. Xie, S. Lu, and X. Tang, "TSI-based hierarchical clustering method and regular-hypersphere model for product quality detection," *Comput. Ind. Eng.*, vol. 177, no. 2, pp. 109094, 2023. doi: 10.1016/j.cie.2023.109094.
[8] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. Fifth Berkeley Symp. Math. Stat. Probab.*, vol. 1, pp. 281–297, 1967.
[9] G. Tzortzis and A. Likas, "The MinMax k-means clustering algorithm," *Pattern Recognit.*, vol. 47, no. 7, pp. 2505–2516, 2014. doi: 10.1016/j.patcog.2014.01.015.
[10] D. Cheng, J. Huang, S. Zhang, S. Xia, G. Wang and J. Xie, "K-means clustering with natural density peaks for discovering arbitrary-shaped clusters," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, 2023. doi: 10.1109/TNNLS.2023.3248064.
[11] Q. Zhu, J. Feng, and J. Huang, "Natural neighbor: A self-adaptive neighborhood method without parameter K," *Pattern Recognit. Lett.*, vol. 80, no. 1, pp. 30–36, 2016. doi: 10.1016/j.patrec.2016.05.007.
[12] A. Fahim, "Adaptive density-based spatial clustering of applications with noise (ADBSCAN) for clusters of different densities," *Comput. Mater. Contin.*, vol. 75, no. 2, pp. 3695–3712, 2023. doi: 10.32604/cmc.2023.036820.
[13] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Second Int. Conf. Knowl. Discov. Data Min.*, Portland, OR, USA, 1996, vol. 96, pp. 226–231.
[14] Y. Chen *et al.*, "Decentralized clustering by finding loose and distributed density cores," *Inf. Sci.*, vol. 433, no. 1, pp. 510–526, 2018. doi: 10.1016/j.ins.2016.08.009.
[15] M. Daszykowski, B. Walczak, and D. L. Massart, "Looking for natural patterns in data: Part 1. Density-based approach," *Chemometr. Intell. Lab. Syst.*, vol. 56, no. 2, pp. 83–92, 2018. doi: 10.1016/S0169-7439(01)00111-3.
[16] Y. A. Geng, Q. Li, R. Zheng, F. Zhuang, R. He and N. Xiong, "RECOME: A new density-based clustering algorithm using relative KNN kernel density," *Inf. Sci.*, vol. 436, no. 4, pp. 13–30, 2018. doi: 10.1016/j.ins.2018.01.013.
[17] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014. doi: 10.1126/science.1242072.
[18] D. Cheng, Y. Li, S. Xia, G. Wang, J. Huang and S. Zhang, "A fast granular-ball-based density peaks clustering algorithm for large-scale data," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, 2023. doi: 10.1109/TNNLS.2023.3300916.
[19] S. Xia, Y. Liu, X. Ding, G. Wang, H. Yu and Y. Luo, "Granular ball computing classifiers for efficient, scalable and robust learning," *Inf. Sci.*, vol. 483, no. 5, pp. 136–152, 2019. doi: 10.1016/j.ins.2019.01.010.
[20] S. Xia, S. Zheng, G. Wang, X. Gao, and B. Wang, "Granular ball sampling for noisy label classification or imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2144–2155, 2023. doi: 10.1109/TNNLS.2021.3105984.
[21] S. Xia, X. Dai, G. Wang, X. Gao, and E. Giem, "An efficient and adaptive granular-ball generation method in classification problem," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 5319–5331, 2024. doi: 10.1109/TNNLS.2022.3203381.
[22] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Inf. Sci.*, vol. 450, no. 1, pp. 200–226, Oct. 2018. doi: 10.1016/j.ins.2018.03.031.
[23] Z. Chu *et al.*, "An operation health status monitoring algorithm of special transformers based on BIRCH and Gaussian cloud methods," *Energy Rep.*, vol. 7, pp. 253–260, 2021. doi: 10.1016/j.egyr.2021.01.07.
[24] G. Karypis, E. H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999. doi: 10.1109/2.781637.

[25] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," in *Proc. 34th Annu. Design Autom. Conf.*, Anaheim, CA, USA, 1997, no. 4, pp. 526–529. doi: 10.1145/266021.266273.

[26] Y. Zhang, S. Ding, Y. Wang, and H. Hou, "Chameleon algorithm based on improved natural neighbor graph generating sub-clusters," *Appl. Intell.*, vol. 51, no. 11, pp. 8399–8415, 2021. doi: 10.1007/s10489-021-02389-0.

[27] Z. Liang and P. Chen, "An automatic clustering algorithm based on the density-peak framework and chameleon method," *Pattern Recognit. Lett.*, vol. 150, no. 1, pp. 40–48, 2021. doi: 10.1016/j.patrec.2021.06.017.

[28] Y. Zhang, S. Ding, L. Wang, Y. Wang, and L. Ding, "Chameleon algorithm based on mutual k-nearest neighbors," *Appl. Intell.*, vol. 51, no. 4, pp. 2031–2044, 2021. doi: 10.1007/s10489-020-01926-7.

[29] E. H. Walters, "Publication of complex dataset," *Thorax*, vol. 58, no. 4, pp. 368, 2003. doi: 10.1136/thorax.58.4.368-a.

[30] J. R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Trans. Comput.*, vol. C-22, no. 11, pp. 1025–1034, 1973. doi: 10.1109/T-C.1973.223640.

[31] D. Cheng, Q. Zhu, J. Huang, L. Yang, and Q. Wu, "Natural neighbor-based clustering algorithm with local representatives," *Knowl. Based Syst.*, vol. 123, no. 317, pp. 238–253, 2017. doi: 10.1016/j.knosys.2017.02.027.

[32] L. Yang, Q. Zhu, J. Huang, and D. Cheng, "Adaptive edited natural neighbor algorithm," *Neurocomputing*, vol. 230, no. 1, pp. 427–433, Oct. 2017. doi: 10.1016/j.neucom.2016.12.040.

[33] J. Huang, Q. Zhu, L. Yang, D. Cheng, and Q. Wu, "A non-parameter outlier detection algorithm based on natural neighbor," *Knowl. Based Syst.*, vol. 92, no. 3–4, pp. 71–77, 2016. doi: 10.1016/j.knosys.2015.10.014.

[34] D. Cheng, Q. Zhu, J. Huang, Q. Wu, and L. Yang, "A novel cluster validity index based on local cores," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 985–999, 2019. doi: 10.1109/TNNLS.2018.2853710.

[35] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clustering's comparison: Is a correction for chance necessary," in *Proc. 26th Annu. Int. Conf. Machine Learning*, Montreal, Qc, Canada, 2009, pp. 1073–1080. doi: 10.1145/1553374.1553511.

[36] S. Zhang, H. S. Wong, and Y. Shen, "Generalized adjusted rand indices for cluster ensembles," *Pattern Recognit.*, vol. 45, no. 9, pp. 2214–2226, 2012. doi: 10.1016/j.patcog.2011.11.017.

[37] M. Meila, "Comparing clusterings—an information based distance," *J. Multivar. Anal.*, vol. 98, no. 5, pp. 873–895, 2007. doi: 10.1016/j.jmva.2006.11.013.