# An Enhanced Hierarchical Clustering Algorithm: Methodology and Application

Xin Liu[1], Hang Li[1], and Shoulin Yin[1]

1. College of Artificial Intelligence, Shenyang Normal Uinversity
110034 Shenyang, China
Corresponding author: Hang Li and Shoulin Yin

*Received Nov. 16, 2025; Revised and Accepted Dec. 3, 2025*

**Abstract.** Hierarchical clustering is a widely used technique for data grouping in various fields, renowned for its ability to create a dendrogram that provides insights into data structure. However, traditional hierarchical methods often suffer from challenges such as computational inefficiency, sensitivity to noise, and difficulty in determining the optimal number of clusters. This paper introduces an improved hierarchical clustering algorithm that incorporates advanced distance metrics, the use of cluster representatives, and a robust agglomeration strategy designed to address these challenges. The proposed algorithm incorporates a mechanism to dynamically select the distance metric based on the underlying data characteristics. Unlike traditional methods that rely solely on a fixed distance metric, our approach determines the most suitable metric for the given dataset. For instance, a method like Cosine similarity could be employed for high-dimensional data or text data where orientation matters, while Euclidean distance can be effective for low-dimensional, continuous data. Our proposed method is evaluated through comprehensive experiments on synthetic and real-world datasets, showcasing significant enhancements in clustering performance, adaptability to noise, and computational efficiency. The findings indicate that the proposed algorithm outperforms traditional hierarchical methods, demonstrating its potential for broader applications across various domains.

**Keywords:** Hierarchical clustering, Advanced distance metrics, Robust agglomeration strategy, Cosine similarity.

## 1. Introduction

Generally, clustering refers to the process of dividing an unlabeled data set into several categories such that the data within a category is highly similar while the data between categories is less similar. It is an unsupervised machine learning classification method [1,2]. Clustering algorithms can be roughly classified into five categories: partition-based (k-means [3]), density-based (DBSCAN [4], CTW-DPC [5]), grid-based (STING [6]), model-based (such as Gaussian mixture model [7]), and hierarchical clustering algorithm [8]. Among them, hierarchical clustering algorithms are widely used in various fields such as discovering nested structures between texts, spatial hierarchies of objects in images, dynamic associations of community evolution in social networks [9,10], and construction of species evolutionary trees in biology.

Hierarchical clustering can organize data in a tree structure, thereby revealing the implicit relationships or patterns among the data. On one hand, the scalability of traditional hierarchical clustering algorithms is poor. Since most traditional hierarchical clustering algorithms require calculating the pairwise distances of clusters, the time cost of the algorithm is too high. On the other hand, most algorithms need to repeatedly calculate the entire data during the iteration process, and the storage of intermediate results leads to excessive memory usage. In summary, the current hierarchical clustering algorithms cannot well balance the consumption of time and space.

The Chameleon algorithm [11] proposes an adaptive similarity measurement and a two-stage merging strategy, which has made certain progress in handling non-spherical clusters and unevenly distributed data. However, it still has the following main problems. Firstly, when constructing a sparse graph, it is sensitive to the value of $k$. When $k$ is too small or too large, it is prone to causing cluster fragmentation or incorrect connections, and in high-noise scenarios, due to the sensitivity of constructing the k-nearest neighbor graph to noise, it will introduce incorrect mergers, and the mergers are difficult to cancel, which easily leads to the unreasonableness of subsequent clustering results; Secondly, the construction and update of the k-nearest neighbor graph have a significant increase in computational complexity on large-scale or high-dimensional data; Thirdly, the similarity measurement issue. The Chameleon algorithm defines relative connectivity as the sum of edge weights and relative compactness as the average weight of cut edges. When the density within the cluster changes, relative connectivity and relative compactness cannot accurately reflect the distribution characteristics of the data, thereby leading to poor clustering results. In recent years, to solve the problem of constructing sparse graphs, most have improved by using other algorithms to construct sparse graphs [12]. Generally, parameters are introduced to describe density changes [13], and the clustering method of minimum spanning tree (MST) [14] is focused on, but there is still a lack of effective strategies for combining statistical features for sample allocation.

Hierarchical clustering can organize data into a tree structure, facilitating users' reading and analysis of the data. Depending on the construction method, hierarchical clustering can be divided into agglomerative and divisive types. The agglomerative hierarchical clustering algorithm merges similar clusters through the proximity between clusters during the iterative process, constructing a clustering tree from the bottom up. However, regardless of the aggregation method (single connection, full connection, etc.), a large number of pairwise distance calculations are required. In contrast, HK-means (Hierarchical K-means) [15] is a typical divisive algorithm, which divides the current clusters iteratively using the K-means algorithm and constructs a clustering tree from the top down. This type of algorithm has a simple principle and wide application, but its high complexity undoubtedly restricts the application prospects of this type of algorithm in large-scale data.

Most clustering algorithms fail to achieve satisfactory results when dealing with clusters of different shapes and densities. At this point, the density-based clustering algorithm DBSCAN is introduced. This algorithm defines the core objects based on the concept of density reachability. When choosing the appropriate radius parameter $Eps$ and the sample number $Minpts$, DBSCAN can discover clusters of different shapes. However, the parameters are often difficult to determine. Later, Yu et al. [16] proposed the Density Peaks Clustering (DPC) algorithm. Compared with DBSCAN, DPC was easier to determine the parameters and had higher clustering accuracy. However, its method based on the distance matrix leaded to little improvement in scalability. Subsequent works have various improved schemes based on DPC, such as the CFSFDPHD algorithm which designs a non-parametric density estimation method to optimize local density. The DPCG algorithm uses grid objects instead of corresponding data objects to accelerate calculation. The VDPC [17] introduces a new variational density peak clustering algorithm to solve the problem of discriminating cluster density changes, aiming to execute the clustering task in an orderly and autonomous manner, especially suitable for data sets with diverse density distributions. The DPC-NNMS [18] designs a density peak clustering algorithm based on the natural neighborhood merging strategy, by identifying the natural neighbor set of each data, adaptively obtaining its local density.

In applications involving large-scale clustering, density-based algorithms are often limited by the high time and space costs brought about by global density estimation calculations. Therefore, researchers have proposed various optimization schemes, such as reducing distance calculations based on sampling strategies, and improving allocation strategies based on the K-nearest neighbor assumption, etc. However, regardless of the method adopted, the balance between time and space costs still needs to be addressed. The application of nearest neighbor relationships in clustering algorithms, especially in large-scale data processing, has attracted much attention because of its low time cost. Moreover, the data structure constructed by nearest neighbor relationships inherently has the characteristics of high-density regions, and reciprocal nearest neighbors can be regarded as the local density-maximizing regions within that area. By using an improved local density strategy to select more suitable representative points from reciprocal nearest neighbors, no intermediate variables are generated, thus significantly reducing the spatial consumption. In conclusion, this strategy can effectively handle the processing of large-scale data.

The motivation behind improving hierarchical clustering lies in growing data complexities in contemporary applications, where traditional methods often fail to provide satisfactory results. By addressing the inefficiencies inherent in these methods, our research seeks to develop a more effective technique that is both robust and scalable, enabling practitioners to glean insightful patterns from increasingly large and noisy datasets.

## 2. Proposed Hierarchical Clustering Algorithm

Density-based Scalable Hierarchical Clustering (DBSC) is a bottom-up hierarchical clustering algorithm. Its main process consists of three parts: (1) Construction of the nearest neighbor graph NNCsCon; (2) Reconnection of isolated reciprocal nearest neighbors iNNCsRep; (3) Selection of representative points RootsDet. The execution process of DBSC is shown in **Algorithm 1**.

Firstly, the data is stored in the data object set $r$, which serves as the input for subsequent calculations and records the root nodes during each iteration process. Then, the algorithm enters the iterative process. The iterative process is the main body of DBSC. Through constructing the nearest neighbor graph, reconnecting isolated reciprocal nearest neighbors, and detecting representative points, the nearest neighbor component set $T$ is updated iteratively. The cycle process is controlled by the maximum number of iterations $t$. After the iteration terminates, $LABELING$ regards each nearest neighbor component in the set $T$ as a cluster, and marks the nodes belonging to different nearest neighbor components. This marking will serve as the clustering result and be output.

Figure 1 is an example of the DBSC algorithm, where the distribution of the initial nodes is shown in Figure 1(a). Figure 1(b) illustrates the workflow of the DBSC algorithm. Firstly, NNCsCon calculates the nearest neighbors for each initial node and connects them together. Once all nodes are connected to their corresponding nearest

**Algorithm 1** DBSC method

```
Input: Data $X$, truncation threshold $\alpha$, maximum number of iterations $t$
Output: Data label $L$;
 1: $r \leftarrow X$;
 2: for all $i = 0, 1, \cdots, t$ do
 3:     $T \leftarrow NNCsCon(r)$;
 4:     By $iNNCsRep(T)$ to update $T$;
 5:     By $RootsDet(T, \alpha)$ to update $T$;
 6: end for
 7: if $|T| = 1$ then break then
 8:     $L \leftarrow LABELING(T)$;
 9: else
10:     none;
11: end if
12: return $L$
```

neighbors, a set of nearest neighbor components (represented by different colors) is obtained, where the nodes connected by the bidirectional arrows are a pair of reciprocal nearest neighbors R. Then, for the isolated nearest neighbor component $t$, $iNNCsRep$ will calculate and locate the nearest neighbor component $t$ to which it is most similar, and add an edge (indicated by a dotted line) between $r$ and its corresponding $t$. Finally, $RootsDet$ selects the appropriate representative points based on the local density (square nodes) and these representative nodes will become the initial nodes for the next round of iteration. By repeatedly performing the above steps, a clustering tree can be further constructed. Eventually, as shown in Figure 1(c), when the candidate nodes are unique (red triangle), the DBSC algorithm terminates.

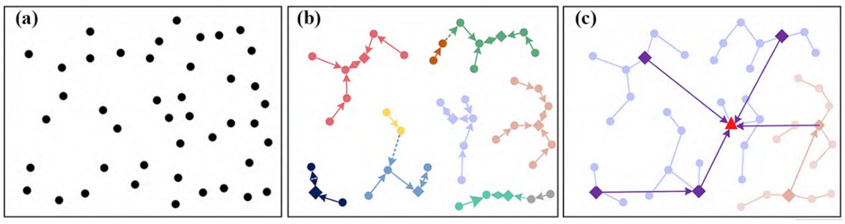

**Fig. 1.** The processing of DBSC

## 2.1. Constructing the Nearest Neighbor Component

Constructing the nearest neighbor component is the fundamental step of DBSC. Its main purpose is: (1) to divide the dataset based on the nearest neighbor relationship; (2) to locate the reciprocal nearest neighbors on each nearest neighbor component, using the Ball-Tree algorithm [19] to quickly obtain the nearest neighbor relationship between nodes, then connect each nearest neighbor pair. If it is found that a pair of nodes are repeatedly connected, it indicates that these two nodes are reciprocal nearest neighbors. Finally, starting from the reciprocal nearest neighbors, a breadth-first search is executed to obtain multiple nearest neighbor components. In addition, if there are isolated nearest neighbor components, re-optimization through $iNNCsRep$ is carried out. And $RootsDet$ will select the best representative point in the reciprocal nearest neighbors based on local density.

## 2.2. Filtering Strategy Based on Nearest Neighbor Distance

In this paper, a local distance threshold of $\delta$ is adopted as the filtering condition for the edges, and $\delta$ is used to divide the MST of the dataset into several connected components. Each connected component consists of a group of closely connected nodes, representing potential clusters. When the edge weight between two nodes exceeds $\delta$, they no longer belong to the same connected component. The filtering strategy process is shown in **Algorithm 2**.

**Definition 1**. Average Nearest Neighbor Distance. For each node $v_j$, let $d_i$ denote the distance (i.e., edge weight) from $v_i$ to its nearest neighbor. Then, the average nearest neighbor distance $\mu_{NN}$ of the node is defined as:

**Algorithm 2** Clustering filtering algorithm

```
Input: Dataset $D = p_1, p_2, \cdots, p_n$;
Output: Connected component set $C = C_1, C_2, \cdots, C_m$, outlier set $O \subseteq D$;
    Calculate the distance between each point pair, obtain the distance matrix $M$, calculate the average nearest neighbor distance $\mu_{ANN}$ according to Equation (1), calculate the local distance threshold $\delta$ according to Equation (3), and construct the minimum spanning tree $T$ based on $M$.
 2: for all each $e \in T$ do
        if $w(e) > \delta$ then then
 4:         remove edge $e$;
        end if
 6: end for
    Initializing $O = \emptyset$;
 8: for all each $r \in T$ do
        if $deg(v) == 0$ then
10:         Add $v$ into $O$;
        end if
12: end for
```

$$\mu_{NN} = \frac{1}{n}\sum_{i=1}^{n} d_i. \tag{1}$$

The average nearest neighbor distance reflects the local density distribution characteristics of the dataset. Specifically, a lower value indicates that the data is densely distributed in the local area, while a higher value suggests that the data is sparsely distributed or there is noise interference. The nearest neighbor distance of noise points is usually significantly greater than the edge weights within the core clusters. Therefore, $\mu_{NN}$ can be used as a reference benchmark to distinguish the core area from noise.

**Definition 2**. Variable coefficient. For dataset $D$ with mean value $\mu$, standard deviation $\sigma$, its coefficient $c_v$ of variation is defined as:

$$c_v = \frac{\sigma}{\mu}. \tag{2}$$

The coefficient of variation is a relative measure of the dispersion of a dataset $D$.

**Definition 3**. Local distance Threshold. The local distance threshold $\delta$ is defined as:

$$\delta = (1 + c_v) \cdot \mu_{NN}. \tag{3}$$

Where $c_v$ is the coefficient of variation of the dataset. $\delta$ can effectively filter noisy edges while preserving the integrity of the core clusters and ensure the tightness of the clustering results. For example, when $c_v = 1.5$, all MST edges with edge weights exceeding $2.54\mu_{NN}$ will be cut off to form the set of connected components.

## 2.3. The Outlier Assignment Strategy

In the hierarchical clustering framework, after the merging strategy based on statistical features, the algorithm has generated a set of clustering results $C_{merge}$. However, in practical applications, users usually want to specify the target number of clusters $k$ to obtain clustering results with a specific granularity. To this end, an adaptive adjustment strategy is proposed in this section to gradually reduce the number of clusters to a user-specified target value by iteratively merging the most similar cluster pairs.

Specifically, after the initial clustering set $C_{merge}$ is generated based on the merging strategy, if the number of clusters is equal to $k$, the result is directly output. If it is larger than $k$, it needs to be further merged into $k$ clusters. The merging process follows the following criterion: the cluster pair with the highest inter-cluster similarity is preferred, that is, the cluster pair with the smallest edge weight across clusters. In this stage, the key is to balance the global similarity measure and the local density distribution characteristics to ensure that the merged clusters not only meet the needs of users, but also maintain the tightness of the internal structure. If it is less than $k$, it will prompt the user that the optimal clustering result is less than the specified $k$ value.

The assignment of outliers should be carried out on the premise of ensuring the stability of the original cluster structure. According to Theorem 2, outliers are connected to the core cluster by only one long edge, and their assignment should avoid disturbing the topological integrity of the core cluster. To this end, we propose a conservative allocation strategy:

(1) Calculating the distance of the outlier to each cluster.

For each outlier $p$, its distance to each cluster $C_i$ is calculated. Here, the distance is the distance from the outlier $p$ to the closest point in cluster $C_i$:

$$d(p, C_i) = min_{q \in C_i} d(p, q). \tag{4}$$

Where $d(p, q)$ is the distance between points $p$ and $q$.

(2) Selecting the nearest cluster for assignment.

Find the minimum distance between an outlier $p$ and all clusters:

$$d_{min}(p) = min_{C_i} d(p, C_i). \tag{5}$$

Assign the outlier $p$ to the nearest cluster $C_{final}$:

$$C_{final} = argmin_{C_i} d(p, C_i). \tag{6}$$

Where $argmin$ returns the sequence number of the minimum value.

(3) The cluster structure is not updated after allocation.

After the outlier $p$ is assigned to cluster $C_{final}$, the structure of cluster $C_{final}$ is not updated, that is, the assignment of other points in the cluster is not readjust. In this way, the tightness of the original clusters can be maintained and noise propagation can be prevented.

Integrating the filtering, merging, merging according to the specified number of clusters, and outlier assignment process of the algorithm, the proposed final clustering process in this paper is shown in **Algorithm 3**.

**Algorithm 3** Clustering filtering algorithm

```
Input: Dataset $D = p_1, p_2, \cdots, p_n$, merging coefficient $\alpha$, the specified number of clusters $k$;
Output: Final clustering result $C_{final}$;
    Algorithm 1 is used to obtain the set $C$ of connected components and the set $O$ of outliers.
 2: The clustering results $C_{merge}$ are obtained using Algorithm 2.
    While $|C_{merge}| > k$ do
 4: Initializing the queue $Q$.
    for each $() \in C_{merge}$ do
 6: Find the minimum connected edge weight $w_{ij}$ and insert $(C_i, C_j, w_{ij})$ into queue $Q$ in ascending order of $w_{ij}$.
    End for
 8: Take the least weight element $(C_i, C_j, w_{ij})$ of $Q$ and merge clusters $C_i$ and $C_j$.
    The connection side information between the new cluster and its neighboring clusters is recomputed, and the queue $Q$ is updated.
10: end While
    for each $o \in Q$ do
12: Calculating the nearest distance between the outlier and each cluster.
    end for
```

### 2.4. Analysis of Algorithm Complexity

The time complexity of the proposed clustering algorithm mainly depends on the construction of MST, filtering strategy, merging strategy, and specified number of cluster merging. If $n$ is the number of data points, If $m$ is the initial number of clustering units, the time complexities are $O(n \log n)$, $O(n)$ and $O(m^2 \log m)$ respectively, and the time complexity of merging into $k$ clusters is $O((m-k) \log m)$, so the time complexity of the algorithm is $O(n \log n)$.

## 3. Experimental Results and Analysis

To verify the accuracy of the proposed algorithm in this paper, it is compared with the Chameleon algorithm [20], CURE algorithm [21], CH algorithm [22], and GBSC algorithm [23] on 30 datasets. The Chameleon algorithm is a graph-based hierarchical clustering algorithm. Its core process consists of four stages: constructing the k-nearest neighbor graph, graph partitioning, calculating cluster similarity, and iterative merging of clusters. For a dataset consisting of $n$ sample points, each with a dimension of $d$, the time complexities are $O(n^2 d)$, $O(kn + n \log n)$,

$O(c^2)$, and $O(c \log c)$, respectively. Here, $c$ represents the number of small clusters being merged. Therefore, the complexity is higher when there are more small clusters. The CURE algorithm is a density-based and hierarchical clustering algorithm that captures the shape of clusters by selecting representative points and is suitable for non-spherical clusters and clusters of different sizes. The time complexity of the algorithm is mainly determined by initialization, cluster distance calculation, merging operation and representative point update. For the data set containing $n$ sample points, the time complexities are $O(n)$, $O(c^2)$, $\sum_{c=k+1}^{n} O(c^2)$ and $O(n)$, respectively. Therefore, the time complexity of the algorithm is $\sum_{c=k+1}^{n} O(c^2) \approx O(n^3)$, where $c$ is the current number of clusters. BIRCH algorithm is an efficient clustering algorithm for large-scale data sets. Its advantage is to reduce the time and space overhead by using clustering features and CF tree to achieve data compression, and its time complexity is $O(n)$. GBSC algorithm is a spectral clustering algorithm based on granular spheres. The core idea is to represent the data with multiple granularities through granular spheres, which greatly reduces the computational complexity of traditional spectral clustering. The core process includes pellet generation, similarity matrix construction, spectral clustering and label assignment. For a data set containing $n$ sample points and each sample point dimension is $d$, the time complexity is $O(n^2 d)$, $O(g^2 d)$, $O(g^3)$, $O(ng)$ by constructing the clustering process of $g$ pellets. If the number of particles is far less than the number of sample points, the overall time complexity of the algorithm can be reduced to $O(n^2 d)$.

These 30 datasets include 20 two-dimensional artificial datasets [24] and 10 real-world datasets from UCI [25], including the Pendigits dataset with 10992 samples and the Internet-ad dataset with 1558 features. Detailed information is shown in Tables 1 and Table 2. Since the parameters of the optimal solution obtained by the compared algorithms are not the same, this paper adopts the grid search method to search the optimal parameters of each algorithm.

**Table 1.** Synthetic datasets

| | | | |
|---|---|---|---|
| 3MC | 300 | 2 | 3 |
| 3-spiral | 312 | 2 | 3 |
| Atom | 800 | 2 | 3 |
| Cassini | 1 000 | 2 | 3 |
| Complex8 | 2 551 | 2 | 8 |
| Complex9 | 3 031 | 2 | 9 |
| Compound | 399 | 2 | 6 |
| Cure-t0-2000n-2D | 2 000 | 2 | 3 |
| Dartboard2 | 1 000 | 2 | 4 |
| Donut3 | 999 | 2 | 3 |
| Lsun | 400 | 2 | 3 |
| Shapes | 1 000 | 2 | 4 |
| Sizes5 | 1 000 | 2 | 4 |
| Smile1 | 1 000 | 2 | 4 |
| Spherical_6_2 | 300 | 2 | 6 |
| Target | 770 | 2 | 6 |
| Triangle2 | 1 000 | 2 | 4 |
| Wingnut | 1 016 | 2 | 2 |
| Zelnik3 | 266 | 2 | 3 |
| Zelnik5 | 512 | 2 | 4 |

### 3.1. Experimental Results on Artificial Datasets

The performance of the clustering algorithms was evaluated on 20 artificial datasets. The experimental results are shown in Table 3. The experiment uses two indicators to evaluate the clustering quality from different perspectives. FMI quantifies the matching degree between the clustering results and the true labels by combining precision and recall rate. Its value range is [0,1] with a higher value indicating a better clustering effect. AMI is an information-theoretic evaluation method used to measure the correlation between the clustering results and the true labels. Its value range is [-1,1] with a higher value indicating a higher consistency between the two.

As can be seen from Table 3 and Table 4, the performance of the proposed algorithm on Complex8, Complex9, Compound and other datasets has been improved. For 3-spiral and Atom datasets, the proposed method achieves 100% clustering accuracy (AMI=1.0) due to the global connectivity advantage of MST, while Chameleon has low accuracy due to the local connection fragmentation problem of sparse graph constructed by K-nearest neighbor

**Table 2.** Real-world datasets

| Dataset | Sample number | Feature number | Category number |
|---|---|---|---|
| Breast-cancer-wisconsin | 699 | 9 | 2 |
| Dermatology | 366 | 34 | 6 |
| Haberman | 306 | 3 | 2 |
| Internet-ad | 3 279 | 1 558 | 2 |
| Iris | 150 | 4 | 3 |
| Newthyroid | 215 | 5 | 3 |
| Pendigits | 10 992 | 16 | 10 |
| WDBC | 569 | 30 | 2 |
| Wine | 178 | 13 | 3 |
| Zoo | 101 | 16 | 7 |

algorithm. In addition, the proposed method can effectively isolate the low density noise area in the Sizes5 data set with significant density difference, and the FMI is 0.9915, which is better than other comparison algorithms. The visualization results in Figure 2 further verify the adaptability of the proposed method to structurally non-convex datasets.

**Table 3.** Result of AMI on synthetic datasets

| Dataset | AMI | AMI | AMI | AMI | AMI |
|---|---|---|---|---|---|
| Method | Proposed | Chameleon | CURE | BIRCH | GBSC |
| 3MC | 1.000 0 | 0.754 2 | 1.000 0 | 0.491 2 | 0.961 9 |
| 3-spiral | 1.000 0 | 0.410 7 | 0.046 6 | 0.026 7 | 0.002 3 |
| Atom | 1.000 0 | 1.000 0 | 0.306 0 | 0.288 9 | 0.000 0 |
| Cassini | 1.000 0 | 1.000 0 | 0.738 4 | 1.000 0 | 0.670 7 |
| Complex8 | 0.944 7 | 0.742 1 | 0.701 2 | 0.580 1 | 0.715 2 |
| Complex9 | 0.968 6 | 0.783 6 | 0.671 2 | 0.706 0 | 0.746 0 |
| Compound | 0.804 1 | 0.742 8 | 0.736 9 | 0.720 2 | 0.419 3 |
| Cure-t0-2000n-2D | 1.000 0 | 0.590 8 | 0.626 1 | 0.498 6 | 1.000 0 |
| Dartboard2 | 1.000 0 | 0.577 6 | 0.532 8 | 0.463 5 | 0.000 0 |
| Donut3 | 1.000 0 | 1.000 0 | 0.291 6 | 0.000 0 | 0.982 6 |
| Lsun | 1.000 0 | 1.000 0 | 1.000 0 | 0.722 7 | 1.000 0 |
| Shapes | 1.000 0 | 0.842 6 | 1.000 0 | 0.974 3 | 1.000 0 |
| Sizes5 | 0.952 3 | 0.546 8 | 0.634 3 | 0.898 0 | 0.952 5 |
| Smile1 | 1.000 0 | 1.000 0 | 0.590 5 | 0.766 0 | 0.995 2 |
| Spherical_6_2 | 1.000 0 | 0.648 2 | 1.000 0 | 0.613 9 | 0.842 6 |
| Target | 1.000 0 | 0.564 4 | 0.392 8 | 0.583 5 | 0.720 2 |
| Triangle2 | 0.981 5 | 0.977 7 | 0.854 8 | 0.859 9 | 0.972 8 |
| Wingnut | 1.000 0 | 1.000 0 | 1.000 0 | 0.602 6 | 0.553 4 |
| Zelnik3 | 1.000 0 | 0.686 8 | 0.519 9 | 0.666 5 | 1.000 0 |
| Zelnik5 | 1.000 0 | 0.592 8 | 0.668 3 | 0.589 5 | 1.000 0 |

In Figure 2, each row corresponds to a dataset and each column corresponds to an algorithm. It can be seen that the proposed algorithm performs better than the comparison algorithms on the data sets with ribbon, spiral and large density differences.

## 3.2. Experimental Results on Real Datasets

Table 5 shows the comparison results of each algorithm on 10 real data sets. Compared with the comparison algorithm, the proposed algorithm has improved on most data sets. For example, in the Iris dataset, the AMI and FMI of the proposed method are 0.8287 and 0.8999, which are better than 0.8032 and 0.8407 of Chameleon. This is mainly attributed to the robustness of its dynamic statistical merging strategy to category overlap. Although the performance on datasets such as Breast-cancer-wisconsin, Newthyroid, Pendigits is not as good as that of the comparison algorithms, the overall performance is stable and ranked high, and there is little difference with the optimal evaluation index results. Through the average ranking of the evaluation index, it can be seen that the average ranking of the proposed method is 1.6 for AMI and 1.7 for FMI, which is higher than the comparison algorithm and better than other algorithms.

**Table 4.** Result on synthetic datasets

| Dataset | FMI | FMI | FMI | FMI | FMI |
|---|---|---|---|---|---|
| Method | Proposed | Chameleon | CURE | BIRCH | GBSC |
| 3MC | 1.000 0 | 0.812 9 | 1.000 0 | 0.577 4 | 0.984 7 |
| 3-spiral | 1.000 0 | 0.557 3 | 0.387 0 | 0.350 2 | 0.339 8 |
| Atom | 1.000 0 | 1.000 0 | 0.658 1 | 0.653 9 | 0.706 7 |
| Cassini | 1.000 0 | 1.000 0 | 0.779 0 | 1.000 0 | 0.703 0 |
| Complex8 | 0.927 1 | 0.626 1 | 0.611 6 | 0.467 6 | 0.647 8 |
| Complex9 | 0.946 3 | 0.589 3 | 0.472 3 | 0.541 8 | 0.585 0 |
| Compound | 0.830 4 | 0.614 6 | 0.673 7 | 0.637 6 | 0.512 7 |
| Cure-t0-2000n-2D | 1.000 0 | 0.698 5 | 0.773 5 | 0.669 0 | 1.000 0 |
| Dartboard2 | 1.000 0 | 0.632 5 | 0.586 1 | 0.539 2 | 0.499 2 |
| Donut3 | 1.000 0 | 1.000 0 | 0.547 6 | 0.576 8 | 0.994 0 |
| Lsun | 1.000 0 | 1.000 0 | 1.000 0 | 0.719 2 | 1.000 0 |
| Shapes | 1.000 0 | 0.816 9 | 1.000 0 | 0.984 2 | 1.000 0 |
| Sizes5 | 0.991 5 | 0.655 4 | 0.755 4 | 0.978 1 | 0.992 6 |
| Smile1 | 1.000 0 | 1.000 0 | 0.612 4 | 0.785 0 | 0.997 9 |
| Spherical_6_2 | 1.000 0 | 0.522 8 | 1.000 0 | 0.404 6 | 0.851 3 |
| Target | 1.000 0 | 0.614 4 | 0.667 0 | 0.766 9 | 0.833 2 |
| Triangle2 | 0.991 3 | 0.989 7 | 0.896 6 | 0.869 7 | 0.990 7 |
| Wingnut | 1.000 0 | 1.000 0 | 1.000 0 | 0.853 1 | 0.793 9 |
| Zelnik3 | 1.000 0 | 0.690 6 | 0.609 7 | 0.671 0 | 1.000 0 |
| Zelnik5 | 1.000 0 | 0.562 2 | 0.649 6 | 0.560 2 | 1.000 0 |

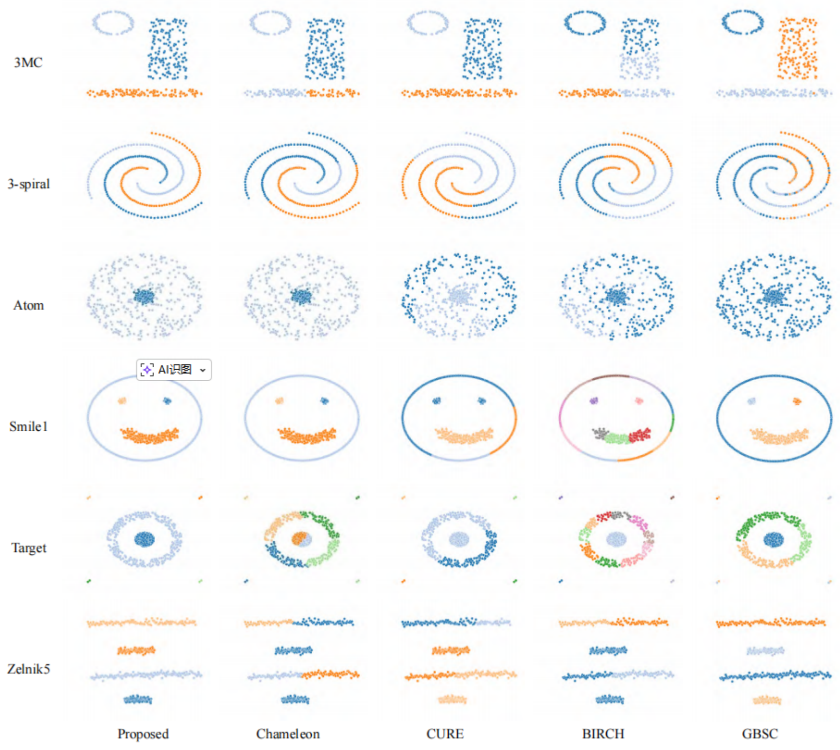

**Fig. 2.** Comparison of clustering results

**Table 5.** Result on real-world datasets

| Dataset | Index | Proposed | Chameleon | CURE | BIRCH | GBSCT |
|---|---|---|---|---|---|---|
| Breast-cancer-wisconsin | AMI | 0.669 8(3) | 0.811 4(1) | 0.524 1(4) | 0.518 2(5) | 0.731 6(2) |
| Breast-cancer-wisconsin | FMI | 0.897 3(3) | 0.949 9(1) | 0.840 3(4) | 0.829 7(5) | 0.923 9(2) |
| Dermatology | AMI | 0.656 7(1) | 0.342 2(3) | 0.259 4(4) | 0.099 0(5) | 0.600 1(2) |
| Dermatology | FMI | 0.642 4(1) | 0.352 8(3) | 0.306 3(4) | 0.230 7(5) | 0.536 4(2) |
| Haberman | AMI | 0.069 3(1) | 0.002 0(3) | 0.011 3(2) | 0.000 2(4) | -0.001 7(5) |
| Haberman | FMI | 0.761 6(1) | 0.566 8(4) | 0.758 6(2) | 0.634 6(3) | 0.550 2(5) |
| Internet-ad | AMI | 0.343 3(1) | 0.041 6(4) | 0.322 1(3) | 0.334 4(2) | 0.005 9(5) |
| Internet-ad | FMI | 0.888 4(1) | 0.856 5(4) | 0.882 0(3) | 0.885 6(2) | 0.854 2(5) |
| Iris | AMI | 0.828 7(1) | 0.803 2(2) | 0.709 3(4) | 0.701 2(5) | 0.710 1(3) |
| Iris | FMI | 0.899 9(1) | 0.840 7(2) | 0.759 3(4) | 0.751 5(5) | 0.771 2(3) |
| Newthyroid | AMI | 0.245 3(4) | 0.282 6(3) | 0.119 5(5) | 0.423 9(2) | 0.446 2(1) |
| Newthyroid | FMI | 0.759 9(3) | 0.607 5(5) | 0.742 2(4) | 0.798 6(2) | 0.809 5(1) |
| Pendigits | AMI | 0.542 8(2) | 0.581 9(1) | 0.463 9(5) | 0.526 3(3) | 0.496 6(4) |
| Pendigits | FMI | 0.456 0(2) | 0.441 0(1) | 0.382 8(5) | 0.407 1(3) | 0.394 4(4) |
| WDBC | AMI | 0.625 0(1) | 0.413 5(2) | 0.040 2(5) | 0.318 0(3) | 0.077 9(4) |
| WDBC | FMI | 0.873 2(1) | 0.717 0(2) | 0.725 0(4) | 0.739 2(3) | 0.688 9(5) |
| Wine | AMI | 0.569 7(2) | 0.413 8(3) | 0.389 4(5) | 0.409 9(4) | 0.820 1(1) |
| Wine | FMI | 0.700 1(2) | 0.576 2(5) | 0.640 9(3) | 0.582 1(4) | 0.900 4(1) |
| Zoo | AMI | 0.757 7(1) | 0.665 7(4) | 0.747 6(2) | 0.706 8(3) | 0.551 9(5) |
| Zoo | FMI | 0.718 1(1) | 0.592 9(4) | 0.718 0(2) | 0.703 4(3) | 0.507 7(5) |

## 4. Conclusion

In this paper, we introduced an enhanced hierarchical clustering algorithm that dynamically selects distance metrics and integrates robust agglomeration strategies to address the limitations of traditional methods. By leveraging nearest neighbor graphs, local density-based filtering, and representative point selection, our approach significantly improves clustering accuracy, noise resilience, and computational efficiency. Experimental results on 30 datasetsł-comprising both synthetic and real-world datałdemonstrate that the proposed method outperforms state-of-the-art algorithms such as Chameleon, CURE, BIRCH, and GBSC in terms of AMI and FMI metrics, particularly on datasets with complex shapes, varying densities, and noisy outliers.

Despite these advancements, several directions remain open for future research. First, extending the algorithm to handle high-dimensional and streaming data poses a challenge due to the curse of dimensionality and dynamic data distribution. Incorporating dimensionality reduction techniques or adaptive sampling strategies could enhance scalability. Second, while the current method dynamically selects distance metrics, a more intelligent, data-driven metric learning framework could further improve adaptability. Third, parallelization and GPU acceleration should be explored to reduce runtime for large-scale datasets. Lastly, integrating semi-supervised or weakly supervised information could guide clustering in ambiguous regions, improving performance in real-world applications such as text mining, image segmentation, and bioinformatics.

## 5. Conflict of Interest

The authors declare that there are no conflict of interests, we do not have any possible conflicts of interest.

**Acknowledgments.** None.

## References

1. Pantanowitz L, Pearce T, Abukhiran I, et al. Nongenerative artificial intelligence in medicine: advancements and applications in supervised and unsupervised machine learning[J]. Modern Pathology, 2025, 38(3): 100680.
2. Yu T, Huang W, Tang X, et al. A hybrid unsupervised machine learning model with spectral clustering and semi-supervised support vector machine for credit risk assessment[J]. PloS one, 2025, 20(1): e0316557.
3. Yu J, Li H, Yin S. New intelligent interface study based on K-means gaze tracking[J]. International Journal of Computational Science and Engineering, 2019, 18(1): 12-20.
4. Deng D. DBSCAN clustering algorithm based on density[C]//2020 7th international forum on electrical engineering and automation (IFEEA). IEEE, 2020: 949-953.

5. Wang J, Cain M, Qi M. Three-way Density Peaks Clustering Algorithm based on connectivity.[J]. Journal of Hebei University of Science & Technology, 2023, 44(6).
6. Rani P. A Survey on STING and CLIQUE Grid Based Clustering Methods[J]. International Journal of Advanced Research in Computer Science, 2017, 8(5).
7. Gogebakan M. A novel approach for Gaussian mixture model clustering based on soft computing method[J]. IEEE Access, 2021, 9: 159987-160003.
8. Ran X, Xi Y, Lu Y, et al. Comprehensive survey on hierarchical clustering algorithms and the recent developments[J]. Artificial Intelligence Review, 2023, 56(8): 8219-8264.
9. Gu Q, Niu Y, Hui Z, et al. A hierarchical clustering algorithm for addressing multi-modal multi-objective optimization problems[J]. Expert Systems with Applications, 2025, 264: 125710.
10. Guan W, Wang P, Jiang W, et al. Hc3: a three-way clustering method based on hierarchical clustering[J]. Cognitive Computation, 2025, 17(1): 8.
11. Xie W, Kang Y, Huang X, et al. MICSA: a multi-strategy integrated chameleon swarm algorithm for community detection with a new objective function[J]. Neural Computing and Applications, 2025: 1-26.
12. Srinivasan D, Kiran A, Parameswari S, et al. Energy efficient hierarchical clustering based dynamic data fusion algorithm for wireless sensor networks in smart agriculture[J]. Scientific Reports, 2025, 15(1): 7207.
13. Chen H Y, McFarlane C. Density and precarious housing: overcrowding, sensorial urbanism, and intervention in Hong Kong[J]. Housing studies, 2025, 40(1): 27-45.
14. Saki H, Zangeneh A, Aghaei J. Distributed minimum spanning tree approach for critical load restoration using microgrid formation in resilient distribution systems[J]. Electric Power Systems Research, 2025, 239: 111186.
15. Fotsa-Mbogne D J, Nguensie-Wakponou A B, Nlong J M, et al. Curvature-Based Change Detection in Road Segmentation: Ascending Hierarchical Clustering vs. K-Means[J]. Mathematics, 2025, 13(12): 1921.
16. Yu S, Engels J, Huang Y, et al. Pecann: Parallel efficient clustering with graph-based approximate nearest neighbor search[C]//2025 Proceedings of the Conference on Applied and Computational Discrete Algorithms (ACDA). Society for Industrial and Applied Mathematics, 2025: 1-17.
17. Selmi A T E, Zerarka M F, Cheriet A. Efficient technique utilizing an embedding hierarchical clustering-based representation into crossed cubes for TSP optimization[J]. Cluster Computing, 2025, 28(2): 92.
18. Feng X, Yang Y, Zhang X, et al. MCS Assisted Accurate Perception Framework for Urban POI Classification[J]. Sensors, 2025, 25(23): 7235.
19. Zhang L, Wang G, Peng L, et al. Applying pareto frontier theory and ball tree algorithms to optimize growth boundaries for sustainable mountain cities[J]. Journal of Urban Management, 2025, 14(2): 468-484.
20. Braik M, Awadallah M A, Alzoubi H, et al. Heterogeneous cognitive learning chameleon swarm algorithm for high-dimensional feature selection[J]. The Journal of Supercomputing, 2025, 81(5): 652.
21. Swefee M, Abdullah A A. Security of SDDN based on Adaptive Clustering Using Representatives (CURE) Algorithm[J]. International Journal of Computing, 2025, 17(1): 1-10.
22. Xiao J, Li Y, Tian Y, et al. Optimising allocation of marketing resources among offline channel retailers: A bi-clustering-based model[J]. Journal of Business Research, 2025, 186: 114914.
23. Swami R, Das P. A new secure data retrieval system based on ECDH and hierarchical clustering with Pearson correlation[J]. Innovations in Systems and Software Engineering, 2025, 21(1): 195-205.
24. Yaqiong L, Pan Z, Yifan L, et al. Dynamic predictive maintenance strategy for multi-component system based on LSTM and hierarchical clustering[J]. Quality and Reliability Engineering International, 2025, 41(1): 30-50.
25. Wu Y, Kang F, Wan G, et al. Automatic operational modal analysis for concrete arch dams integrating improved stabilization diagram with hybrid clustering algorithm[J]. Mechanical Systems and Signal Processing, 2025, 224: 112011.