Mathematical Biosciences and Engineering

MBE, 20(9): 15830–15858.
DOI: 10.3934/mbe.2023705
Received: 13 April 2023
Revised: 13 July 2023
Accepted: 20 July 2023
Published: 31 July 2023

http://www.aimspress.com/journal/MBE

Research article

Fast clustering algorithm based on MST of representative points

Hui Du[1], Depeng Lu[1,*], Zhihe Wang[1], Cuntao Ma[1], Xinxin Shi[1] and Xiaoli Wang[2]

[1] The School of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China
[2] School of Computer Science and Technology, Xidian University, Xi'an 710071, China

* **Correspondence:** Email: 2021222222@nwnu.edu.cn; Tel: +86 18609434184.

Abstract: Minimum spanning tree (MST)-based clustering algorithms are widely used to detect clusters with diverse densities and irregular shapes. However, most algorithms require the entire dataset to construct an MST, which leads to significant computational overhead. To alleviate this issue, our proposed algorithm R-MST utilizes representative points instead of all sample points for constructing MST. Additionally, based on the density and nearest neighbor distance, we improved the representative point selection strategy to enhance the uniform distribution of representative points in sparse areas, enabling the algorithm to perform well on datasets with varying densities. Furthermore, traditional methods for eliminating inconsistent edges generally require prior knowledge about the number of clusters, which is not always readily available in practical applications. Therefore, we propose an adaptive method that employs mutual neighbors to identify inconsistent edges and determine the optimal number of clusters automatically. The experimental results indicate that the R-MST algorithm not only improves the efficiency of clustering but also enhances its accuracy.

Keywords: minimum spanning tree; inconsistent edges; mutual neighbors; clustering; density

1. Introduction

The rapid development of big data technology has been driving research progress in fields such as biomedicine [1,2] and geography [3]. Clustering is an important tool for big data analysis, which can help researchers extract useful information from complex and massive data. The existing clustering algorithms can be broadly categorized into partitional clustering, hierarchical clustering, density-based clustering, deep clustering and so on [4]. Partitional clustering approaches optimize an objective

function by iteratively controlling the division of N data points into K clusters until the optimal solution is found or the termination condition is met, where K is much less than N [5]. While the partitional clustering algorithm performs well on datasets with spherical structures and has a linear time complexity, it is not suitable for non-convex datasets [6]. The hierarchical clustering and density-based clustering algorithms exhibit good performance when dealing with datasets that are non-convex in shape. Hierarchical clustering techniques cluster datasets through either aggregation or splitting [7]. Aggregation methods merge closely related data points in a tree structure until reaching a specific threshold of similarity between nodes [8]. Conversely, splitting methods recursively divide vast datasets based on the distinction between two groups of points according to certain thresholds [9]. Density-based noisy application space clustering (DBSCAN) is a classical clustering algorithm based on density that can cluster datasets of any shape effectively and detect noise points [10]. However, it depends on two input parameters, and its convergence time tends to be long when dealing with large dataset sizes [11]. Density Peak Clustering (DPC) is a density-based clustering algorithm that can quickly identify non-spherical clusters by estimating cluster centers based on the assumption that they are surrounded by neighboring points with low local densities and located in scattered distribution [12]. However, the quadratic time complexity of DPC is a drawback that renders it unsuitable for processing large-scale datasets. Sami and Pasi [13] proposed a fast density peaks algorithm, called FastDP, which utilizes an efficient and adaptable construction of an approximate k-nearest neighbor graph for swift density and increment computation. Utilizing this mechanism, FastDP addresses the quadratic time complexity limitation of DPC. Traditional clustering algorithms rely on geometric concepts such as distance or density. However, they are less effective in clustering non-linear high-dimensional data and identifying complex embedded features and hierarchical structures. This limitation has paved the way for the development of deep clustering algorithms [14]. Xie et al. [15] proposed deep embedded clustering (DEC) based on the automatic encoder, which is a commonly used technique in deep clustering algorithms. DEC enables joint unsupervised representation learning and clustering tasks, and it has exhibited improved clustering accuracy for large-scale high-dimensional datasets. However, the performance of this algorithm heavily relies on the quality of the learned representations via the automatic encoding process. To better extract structural and attribute information from graph data, there has been a surge of interest in researching Graph Neural Networks [16]. Wang et al. [17] proposed deep attentional embedded graph clustering (DAEGC), which uses the graph neural network to obtain the structure information of the graph data, adds the attribute information of the node in the input at the same time, fuses the node information and structure information for representation learning and uses the attention mechanism to more effectively aggregate the neighbor nodes of the node.

Minimum spanning tree (MST) is a highly popular graph structure in graph theory and is extensively employed in clustering analysis owing to its ability to detect clusters with irregular boundaries [18]. Gower and Ross [19] introduced the MST to clustering algorithms in 1969, proposing a single linkage clustering analysis achieved by pruning the MST. This method initially constructs the MST and then removes the longest k-1 edges to obtain single linkage k-partitions. In 1971, Zahn [20] formally introduced a clustering algorithm based on MST, which constructs the MST and iteratively removes inconsistent edges according to the edge weight features. Grygorash et al. [21] proposed HEMST, which removes edges from MST to achieve a reduction in standard deviation of the best possible edge weights. Müller et al. [22] proposed the ITM algorithm, which employs an entropy-based information-theoretical criterion to identify inconsistent edges, considering both cluster size and the average weight of intra-cluster edges. The Genie [23] is an example of a single linkage clustering

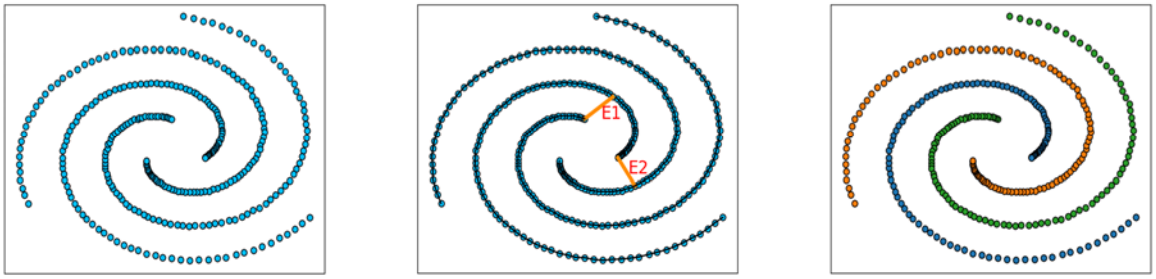
optimization variant, greedily optimizing the total edge length but only allowing the smallest clusters to merge under the constraint that the Gini index of cluster size is higher than a given threshold. The CTCEHC [24] constructs an initial partition based on vertex degree, and then merges clusters based on geodesic distance between cluster centroids. Mishra et al. [25] proposed a hybrid fast MST-based clustering method to improve efficiency. The method first divides the data into a large number of sub-clusters based on discreteness, constructs an MST for sub-cluster centroids and identifies adjacent pairs and finally merges adjacent pairs based on their internal similarity and cohesion. A clustering algorithm based on MST and the Critical Distance Method (MST-CDC) [26] uses a critical distance value as a threshold, removing inconsistent edges from the MST to obtain sub-clusters. Subsequently, it employs shorter inter-cluster distances for merging these sub-clusters. Among these MST-based clustering algorithms, there are mostly two types of problems. First, the computational cost of constructing the MST is too high, especially when dealing with large data sets. The HEMST, ITM, Genie, CTCEHC and MST-CDC all need to construct an MST using a complete graph generated from the entire data set at the initial stage of the algorithm, which is a key factor that affects the low efficiency of the algorithm. The hybrid fast MST-based clustering method use a very small number of sub-cluster centroids to construct MST, so the efficiency of this algorithm is less affected by the construction of MST. The second problem is that identifying inconsistent edges is very challenging, which is the key to the quality of clustering. In the case of the single linkage scheme, the goal is to maximize the sum of the weights of the excised inconsistent edges. This method requires a specified number of clusters, is sensitive to noise and is extremely ineffective on datasets with large differences in density distribution. The MST-based clustering algorithm proposed by Zahn considers the weights of inconsistent edges to be significantly larger than the average weights of nearby edges. This method cannot easily control the number of inconsistent edges and may result in obtaining too many or too few clusters. The HEMST removes inconsistent edges based on the standard deviation of the edge weights, and this method also requires specifying the number of clusters. The ITM utilizes information theory based on entropy to accurately identify inconsistent edges, but it relies on knowing the number of clusters. The MST-CDC uses a critical distance value as a threshold to remove inconsistent edges and obtain sub-clusters, making the algorithm more robust in the presence of outliers. However, this method may overlook density changes within a region. The Genie, CTCEHC, and the hybrid fast MST-based clustering method do not require identifying inconsistent edges. The Genie adopts an agglomerative strategy, but the limitation is that finding a suitable threshold is difficult. The partitioning strategy of CTCEHC based on MST reduces the complexity of the merging process. However, the clustering results of the method based on mixed fast MST are easily influenced by initial partitioning. It is very valuable to research how to improve the efficiency of MST-based clustering algorithms while also achieving automatic identification of inconsistent edges without the need to specify the number of clusters beforehand.

If the information of some key points in the dataset can reflect the overall structure of the dataset, then in some efficiency focused algorithms, these key points can be used to replace the entire dataset to complete the main work. This replacement idea can reduce the computational cost of the algorithm without significantly affecting the clustering quality. Motivated by this idea, we propose an algorithm that constructs the MST of representative points instead of all sample points. The proposed algorithm performs the following major steps and contributions. First, it divides the dataset into two categories, core points and noncore points. Second, it uses a novel selection strategy to pick a set of representative points from the core points. Third, it constructs an MST of representative points, and then uses adaptive

methods to identify and eliminate inconsistent edges. Finally, the algorithm first assigns each non-representative point in the core point, and then assigns the non-core point. Experimental analyses were performed on eight synthetic datasets and twelve UCI datasets. The results show that the algorithm has relatively low execution time and significantly improved clustering quality.

2. The MST-based clustering algorithm

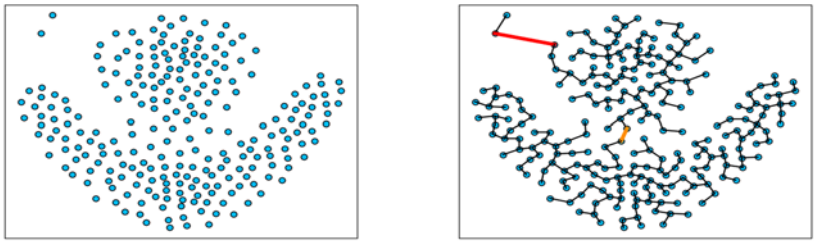
The most basic MST-based clustering algorithm consists of two steps. First, construct an MST on the complete graph of all points. Then, remove inconsistent edges from the MST to complete the clustering. Inconsistent edges are the longest edges under ideal conditions, where there are no outliers and the clusters are well separated. The MST-based clustering algorithm is usually divided into three phases: 1) constructing the MST; 2) eliminating inconsistent edges from the MST graph to create the set of connected components; 3) repeating phase 2 until the termination condition is satisfied. Figure 1 shows the main steps of the MST-based clustering algorithm. Figure 1(a) is the initial graph of the spiral dataset. Figure 1(b) constructs a minimum spanning tree using all the points in the dataset. The black and yellow lines are the edges of MST, and the two yellow lines (E1 and E2) are the two longest edges of the minimum spanning tree. After cutting off these two longest edges, three subtrees are obtained, each representing a cluster. Figure 1(c) shows the final clustering result, which is divided into three clusters.



（a） (b) (c)

Figure 1. The main steps of the MST-based clustering algorithm. (a) Spiral dataset; (b) Constructing an MST using all points; (c) Final clustering results.

However, in the presence of outliers in the dataset, the longest edge does not necessarily correspond to the inconsistent edge. Therefore, a drawback of this algorithm is its vulnerability to outliers. Figure 2 shows a simple example of the impact of outliers on the MST-based clustering algorithm. Figure 2(a) is the initial graph of the Flame dataset. Figure 2(b) constructs an MST using all the points in the dataset. The black lines, red lines and yellow lines are all edges of the MST. Among them, the red edge is the longest edge in the MST. If we cut off this longest edge, it is obvious that we will not obtain the correct clustering result. In this MST, we should remove the yellow edge to achieve the desired result, but this yellow edge is often difficult to find. To solve this problem, our algorithm first eliminates some noises and boundary point that are not conducive to finding inconsistent edges before constructing the minimum spanning tree, creating an ideal environment to alleviate the difficulty of finding inconsistent edges.

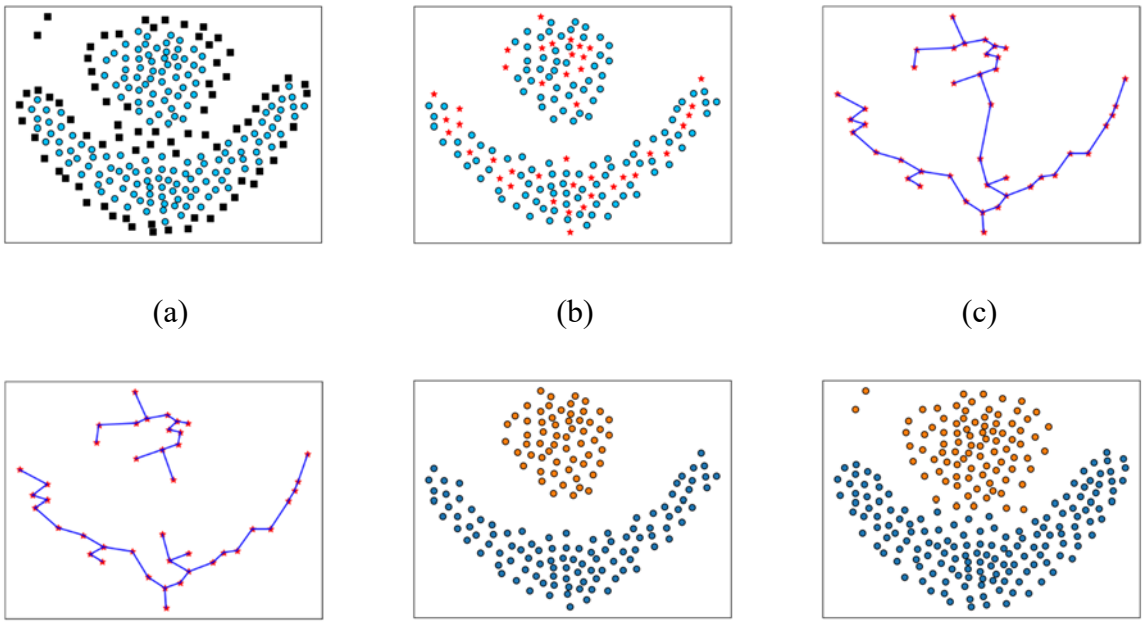


(a) (b)

Figure 2. The impact of outliers on the MST-based clustering algorithm. (a) Flame dataset; (b) Constructing an MST using all points.

For n data points, $O(n^2)$ is the total cost of constructing an MST. The key step of MST-based clustering algorithm is to construct an MST, which is the reason for the low efficiency of the MST-based clustering algorithm. To address this issue, our proposed algorithm uses selected representative points to replace all points to construct an MST, significantly reducing data size without affecting clustering quality, thereby improving the efficiency of the MST-based clustering algorithm.

3. The proposed algorithm



(a) (b) (c)

(d) (e) (f)

Figure 3. The major steps of the R-MST. (a) Dividing the dataset into core points and noncore points; (b) Selecting representative points from the core points; (c) Constructing an MST of representative points; (d) Cutting out inconsistent edges in MST; (e) Assigning nonrepresentative points from core points; (f) Assigning noncore points.

Most MST-based clustering algorithms use the entire dataset to construct the MST, which leads to high computational overhead and sensitivity to outliers. Therefore, we propose a clustering algorithm based on an MST of representative points (R-MST). A simple example is given in Figure 3. First, we find the core points, which are shown by the circular points (sky-blue points) in Figure 3(a). The square points (black points) in Figure 3(a) are noncore points. Second, we select representative points from the core points, which are shown by the star-shaped points (red points) in Figure 3(b). The circular points (sky-blue points) in Figure 3(b) are the nonrepresentative points from the core points. Third, we construct an MST of representative points, as shown in Figure 3(c). Then, the inconsistent edges in the MST are identified and cut off. Figure 3(d) shows two subtrees obtained by cutting off an inconsistent edge. Each subtree represents a cluster, and the representative points on the subtree belong to the same cluster. Finally, each nonrepresentative point from the core points is assigned to the cluster to which its representative point belongs, as shown in Figure 3(e). Each noncore point is assigned to the cluster to which the nearest core point belongs, as shown in Figure 3(f). The specific process of R-MST is illustrated in Algorithm 4.

3.1. Core points

The MST-based clustering algorithm is difficult to find the correct inconsistent edges under the interference of some noises and boundary point. Therefore, we eliminate noise and boundary point before constructing MST and create a relatively ideal environment, which makes it easier for the algorithm to find inconsistent edges. As shown in Figure 2, the two clusters in the Flame dataset are tightly connected, making it difficult to find suitable inconsistent edges. By removing the noise and boundary point, the core points of the two clusters are clearly separated, which is very conducive to the subsequent search for inconsistent edges. In our algorithm, noise and boundary point are classified as noncore points, and all points except noise and boundary point are classified as core points. The subsequent steps are mainly completed at the core point, and the noncore point can be allocated nearby at the final stage of the algorithm.

The reverse nearest neighbors of a point refer to the set of data objects in the dataset that consider this data point as one of their nearest neighbors. According to the concept of natural neighborhood, the number of reverse neighbors of noise and boundary point is relatively small, which means that there are fewer points around them. On the contrary, the core points have relatively many reverse neighbors, and the core points are closer to the points around them. Therefore, we screen the core points based on reverse nearest neighbor and critical distance.

Let $X = x_1, x_2, \ldots, x_n$ be a data set containing n samples. For each point x_i, $N_k(x_i)$ denotes the kth point closest to the x_i, $d(x_i, x_j)$ is the Euclidean distance between the point x_i and point x_j. The set of k nearest neighbors to the x_i, denoted by $KN_k(x_i)$ [27], can be further expressed as Definition 1. If point x_i is one of the neighbors of point x_m in its $KN_k(x_m)$, then x_m is a reverse neighbor of x_i. The set of reverse nearest neighbors to point x_i, denoted by $RN_k(x_i)$ [27], can be further expressed as Definition 2. The sum of the distances from x_i to the k nearest points is $DN_k(x_i)$, can be further expressed as Definition 3. $LRN_k(x_i)$ denotes the number of the reverse nearest neighbors of point x_i, can be further expressed as Definition 4. LRN_k_med denotes the median of the number of the reverse nearest neighbors of all points, can be further expressed as Definition 5.

Definition 1: (*k* nearest neighbors)

$$KN_k(x_i)=\left\{x_j \in X \mid d(x_i,x_j) \le d(x_i,N_k(x_i))\right\} \tag{1}$$

Definition 2: (Reverse nearest neighbors)

$$RN_k(x_i)=\left\{x_m \in X \mid x_i \in KN_k(x_m)\right\} \tag{2}$$

Definition 3: (The sum of the distances from x_i to the *k* nearest points)

$$DN_k(x_i)=\sum_{j=1}^{k} d(x_i,N_j(x_i)) \tag{3}$$

Definition 4: (The number of the reverse nearest neighbors of point x_i)

$$LRN_k(x_i)=\mid RN_k(x_i)\mid \tag{4}$$

Definition 5: (The median of the number of the reverse nearest neighbors of all points)

$$LRN_k\_med = Median\left(\left\{LRN_k(x_i)\mid x_i \in X\right\}\right) \tag{5}$$

Definition 6: (Core point) For any point $x_i \in D$, $D = x_1, x_2, \ldots, x_n$ denotes the initial dataset comprising *n* samples. If a point satisfies Condition 1 or Condition 2, it is considered a core point. Additionally, if a point satisfies neither Condition 1 nor Condition 2, it is considered as a noncore point. Condition 1 is to discover the core point from the perspective of reverse nearest neighbors, which is specifically represented as the number of the reverse nearest neighbors of point x_i is not less than the median of the reverse nearest neighbors of all points. Condition 2 is to discover the core point from the perspective of distance, which is specifically represented as the $DN_{k_1}(x_i)$ is not greater than the average of the sum of the distances of all points to their nearest k_1 neighbors. The parameter k_1 refers to the value of the number of nearest neighbors in the initial dataset (D).

Condition 1:

$$LRN_{k_1}(x_i) \ge LRN_{k_1}\_med \tag{6}$$

Condition 2:

$$ND_{k_1}(x_i) \le \sum_{j=1}^{n} ND_{k_1}(x_j)/n \tag{7}$$

Let $C = \{c_1, c_2, \ldots, c_m\}$ denote the set of core points, where *m* denotes the number of core points. The algorithm for dividing the dataset into core points and noncore points is shown in Algorithm 1.

Algorithm 1: Dividing the dataset into core points and noncore points

Input: Dataset D=$\{x_1, x_2, x_3, \ldots x_n\}, k_1$

Output: The set of core points $C = \{c_1, c_2, c_3, \ldots c_m\}$, the set of noncore points $O = \{o_1, o_2, o_3, \ldots o_{n-m}\}$

/* Obtain k_1 nearest neighbors for each point in the initial dataset. */

For each x_i in D **do**

Calculate $KN_{k_1}(x_i)$

End for

/* Obtain reverse neighbors for each point in the initial dataset. */

For each x_i in D **do**

Calculate $RN_{k_1}(x_i)$

End for

Create $C = \emptyset$, $O = \emptyset$

/*Determine whether each point is a core point or a noncore point sequentially. */

For each x_i in D **do**

If x_i meets Condition 1 or Condition 2 **then**

$C \leftarrow C \cup x_i$

Else

$O \leftarrow O \cup x_i$

End if

End for

3.2. Representative points

Representative points can be seen as an important subset of a dataset, as they often have higher information value or representativeness, allowing for a reduction in computational and storage costs without sacrificing accuracy. When selecting representative points, it is important to consider factors such as the distribution, data density and distance measurement of the dataset to ensure an effective representation of its diversity. In addition, the number of representative points should be significantly smaller than the size of the original dataset. Inspired by the concept of representative points, Chowdhury et al. treat each representative point found as a subcluster, and they then complete the clustering by merging the subclusters [27]. The method they proposed for selecting representative points is to find the point with the highest density within a neighborhood of a point as its representative point. This method is effective for datasets with relatively uniform density distributions. However, in datasets with varying densities, it may lead to the scarcity and dispersion of representative points within clusters composed of low-density points. Therefore, we propose an improved strategy for selecting representative points, aiming to achieve a more uniform distribution of representative points within clusters composed of low-density points.

Our improved selection strategy for representative points considers both density and nearest neighbor distance. Based on the strategy of selecting the highest density points in the neighborhood as representative points, an additional criterion is added to allow some points with low density and a large average distance between them and their surrounding neighbors to select themselves as representative points. If any point chooses the point with the highest density within its nearest neighbor range as the representative point, it may result in abnormally large distances between adjacent representative points.

For example, the adjacent representative points v3 and v4 in Figure 5(d). If point v0 between points v3 and v4 is chosen as the representative point, this problem can be avoided, as shown in Figure 5(b). The characteristics of key points like v0 are low density and large average distance from surrounding neighbors. They play a very important role in sparse areas. If these points are lost, it will damage the overall structure of the sparse area, causing abnormal long edges to be generated inside the sparse cluster during MST construction, which may result in incorrect clustering results when a sparse cluster is divided into multiple clusters. Therefore, selecting such points as representative points will greatly improve the distribution uniformity of representative points in sparse areas. In addition, our improved strategy has almost no impact on the selection of representative points for dense regions, because the point density in this region is generally high, so it does not meet our additional criteria. The flow diagram illustrating our improved method for selecting representative points is shown in Figure 4. Algorithm 2 shows the details of selecting representative points.

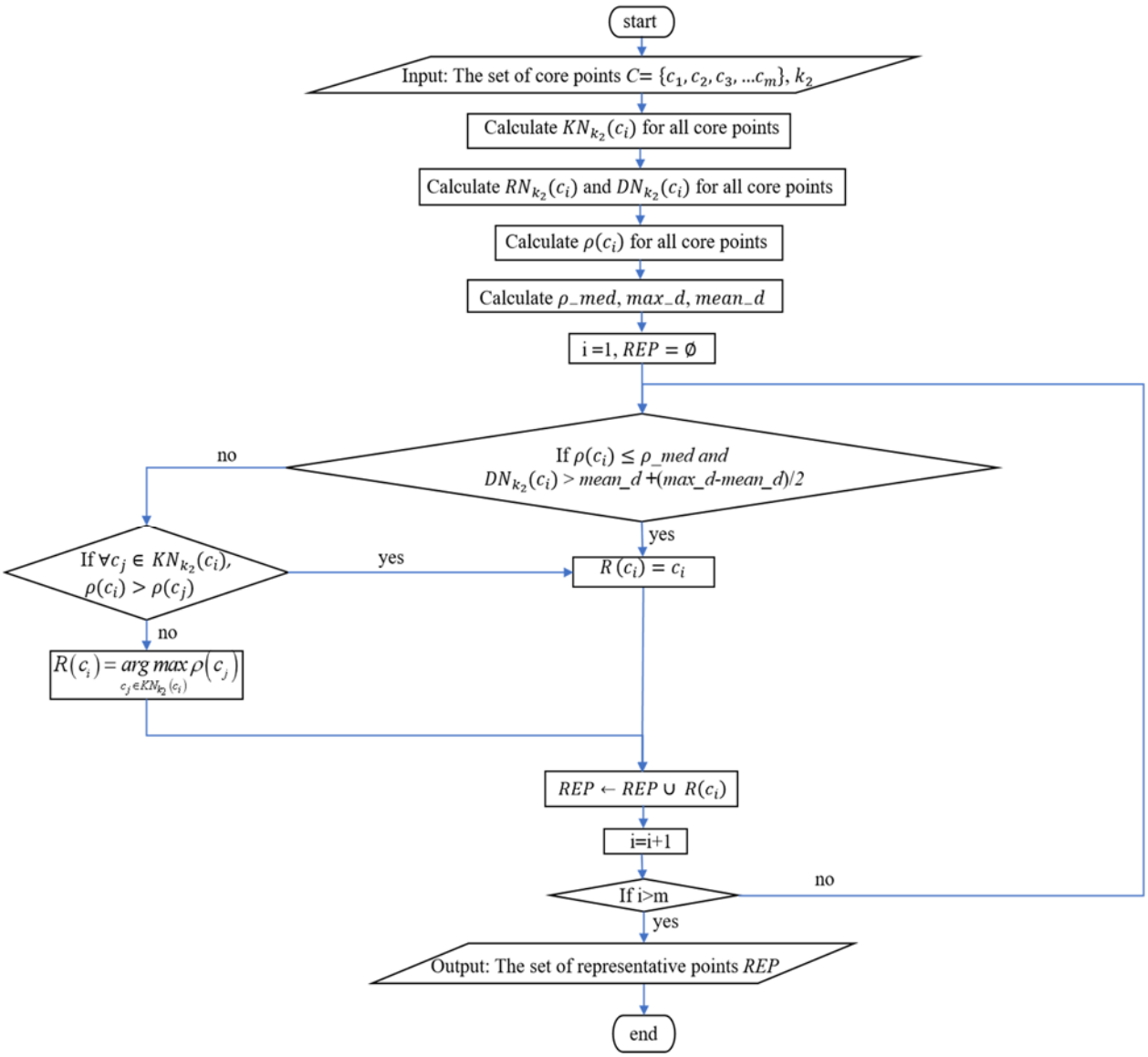


Figure 4. Flow diagram for selecting representative points.

Algorithm 2: Selecting representative points

Input: The set of core points $C = \{c_1, c_2, c_3, \ldots c_m\}$, k_2

Output: The set of representative points $DREP = \{drep_1, drep_2, \ldots, drep_{nrep}\}$, the set of nonrepresentative points $NDREP = \{ndrep_1, ndrep_2, \ldots, ndrep_{m-nrep}\}$

/* Obtain the k_2 nearest neighbors for each core point in the set *C*. */

For c_i in *C* **do**

Calculate $KN_{k_2}(c_i)$

End for

/* Obtain the reverse neighbors for each core point in the set *C*. */

/* Obtain the sum of distances between each core point and its k_2 nearest neighbors within the set *C*. */

For c_i in *C* **do**

Calculate $RN_{k_2}(c_i)$

Calculate $DN_{k_2}(c_i)$

End for

/* Obtain the density of each core point*/

For c_i in *C* **do**

Calculate $\rho(c_i)$

End for

Calculate ρ_med, max_d, $mean\_d$

/* The set *REP* stores the representative points corresponding to each core point. */

Create a set $REP = \emptyset$

For c_i in *C* **do**

If c_i meets Condition 3 **then** /*Scenario (1) for selecting representative points. */

$R(c_i) = c_i$

Else if c_i meets Condition 4 **then** /*Scenario (2) for selecting representative points. */

$R(c_i) = c_i$

Else /*Scenario (3) for selecting representative points. */

Calculate $R(c_i)$ according to Eq (9)

End if

End if

/*Add the representative point corresponding to core point to the *REP*. */

$REP \leftarrow REP \cup R(c_i)$

End for

Remove duplicate elements from *REP* to obtain the final representative point set *DREP*.

Obtain the set of nonrepresentative points *NDREP* according to Eq (16).

We specify the representative points for each core point in three different scenarios. Scenario (1): if the density of a point is low and it is far from its neighbors, its representative point is also itself. Scenario (2): if a point does not satisfy scenario (1), but its density is higher than the density of any point within its k_2 nearest neighbor range, its representative point remains itself. Scenario (3): if a point does not satisfy scenarios (1) and (2), its representative point is chosen as the neighbor within the k_2 nearest neighbor range with the highest density. The density is represented by Definition 7. The representative point is described as shown in Definition 8. Algorithm 2 shows the details of

selecting a representative point. For any point $c_i \in C$, $C = c_1, c_2, ..., c_m$ denotes the core point set comprising *m* core points. The parameter k_2 refers to the value of the number of nearest neighbors in the core point set (C).

Definition 7: (Density) The traditional radius-based density estimation method is difficult to accurately evaluate the density level of each data point. Therefore, in order to better adapt to the dataset with different densities, the nearest neighbor characteristics of adjacent points around each point are comprehensively considered when evaluating the density of each point. If the number of the reverse nearest neighbors of the point is greater than 0, the density is the number of the reverse nearest neighbors of this point plus the number of the reverse nearest neighbors for each of its reverse nearest neighbors. If the number of the reverse nearest neighbors of the point is 0, the density of the point is 0. $\rho(c_i)$ represents the density of point c_i.

$$\rho(c_i) = \begin{cases} LRN_{k_2}(c_i) + \sum_{c_j \in RN_{k_2}(c_i)} LRN_{k_2}(c_j) & , LRN_{k_2}(c_i) > 0 \\ 0 & , LRN_{k_2}(c_i) = 0 \end{cases} \tag{8}$$

Definition 8: (Representative points) $\forall\ c_i \in C$, $R(c_i)$ represents the representative point of c_i. If c_i satisfies Condition 3, then $R(c_i) = c_i$. If c_i does not satisfy Condition 3, but satisfies Condition 4, then $R(c_i) = c_i$. If c_i does not satisfy Condition 3 and does not satisfy Condition 4, then

$$R(c_i) = \underset{c_j \in KN_{k_2}(c_i)}{arg\,max}\, \rho(c_j) \tag{9}$$

Condition 3:

$$\rho(c_i) \le \rho\_med \tag{10}$$

$$\text{And}\quad DN_{k_2}(c_i) > \left(mean\_d + \frac{max\_d - mean\_d}{2} \right) \tag{11}$$

Condition 4:

$$\forall c_j \in KN_{k_2}(c_i), \rho(c_i) > \rho(c_j) \tag{12}$$

where the constant 2 in Eq (11) was obtained based on a large number of experiments. ρ_med is the median of the density of all points in the core point set *C*. max_d is the maximum value of the sum of distances between a point in the core point set *C* and the points within its k_2 nearest neighbor range. $mean\_d$ is the average of the sum of distances between all core points and points within their k_2 nearest neighbor range.

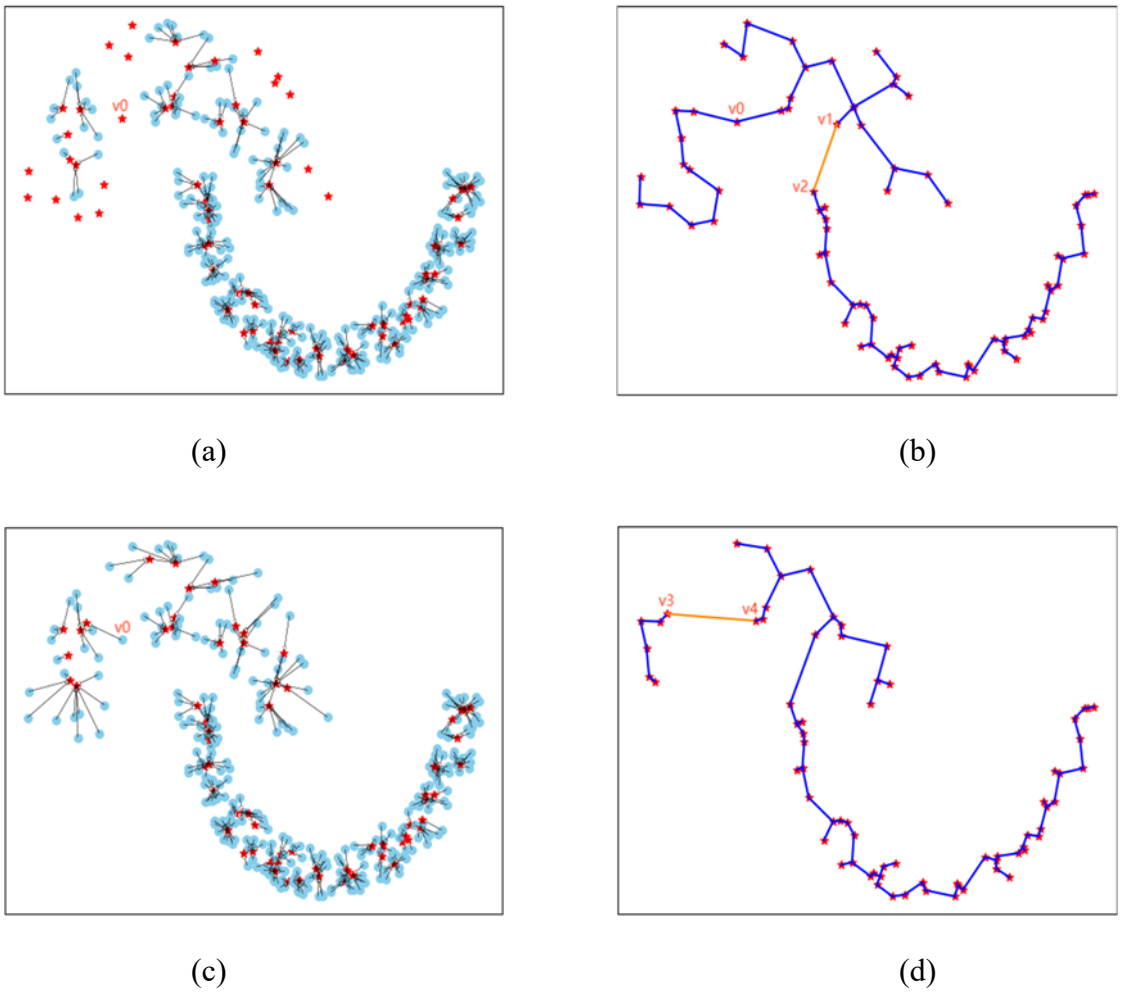
$$\rho\_med = Median\left(\{ \rho(c_i) \mid c_i \in C \} \right) \tag{13}$$

$$max\_d = \underset{c_j \in C}{max}\, DN_{k_2}(c_j) \tag{14}$$

$$mean\_d = \sum_{i=1}^{m} DN_{k_2}(c_i) / m \tag{15}$$

Get a set of representative points $REP = \{R(c_1), R(c_2), \ldots, R(c_m)\}$ until the end of the election. Some points with higher density may serve as representative points for multiple points, as shown in Figure 5. Therefore, the number of representative points is much smaller than the number of sample points. By removing redundant duplicate elements from the REP, we can obtain the final set of representative points, denoted as $DREP = \{drep_1, drep_2, \ldots, drep_{nrep}\}$, where $nrep$ is the number of representative points. In the set C, all nonrepresentative points are considered as the set $NDREP$, which is denoted as:

$$NDREP = \{c_i \in C \wedge c_i \notin DREP\} \tag{16}$$



(a) (b)

(c) (d)

Figure 5. The selection result of the representative point. (a) The selection results of our improved representative point selection method; (b) Construct an MST using the representative points from Figure 5(a); (c) The selection results of the representative point method based solely on density. (d) Construct an MST using the representative points from Figure 5(c).

In Figure 5(a),(c), the black lines with arrows point to the representative points of each point. If a point is not pointing to any other point, it means that it has chosen itself as the representative point. All selected representative points are denoted by red star-shaped points, while nonrepresentative points are represented by blue circular points. In Figure 5(b), the yellow edge connecting vertex v1 and vertex v2 is the longest edge on MST, and in Figure 5(d), the yellow edge connecting vertex v3 and vertex v4 is the longest edge on MST.

3.3. Constructing an MST of representative points and identifying inconsistent edges

We use the selected representative points to construct the MST. The MST of representative points is described by Definition 9. The specific algorithm for constructing MST adopts Prim [28] algorithm, which is an algorithm based on greedy strategy to obtain the MST in a weighted connected graph. The basic flow of the algorithm includes the following steps: randomly select a node from the graph as the starting point, add it to the minimum spanning tree and then add the node connected with the edge with the minimum weight among the adjacent nodes to the minimum spanning tree, and repeat this process until the minimum spanning tree contains all nodes.

Definition 9: (The MST of representative points) Assume that $G(X) =< V, E >$ denotes a weighted undirected connected graph with edge set $E = \{e_{ij} = \{x_i, x_j\} | x_i, x_j \in DREP, i \neq j\}$, point set $V = DREP$, the weight of each edge e_{ij} in graph $G(X)$ is denoted as $w(x_i, x_j)$. The MST of the graph $G(X)$ is denoted by $W(MST) = min\{\sum_{x_i, x_j \in V, e_{ij} \in E} w(x_i, x_j)\}$, defined as a subset of E, connecting all vertices in V with minimum total weight and without cycles.

In most MST-based clustering algorithms, it is often necessary to provide the number of clusters or a distance threshold when cutting inconsistent edges. However, in reality, we may not know the number of clusters in advance and find it difficult to find a suitable distance threshold. Therefore, the current challenge is how to adaptively identify inconsistent edges. As we mentioned earlier, before building the minimum spanning tree, after removing the noise and some boundary point, the core points of two adjacent clusters are obviously separated, which creates favorable conditions for identifying inconsistent edges. At this point, we can easily achieve adaptive recognition of inconsistent edges using the nearest neighbor method, without the need to provide the number of clusters and distance thresholds. The main steps of our proposed adaptive recognition of inconsistent edges include: 1) Arranging all edges in MST in descending order of edge length. 2) Starting from the longest edge, determine whether the condition for inconsistent edges is met. 3) Determine the remaining edges in the order sorted by their length until the first time an edge does not meet the condition for inconsistent edges, and then end the search for inconsistent edges. After the search for inconsistent edges ends, the number of clusters will be automatically obtained. We set the initial cluster number to 1, and each time an inconsistent edge is cut, the number of clusters will increase by 1. Therefore, the number of clusters equals to the number of inconsistent edges plus 1. The specific algorithm process is shown in Algorithm 3.

Definition 10: (mutual *k* nearest neighbor)

$\exists\ point\ c_i, point\ c_j, if\ c_i \in KN_k(c_j) \wedge c_j \in KN_k(c_i)$, then c_i and c_j are mutual k nearest neighbors.

The condition for determining inconsistent edges: As shown in Figure 6, assuming that two vertices connecting an edge are p_1 and p_2, p_1 and its affiliated points (a point whose representative

point is p_1 is called an affiliated point of p_1, such as p_3, p_4, p_7, p_8 in Figure 6) form the set $S1$, and p_2 and its affiliated points (a point whose representative point is p_2 is called an affiliated point of p_2, such as p_5, p_6, p_9 in Figure 6) form the set $S2$. If any point in the set $S1$ has no mutual k_2 neighbor relationship with any point in set $S2$, then the edge is inconsistent. As long as there is a point in set $S1$ and any point in set $S2$ that have a mutual k_2 neighbor relationship, then that edge is not an inconsistent edge. Mutual *k* nearest neighbor [29] is represented by definition 10.

Algorithm 3: Identifying inconsistent edges

Input: An MST of representative points, k_2

Output: The set of inconsistent edges S, the number of clusters nc

/*E denotes the set of edges of the MST, $e(drep_{i,}\, drep_j) \in E$ and $w(drep_{i,}\, drep_j)$ is the weight corresponding to $e(drep_{i,}\, drep_j)$*/

Sort all MST edges in descending order by weight size to get a weight list w_{sorted}

Create $S = \emptyset,\ S1 = \emptyset,\ S2 = \emptyset,\ nc = 1$

For $w(drep_{i,}\, drep_j)$ in w_{sorted} **do**

/*The edge $e(drep_{i,}\, drep_j)$ associated with weight $w(drep_{i,}\, drep_j)$, connects two representative points $drep_i$ and $drep_j$, which are subsequently added to the empty sets *S1* and *S2*, respectively. */

$S1 \leftarrow S1 \cup\ drep_i$

$S2 \leftarrow S2 \cup\ drep_j$

Add all the points in the core point set whose representative point is $drep_i$ to *S1*.

Add all the points in the core point set whose representative point is $drep_j$ to *S2*.

/*When an edge is not found to be inconsistent for the first time, the iteration is halted, marking the conclusion of the process of identifying inconsistent edges. */

If exists $c_i \in S1,\ c_j \in S2,\ c_i \in KN_{k_2}(c_j) \wedge c_j \in KN_{k_2}(c_i)$ **then**

break

Else

/*The identified inconsistent edges are added to the set *S* of inconsistent edges. */

$S \leftarrow S \cup\ e(drep_{i,}\, drep_j)$

/**S1* and *S2* are emptied to be used for the next iteration. */

$S1 = \emptyset$

$S2 = \emptyset$

/*Each time an inconsistent edge is detected, the number of clusters increases by 1. */

$nc = nc + 1$

End if

End for

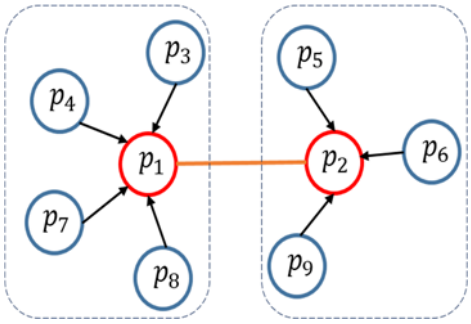


Figure 6. Identify inconsistent edges.

3.4. Assigning nonrepresentative points and noncore points

Assuming that the number of inconsistent edges is m, then $m + 1$ subtrees are obtained by cutting these inconsistent edges from the MST. Each subtree represents a cluster, so the number of clusters is also $m + 1$. Representative points on the same subtree belong to a cluster, and representative points on different subtrees are not in a cluster. The next step is to assign each nonrepresentative point to the cluster to which their representative points belong. If point x_i is a nonrepresentative point, its representative point is x_j, that is, $R(x_i) = x_j$. If x_j belongs to cluster c, then x_i will join c. The last step is to assign noncore points (noise and some boundary points). For the processing of such points, we adopt the measure of assigning them to the cluster where the nearest core point belongs. If point x_i is a noncore point, the closest core point to it is x_j. If x_j belongs to cluster c, then x_i will join c. After introducing all the steps, we provide Algorithm 4 to describe the entire process of R-MST.

Algorithm 4: R-MST: fast clustering algorithm based on MST of representative points

Input: Dataset D=$\{x_1, x_2, x_3, \ldots x_n\}$, k_1, k_2

Output: Clustering results of dataset D

Step1: According to Algorithm 1, the points in the dataset D are divided into core points and noncore points.

/* At the end of step 1, the set of core points $C = \{c_1, c_2, c_3, \ldots c_m\}$ and the set of noncore points $O = \{o_1, o_2, o_3, \ldots o_{n-m}\}$ are obtained, where m is the number of core points and n-m is the number of noncore points. */

Step2: According to Algorithm 2, the representative points for each core point are selected.

/*At the end of step 2, the set of representative points $DREP = \{drep_1, drep_2, \ldots, drep_{nrep}\}$ and the set of nonrepresentative points $NDREP = \{ndrep_1, ndrep_2, \ldots, ndrep_{m-nrep}\}$ are obtained, where *nrep* is the number of representative points and *m-nrep* is the number of nonrepresentative points. */

Step3: Constructing an MST using the Prime algorithm on the complete graph generated by the representative points.

Step4: Identifying inconsistent edges in MST according to Algorithm 3, and then remove all inconsistent edges.

/* At the end of step 4, the clustering results of the representative points are obtained. */

Step5: Assigning each nonrepresentative point.

/* At the end of step 5, the clustering results of the core points are obtained. */

Step6: Assigning each noncore point.

/* At the end of step 6, the clustering results of dataset D are obtained. */

3.5. Time complexity analysis

For a dataset containing n sample points, the time overhead of the R-MST to complete the clustering is mainly in the following aspects: 1) The time complexity of finding core points is $0(n)$. 2) The time complexity of selecting representative points is $0(m)$, where m is the number of core points and m is less than n. 3) The time complexity of constructing an MST of representative points is $0(nrep^2)$, where $nrep$ is the number of representative points and $nrep$ is much smaller than n. 4) The time complexity of identifying inconsistent edges is $0(nrep)$. 5) The time complexity of assigning nonrepresentative points is $0(m)$. 6) The time complexity of assigning noncore points is less than $0(n)$. In summary, the time complexity of the R-MST is approximated as $0(nrep^2)$.

Based on the experimental analysis that follows, in an ideal state, the number of representative points is approximately 1/20 of the entire dataset. That is, $nrep: n \approx 1: 20$, so $0(nrep^2) \approx 0(n^2)/400$. Although the R-MST is relatively efficient, the number of representative points also increases proportionally as the dataset grows, which limits the application of the algorithm on extremely large-scale datasets. To alleviate this constraint, we will investigate in our future work how to reduce the dependency of the number of representative points on dataset size, such that the number of representative points can be kept very low even for very large datasets. Additionally, it is crucial to avoid the quadratic time complexity of constructing minimum spanning trees, and this future work will be discussed in detail in the conclusion.

4. Experimental result and analysis

4.1. Experiment preparation

For the experiment, we tested the R-MST on synthetic datasets and UCI datasets. The comparison algorithms included DPC [12], FastDP [13], DBSCAN [10] and MST-CDC [26]. These four comparison algorithms are all very distinctive and can be compared with our proposed algorithm (R-MST) in a comprehensive manner from different perspectives. The advantage of the DPC is its ability to handle non-spherical, complexly distributed data sets and does not require an artificially set number of clusters. The FastDP is an optimization method based on the DPC and has the advantage of being able to handle large data sets quickly and efficiently. The advantage of DBSCAN clustering algorithm is that it can automatically handle clusters of arbitrary shape and size, and can efficiently handle noisy data points. MST-CDC can also identify inconsistent edges on data sets containing noisy points to obtain optimal clusters.

It is difficult to comprehensively assess the merits of clustering results with a single clustering metric. Different clustering metrics focusing on different aspects can help us understand the clustering results from different perspectives and help to better evaluate the effectiveness of clustering algorithms. Therefore, we used three metrics. These evaluation metrics included the adjusted rand index (ARI) [30], normalized mutual information (NMI) [31] and homogeneity (Homo) [32]. ARI is a metric used to compare the similarity between the clustering algorithm results and the true clustering labels. When comparing the clustering algorithm results, the ARI metric takes into account the different arrangements of clustering labels, and thus can reflect the similarity between clustering results more accurately. Furthermore, the ARI metric also considers the metric error due to random chance, which improves the reliability of the comparison results. The NMI clustering metric calculates a score by

measuring the similarity between two clustering results. It uses normalized mutual information from information theory, which treats clustering results as random variables, and is used to represent the mutual information between different clustering results on the same data set. Usually, higher NMI values indicate higher quality of clustering results. The measurement goal of Homo is the score when each cluster contains only a single sample category. This indicator calculates the proportion that all samples in each real category belong to the same cluster, and averages the values of all real categories. Simply put, the higher the Homo, the higher the probability that each cluster represents the clustering result only contains one category, and the more accurate and feasible the clustering result is.

Table 1. Synthetic datasets.

Dataset	Number of instances	Dimension	Number of categories
ED-Hexagon	361	2	2
Jain	373	2	2
Three-circles	299	2	3
Heart-shaped	213	2	3
Ls3	1735	2	6
D31	3100	2	31
2d-20c-no0	1517	2	20
T7	8000	2	9

Table 2. UCI datasets.

Dataset	Number of instances	Dimension	Number of categories
Zoo	101	16	7
Cancer	683	9	2
Seeds	210	7	3
WBC	683	9	2
Wine	178	13	3
Ecoli	336	8	8
Iris	150	4	3
Vote	435	16	2
Vowel	871	3	6
WDBC	569	30	2
Dermatology	358	34	6
Pendigits	3498	16	10

We used datasets of different sizes and dimensions to examine the performances of our algorithms. The eight synthetic datasets contain ED-Hexagon [33], Jain [34], Three-circles [34], Heart-shaped [33], Ls3 [34], D31 [34], 2d-20c-no0 [34] and T7 [34]. The details of these synthetic datasets are shown in Table 1. The twelve UCI datasets [35] contain Zoo, Cancer, Seeds, WBC, Wine, Ecoli, Vote, Vowel, WDBC, Dermatology and Pendigits. The details of these UCI datasets are shown in Table 2. In addition, in the efficiency test, we generated moons datasets with 2000, 4000, 6000, 8000, 10,000, 12,000

sample points using the program. In discussing the effect of the parameter k_2 on the number of representative points and the running time of the algorithm, we generated a dataset with 10,000 sample points for testing.

The experiments were conducted on a PC with an Intel Core i5 3.6 GHZ with 4 GB RAM, Windows 10 and Python 3.7.

4.2. Experimental results on synthetic datasets

In this paper, we have chosen eight synthetic datasets to validate the clustering quality of R-MST and four other algorithms. These eight datasets cover different types of datasets, including varying density, rich noise interference, as well as datasets with diverse linearity, circularity and sphericity. Such datasets can effectively simulate complex distribution scenarios in real-world settings, facilitating the validation of algorithm generalizability. Additionally, in the presence of significant noise, the robustness of the algorithm can be tested as well. The key information for these datasets is shown in Table 1. The optimal parameters of the five algorithms for the eight synthetic datasets are shown in Table 3.

Figure 7 shows the clustering result of ED-Hexagon. The ED-Hexagon dataset contains a convex cluster and a non-convex cluster. The density distribution of this dataset is relatively uniform. Both the R-MST and DBSCAN identified the appropriate clusters, while the DPC, FastDP and MST-CDC produced incorrect clustering results.

Figure 8 shows the clustering results of Jain. Jain is composed of two clusters with large differences in density distribution. The R-MST was able to correctly identify the clusters on this dataset. The other four algorithms produced incorrect results.

Figure 9 shows the clustering results of Three-circles. Three-circles consist of two rings and one solid circle. The R-MST and DBSCAN produced acceptable clustering results. The DPC, FastDP and MST-CDC produced incorrect results. The concentration of multiple high-density points in a cluster can easily lead to the biased selection of cluster centers, which, in turn, can lead to the incorrect distribution of the noncentral points. MST-CDC cuts off too many inconsistent edges, which results in the generation of multiple subtrees by the outermost circle points, and the merging process does not merge all the outermost circle subtrees.

Figure 10 shows the clustering results of Heart-shaped. Heart-shaped consists of three heart shapes with a large difference in density distribution. The R-MST, DPC, FastDP and DBSCAN produced satisfactory clustering results on this dataset. The MST-CDC produced incorrect results because it identified two normal points as noise.

Figure 11 shows the clustering results of Ls3. The Ls3 dataset contains four spherical clusters and two linear clusters. Except for the DPC and FastDP, the other three algorithms obtained correct results. Because the DPC could not choose the clustering centers reasonably, it produced incorrect results.

Figure 12 shows the clustering results of D31. The R-MST, DPC and FastDP produced satisfactory clustering results on this dataset. DBSCAN identifies the noise, and the clustering results are relatively good. The MST-CDC causes unsatisfactory results due to excessive noise interference.

Figure 13 shows the clustering results of 2d-20c-no0. 2d-20c-no0 is composed of a number of linear clusters and blocky clusters. The DPC, FastDP and R-MST all obtained correct clustering results. DBSCAN identified one of the clusters as noise and identified some close clusters as one cluster, but the overall clustering results were relatively good. The MST-CDC produced incorrect clustering results.

Figure 14 shows the clustering results of T7. T7 consists of 9 clusters of different shapes and some noise. DBSCAN and R-MST achieved relatively good clustering results, while the other three algorithms showed unsatisfactory clustering results. The clustering centers of DPC and FastDP were selected incorrectly. MST-CDC identified the noise, but the clustering results were poor.

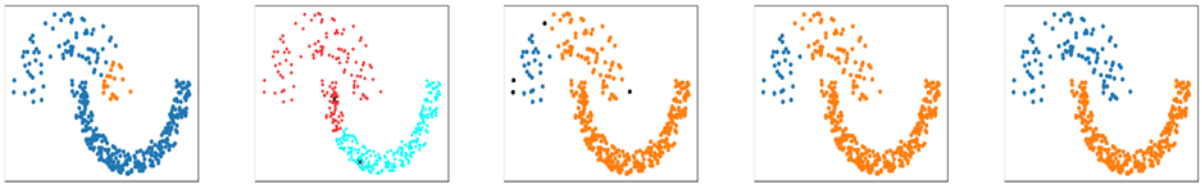
The performance of R-MST in clustering is demonstrated as excellent across these 8 different types of datasets, exhibiting the capability to detect and aggregate clusters of diverse shapes and densities. Additionally, R-MST shows remarkable results in complex datasets like T7, which contain a considerable amount of noise, thereby highlighting the strong robustness of the algorithm.

Table 3. Optimal parameters of 5 algorithms on 8 synthetic datasets.

Algorithm	DPC	FastDP	DBSCAN	MST-CDC	R-MST
Parameters	dc	k	Eps/MinPts	None	k_1/k_2
ED-Hexagon	15	18	20/4	---	2/6
Jain	4.65	18	3.1/8	---	2/9
Threecircles	0.08	18	0.06/4	---	1/6
Heart-shaped	18	18	20/4	---	9/6
Ls3	10.28	18	10/8	---	1/7
D31	1.27	18	0.5/6	---	15/5
2d-20c-no0	1.19	18	1.2/20	---	10/20
T7	29.17	18	10/12	---	80/9

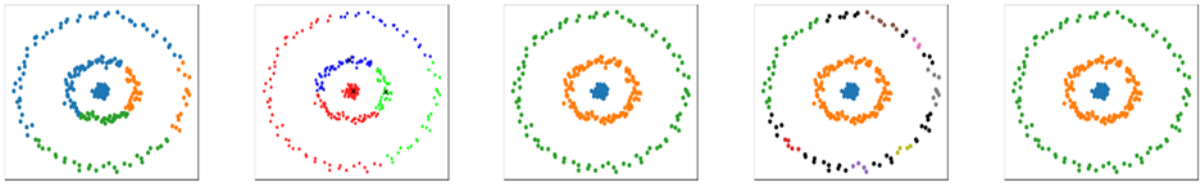
(a) (b) (c) (d) (e)

Figure 7. The result of ED-Hexagon. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.



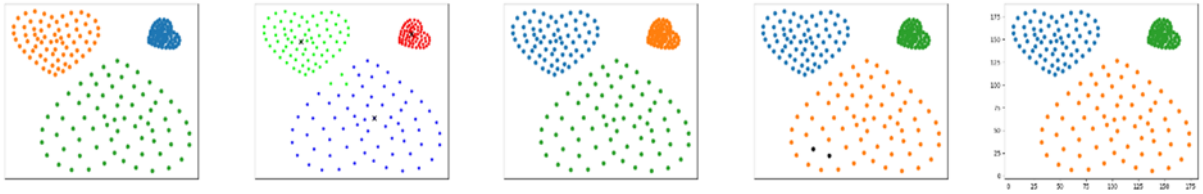
(a) (b) (c) (d) (e)

Figure 8. The result of Jain. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.



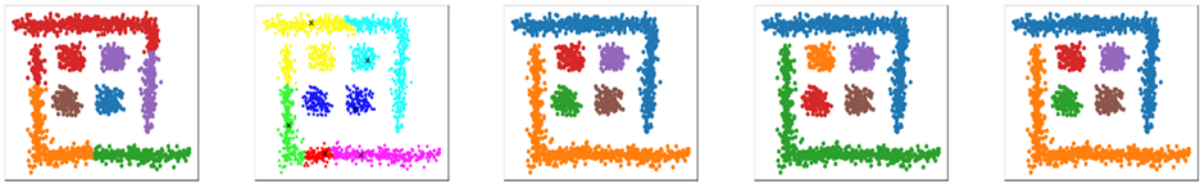
(a) (b) (c) (d) (e)

Figure 9. The result of Three-circles. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.



(a) (b) (c) (d) (e)

Figure 10. The result of Heart-shaped. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.

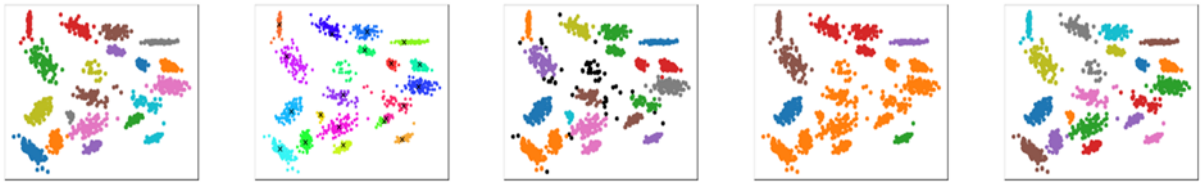


(a) (b) (c) (d) (e)

Figure 11. The result of Ls3. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.

(a) (b) (c) (d) (e)

Figure 12. The result of D31. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.



(a) (b) (c) (d) (e)

Figure 13. The result of 2d-20c-no0. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.

(a) (b) (c) (d) (e)

Figure 14. The result of T7. (a) DPC; (b) FastDP; (c) DBSCAN; (d) MST-CDC; (e) R-MST.

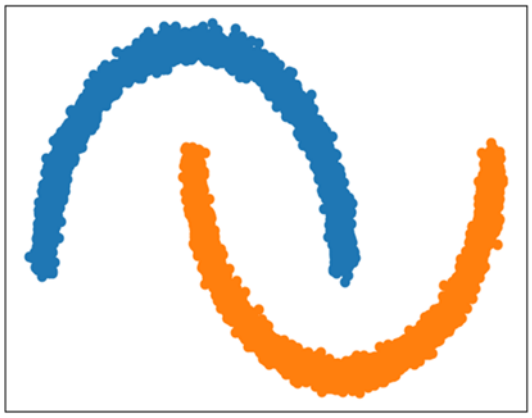


Figure 15. The moons dataset.

It is necessary to test the efficiency of clustering algorithms, as the efficiency of the algorithm directly affects its usability and practicality. In the efficiency test, we tested the runtime of R-MST and four comparative algorithms on datasets with varying numbers of data points. To ensure fairness, we generated moons datasets with 2000, 4000, 6000, 8000, 10,000 and 12,000 sample points. The five algorithms achieved correct clustering results on different numbers of moons datasets. Figure 15 shows the moons dataset. Table 4 shows the running times of the algorithms on the different numbers of moons datasets. From Table 4, we can see that the fastest running algorithm is FastDP, because this algorithm eliminates the quadratic time complexity limitation of DPC. The second fastest algorithm is our proposed R-MST, which uses representative points instead of all points to construct the MST, reducing the computational overhead to a certain extent. The time efficiency of the R-MST is not as fast as FastDP, but the clustering quality is better than FastDP. The third fastest algorithm is DPC. The

relatively slower algorithm is DBSCAN. The slowest algorithm is MST-CDC. As the number of sample points gradually increases, the running time of all five algorithms increases to different degrees. However, FastDP and R-MST change relatively slowly, while the other three algorithms change more rapidly. In summary, R-MST has high efficiency and is capable of handling large datasets.

Table 4. Running time of 5 algorithms on moons datasets (s).

Algorithm	2000	4000	6000	8000	10,000	12,000
DPC	9	36	84	154	312	603
FastDP	0.1	0.4	0.7	0.9	1.3	1.6
DBSCAN	27	112	249	442	902	1872
MST-CDC	54	201	460	802	1532	3553
R-MST	2	6	14	24	54	89

As shown in Figure 16, we used the program to generate a 10-blobs (number of 10-blobs = 10,000) for testing the effect of the parameter k_2 on the number of representative points, and on the running time of the algorithm. These experiments were carried out with constant parameter k_1. Figure 17 demonstrates the change in the number of representative points as k_2 is increased. Figure 18 demonstrates the change in algorithm running time as k_2 is increased.

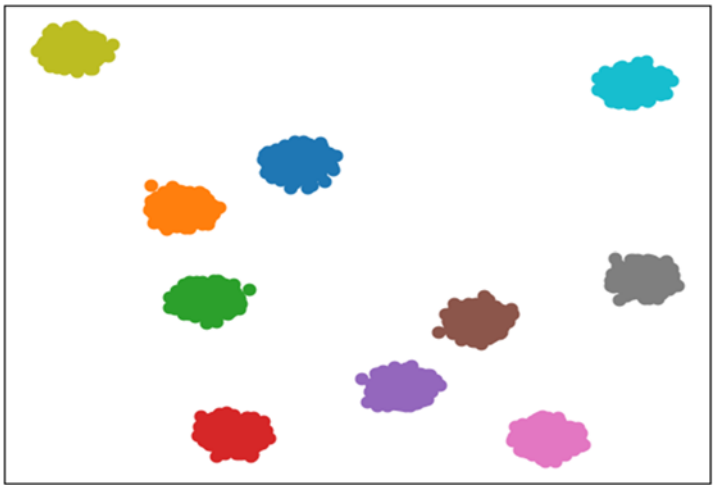


Figure 16. The 10-blobs dataset.

As shown in Figure 17, when k_2 gradually increases, the number of representative points initially decreases, reaching a critical value (when $k_2 = 150$) and maintaining a relatively stable state. Finally, reaching another critical value (when $k_2 = 400$), the number of representative points begins to gradually increase. When the number of representative points reaches a stable state, its ratio to the number of all points in the dataset is approximately 1:20. We know that the time complexity of constructing MST for all points in the dataset using the Prime algorithm is $O(n^2)$, where n represents the number of all points in the dataset. So, the time complexity of constructing MST for representative points is $O((n/20)^2)$. Therefore, the performance of R-MST has been significantly improved.

As shown in Figure 18, when k_2 gradually increases, the running time of the algorithm is kept in a stable range at first, and after the critical value of $k_2 = 450$, the running time begins to increase significantly. When $k_2 = 50$ and $k_2 = 100$, the number of representative points is relatively large, but the running time of the algorithm is less, because when k_2 becomes smaller, the time spent calculating k_2 neighbors decreases, so the overall time changes little. When $k_2 > 450$, the number of representative points increases, and the time spent calculating k_2 neighbors also increases, so the overall time will increase significantly. Therefore, the optimal range for parameter k_2 is approximately $n/200 \leq k_2 \leq n/25$, where n represents the size of the original dataset. Within this range, changes in the value of k_2 have a relatively minor impact on the efficiency of the algorithm.

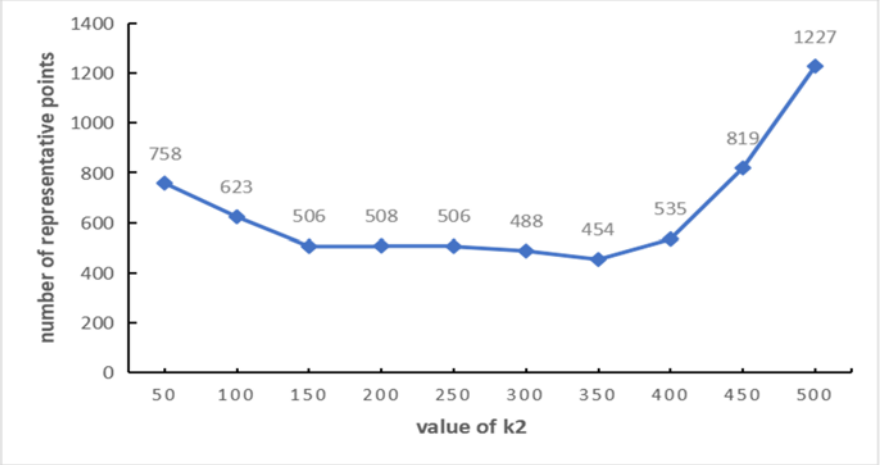


Figure 17. The influence of the value of k_2 on the number of representative points.

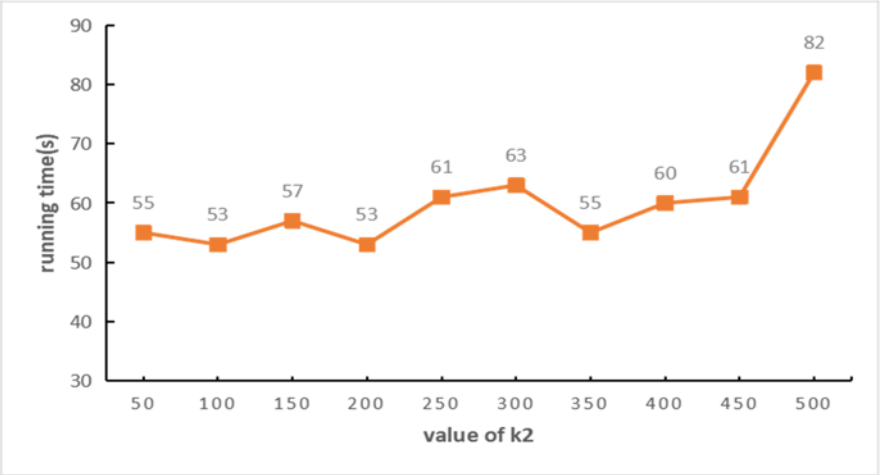


Figure 18. The Influence of k_2 value on Running Time.

4.3. Experimental results on UCI datasets

In this study, we conducted comparative experiments on 12 real high-dimensional UCI datasets to verify the effectiveness of the R-MST algorithm on high-dimensional datasets. These datasets are sourced from various real-world domains, including healthcare, biology, finance and others. As a result, the dimensions and characteristics of the data are highly diverse, making them suitable for testing the

generalizability of clustering algorithms. Additionally, UCI datasets often exhibit issues such as outliers, missing values and duplicates, which provide opportunities to evaluate and assess the robustness and resilience of clustering algorithms. The comparative algorithms involved in the experiment include DPC, DBSCAN and MST-CDC. The FastDP aims to improve the efficiency of DPC, and there is no significant change in clustering quality compared to DPC. Therefore, we did not use it for clustering quality comparison in the UCI datasets. The basic information of the experimental test dataset is shown in Table 2. The optimal parameters of the four algorithms on 12 UCI datasets are shown in Table 5, and the specific clustering performance is shown in Table 6. The results of the four algorithms based on measurements from the UCI dataset indicate that, except for the cancer and WBC datasets, the three indicators of the algorithm proposed in this paper are superior to the other three comparative algorithms on other datasets. For the Cancer and WBC datasets, the R-MST algorithm achieved the highest ARI and NMI values, followed by the HOMO values. In summary, the R-MST performs exceptionally well on UCI datasets, exhibiting better generalizability and robustness in practical applications.

Table 5. Optimal parameters of 4 algorithms on 12 UCI datasets.

Algorithm	DPC	DBSCAN	MST-CDC	R-MST
Parameters	dc	Eps/MinPts	None	k_1/k_2
Zoo	0.82	1.2/4	---	16/3
Cancer	0.98	0.4/6	---	10/19
Seeds	0.08	0.2/9	---	5/16
WBC	1.01	0.4/6	---	10/19
Wine	0.42	0.5/6	---	7/6
Ecoil	0.48	0.2/10	---	5/2
Iris	0.17	0.4/3	---	9/6
Vote	0.03	0.9/10	---	24/17
Vowel	0.15	0.1/10	---	23/8
WDBC	0.08	0.4/20	---	26/14
Dermatology	0.16	1.4/19	---	20/4
Pendigits	0.28	0.3/5	---	23/2

5. Discussion

This paper proposes a fast-clustering algorithm based on MST of representative points. The algorithm replaces all points in the dataset with representative points to construct an MST, reducing a significant amount of computational overhead. In addition, we propose an adaptive method to identify inconsistent edges in the MST. After removing these inconsistent edges, the number of clusters can be effectively obtained. Experimental results demonstrate that this algorithm has high efficiency and clustering quality. However, as the amount of data increases, the number of representative points will also increase. This is unfavorable for handling large-scale datasets.

Table 6. Clustering performance of 4 algorithms on 12 UCI datasets.

Dataset	Algorithm	ARI	NMI	Homo
Zoo	DPC	0.4972	0.7224	0.7490
	DBSCAN	0.9326	0.8968	0.8978
	MST-CDC	0	0	0
	R-MST	**0.9515**	**0.9137**	**0.9109**
Cancer	DPC	0.4934	0.4404	0.3964
	DBSCAN	0.8362	0.7456	**0.7537**
	MST-CDC	0.0050	0.0271	0.0048
	R-MST	**0.8522**	**0.7530**	0.7530
Seeds	DPC	0.7448	0.7194	0.7169
	DBSCAN	0.3693	0.5062	0.5788
	MST-CDC	0	0.0233	0.0061
	R-MST	**0.8109**	**0.7707**	**0.7702**
WBC	DPC	0.4934	0.4404	0.3964
	DBSCAN	0.8362	0.7456	**0.7537**
	MST-CDC	0.0050	0.0271	0.0048
	R-MST	**0.8522**	**0.7530**	0.7530
Wine	DPC	0.6724	0.7104	0.7096
	DBSCAN	0.4264	0.5266	0.4978
	MST-CDC	-0.0087	0.0881	0.0344
	R-MST	**0.7847**	**0.7872**	**0.7896**
Ecoli	DPC	0.5618	0.5761	0.5017
	DBSCAN	0.4999	0.5109	0.4104
	MST-CDC	0.0610	0.1849	0.0796
	R-MST	**0.7691**	**0.7279**	**0.7048**
Iris	DPC	0.8857	0.8642	0.8640
	DBSCAN	0.5681	0.7337	0.5794
	MST-CDC	0.5681	0.7337	0.5794
	R-MST	**0.9222**	**0.9011**	**0.9009**
Vote	DPC	0.5921	0.5150	0.5241
	DBSCAN	0.4481	0.3977	0.5035
	MST-CDC	0.0746	0.0951	0.0349
	R-MST	**0.6353**	**0.5438**	**0.5520**
Vowel	DPC	0.4596	0.5658	0.5564
	DBSCAN	0.0076	0.0187	0.0105
	MST-CDC	0.2487	0.4717	0.5834
	R-MST	**0.5132**	**0.6030**	**0.6603**
WDBC	DPC	0.4964	0.4822	0.4374
	DBSCAN	0.4515	0.3560	0.3622
	MST-CDC	0.0048	0.0102	0.0053
	R-MST	**0.6879**	**0.5828**	**0.5680**
Dermatology	DPC	0.5293	0.6851	0.5531
	DBSCAN	0.4639	0.6522	0.5661
	MST-CDC	0.2048	0.4514	0.2959
	R-MST	**0.7756**	**0.8484**	**0.8122**
Pendigits	DPC	0.6478	0.7776	0.7630
	DBSCAN	0.5633	0.7384	0.7922
	MST-CDC	0.2053	0.0951	0.0349
	R-MST	**0.7356**	**0.8323**	**0.9020**

We have found that only a few edges in the complete graph play a role in constructing the MST, and these useful edges mostly connect vertices and their neighbors. This gives us some ideas, and we will try to construct a graph based on the neighbor relationship that can connect all vertices but has very few edges. The number of edges in this graph is only a few times the number of vertices, and then we will use the Kruskal algorithm [36] on this graph to obtain the minimum spanning tree. This will reduce the time complexity of our algorithm to $0(nlogn)$ and break the quadratic time complexity limitation of constructing MST in clustering algorithms. Because the time complexity of Kruskal algorithm is $0(eloge)$, where e is the number of edges in the graph, this algorithm is particularly suitable for finding the minimum spanning tree of graphs with sparse edges. The reason why Kruskal algorithm is not used in complete graphs is that the number of edges in a complete graph is $n(n-1)/2$, which would make the overall time complexity of clustering algorithms become $0(n^2logn)$. Furthermore, preserving a small set of representative points for extremely large datasets poses a challenge. In our future work, we plan to alleviate this problem by employing grid-based techniques. First, we divide the data space into regular grid cells. Then, based on characteristics such as the data volume and density within the grid cell, along with the features of adjacent grids, we select one or more suitable representative points from each grid. We can efficiently control the number of representative points by flexibly adjusting the grid size according to our needs. For instance, if we want to reduce the number of representative points, we can achieve this by enlarging the grid. The trade-off between grid size and accuracy is a challenge. On the one hand, a smaller grid size can provide more detailed information, but it may increase computational complexity and memory requirements. On the other hand, larger mesh sizes may sacrifice accuracy or fail to capture local changes. Finally, we will promote the combination of "selecting core points + selecting representative points" to more clustering algorithms to improve their performance. The essence of selecting core points is to roughly remove noise and boundary points from the dataset, as these points can have a significant impact on the effectiveness of clustering. Then, selecting representative points from the core points allows for reducing the data size while preserving the characteristics of the clusters. This combination can be seen as a preprocessing step for clustering algorithms, which not only mitigates the interference of noise and boundary points on the algorithm but also improves clustering efficiency to some extent. However, this combination also faces some challenges. First, the definition criteria for noise and boundary points may need to be determined based on specific application scenarios and requirements, which can lead to unstable selection of core points. For example, determining noise points and boundary points by setting distance thresholds. In density-based clustering algorithm DBSCAN, if the number of points in the neighborhood of a point is less than a certain threshold, the point is considered a noise point or boundary point. The selection criteria for noise points and boundary points can also be defined based on specialized knowledge in specific fields. For example, in image processing, changes in pixel intensity or texture continuity can be considered to determine noise points and boundary points. Second, selecting representative points requires balancing the preservation of key cluster features with the goal of compressing data to avoid information loss. This may necessitate the adoption of different strategies and metrics for selecting representative points. For example, spectral clustering can select nodes in each partition that are highly connected to other partitions as representative points. These nodes cannot only represent the characteristics of the partition they belong to, but also have some differences. The density peak clustering selects the center point of each cluster as a representative point to better represent the characteristics of the cluster.

Use of AI tools declaration

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

Acknowledgments

The financial support for this project is provided by the National Natural Science Foundation of China [61962054].

Conflict of interest

The authors declare there is no conflict of interest.

References

1. X. Xue, J. Chen, Matching biomedical ontologies through compact differential evolution algorithm with compact adaption schemes on control parameters, *Neurocomputing*, **458** (2021), 526–534. https://doi.org/10.1016/j.neucom.2020.03.122
2. X. Xue, Y. Wang, Ontology alignment based on instance using NSGA-II, *J. Inf. Sci.*, **41** (2015), 58–70. https://doi.org/10.1177/0165551514550142
3. D. S. Silva, M. Holanda, Applications of geospatial big data in the Internet of Things, *Trans. GIS*, **26** (2022), 41–71. https://doi.org/10.1111/tgis.12846
4. T. Xu, J. Jiang, A graph adaptive density peaks clustering algorithm for automatic centroid selection and effective aggregation, *Expert Syst. Appl*., **195** (2022), 116539. https://doi.org/10.1016/j.eswa.2022.116539
5. F. U. Siddiqui, A. Yahya, F. U. Siddiqui, A. Yahya, Partitioning clustering techniques, in *Clustering Techniques for Image Segmentation*, Springer, (2022), 35–67. https://doi.org/10.1007/978-3-030-81230-0_2
6. F. U. Siddiqui, A. Yahya, F. U. Siddiqui, A. Yahya, Novel partitioning clustering, in *Clustering Techniques for Image Segmentation*, Springer, (2022), 69–91. https://doi.org/10.1007/978-3-030-81230-0_3
7. C. K. Reddy, B. Vinzamuri, A survey of partitional and hierarchical clustering algorithms, in *Data Clustering*, Chapman and Hall/CRC, (2018), 87–110. https://doi.org/10.1201/9781315373515-4
8. S. Zhou, Z. Xu, F. Liu, Method for determining the optimal number of clusters based on agglomerative hierarchical clustering, *IEEE Trans*. *Neural Networks Learn. Syst*., **28** (2016), 3007–3017. https://doi.org/10.1109/TNNLS.2016.2608001
9. E. C. Chi, K. Lange, Splitting methods for convex clustering, *J. Comput. Graphical Stat*., **24** (2015), 994–1013. https://doi.org/10.1080/10618600.2014.948181
10. M. Ester, H. P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *kdd*, **96** (1996), 226–231.
11. P. Bhattacharjee, P. Mitra, A survey of density based clustering algorithms, *Front. Comput. Sci.*, **15** (2021), 1–27. https://doi.org/10.1007/s11704-019-9059-3
12. A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, *Science*, **344** (2014), 1492–1496. https://doi.org/10.1126/science.1242072

13. S. Sieranoja, P. Fränti, Fast and general density peaks clustering, *Pattern Recognit. Lett*., **128** (2019), 551–558. https://doi.org/10.1016/j.patrec.2019.10.019
14. A. Joshi, E. Fidalgo, E. Alegre, L. Fernández-Robles, SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders, *Expert Syst. Appl*., **129** (2019), 200–215. https://doi.org/10.1016/j.eswa.2019.03.045
15. J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in *International Conference on Machine Learning*, PMLR, (2016), 478–487. https://doi.org/10.48550/arXiv.1511.06335
16. M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, IEEE, (2005), 729–734. https://doi.org/10.1109/IJCNN.2005.1555942
17. C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach, preprint, arXiv:1906.06532.
18. R. Jothi, S. K. Mohanty, A. Ojha, Fast approximate minimum spanning tree based clustering algorithm, *Neurocomputing*, **272** (2018), 542–557. https://doi.org/10.1016/j.neucom.2017.07.038
19. J. C. Gower, G. J. Ross, Minimum spanning trees and single linkage cluster analysis, *J. R. Stat. Soc. C*, **18** (1969), 54–64. https://doi.org/10.2307/2346439
20. C. T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, *IEEE Trans. Comput*., **100** (1971), 68–86. https://doi.org/10.1109/T-C.1971.223083
21. O. Grygorash, Y. Zhou, Z. Jorgensen, Minimum spanning tree based clustering algorithms, in *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, IEEE, (2006), 73–81. https://doi.org/10.1109/ICTAI.2006.83
22. A. C. Müller, S. Nowozin, C. H. Lampert, Information theoretic clustering using minimum spanning trees, in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, Springer, (2012), 205–215. https://doi.org/10.1007/978-3-642-32717-9_21
23. M. Gagolewski, M. Bartoszuk, A. Cena, Genie: A new, fast, and outlier-resistant hierarchical clustering algorithm, *Inf. Sci.*, **363** (2016), 8–23. https://doi.org/10.1016/j.ins.2016.05.003
24. Y. Ma, H. Lin, Y. Wang, H. Huang, X. He, A multi-stage hierarchical clustering algorithm based on centroid of tree and cut edge constraint, *Inf. Sci.*, **557** (2021), 194–219. https://doi.org/10.1016/j.ins.2020.12.016
25. G. Mishra, S. K. Mohanty, A fast hybrid clustering technique based on local nearest neighbor using minimum spanning tree, *Expert Syst. Appl*., **132** (2019), 28–43. https://doi.org/10.1016/j.eswa.2019.04.048
26. F. Şaar, A. E. Topcu, Minimum spanning tree-based cluster analysis: A new algorithm for determining inconsistent edges, *Concurrency Comput. Pract. Exper.*, **34** (2022), e6717. https://doi.org/10.1002/cpe.6717
27. H. A. Chowdhury, D. K. Bhattacharyya, J. K. Kalita, UIFDBC: Effective density based clustering to find clusters of arbitrary shapes without user input, *Expert Syst. Appl*., **186** (2021), 115746. https://doi.org/10.1016/j.eswa.2021.115746
28. R. C. Prim, Shortest connection networks and some generalizations, *Bell Syst. Tech. J.*, **36** (1957), 1389–1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x
29. F. Ros, S. Guillaume, Munec: a mutual neighbor-based clustering algorithm, *Inf. Sci.*, **486** (2019), 148–170. https://doi.org/10.1016/j.ins.2019.02.051

30. D. Steinley, Properties of the hubert-arable adjusted rand index, *Psychol. Methods*, **9** (2004), 386. https://doi.org/10.1037/1082-989X.9.3.386
31. P. A. Estévez, M. Tesmer, C. A. Perez, J. M. Zurada, Normalized mutual information feature selection, *IEEE Trans. Neural Networks*, **20** (2009), 189–201. https://doi.org/10.1109/TNN.2008.2005601
32. M. Sato-Ilic, On evaluation of clustering using homogeneity analysis, in *IEEE International Conference on Systems, Man and Cybernetics*, IEEE, **5** (2000), 3588–3593. https://doi.org/10.1109/ICSMC.2000.886566
33. P. Fränti, Clustering datasets, 2017. Available from: https://cs.uef.fi/sipu/datasets.
34. P. Fränti, S. Sieranoja, K-means properties on six clustering benchmark datasets, *Appl. Intell.*, **48** (2018), 4743–4759. https://doi.org/10.1007/s10489-018-1238-7
35. D. Dua, C. Graff, *UCI Machine Learning Repository*, 2017. Available from: https://archive.ics.uci.edu/ml.
36. J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proc. Am. Math. Soc.*, **7** (1956), 48–50.

AIMS Press