Highlights

Clustering Based on Local Density Peaks and Graph Cut*

Zhiguo Long, Yang Gao, Hua Meng, Yuqin Yao, Tianrui Li

- A novel measure of similarity between local tree-structured clusters to capture complex structures of data is proposed.
- A new Ncut-based loss function is devised in consideration of tree structures.
- The effectiveness of the proposed algorithm is verified on both real-world and synthetic complex data sets.

Clustering Based on Local Density Peaks and Graph Cut

Zhiguo Long[a], Yang Gao[a], Hua Meng[b,*], Yuqin Yao[b], Tianrui Li[a]

[a]*School of Computing and Artificial Intelligences, Southwest Jiaotong University, Chengdu, 611756, China*
[b]*School of Mathematics, Southwest Jiaotong University, Chengdu, 611756, China*

Abstract

Clustering by fast search and find of density peaks (DPC) is a widely used and studied clustering algorithm. In this article, we notice that DPC can achieve highly accurate clustering results when restricted to local neighborhoods. Therefore, by investigating density information in local neighborhoods, we propose to capture latent structures in data with *family trees*, which can reflect density dominations among nearest neighbors of data. A data set will then be partitioned into multiple family trees. In order to obtain the final clustering result, instead of exploiting the error-prone allocation strategy of DPC, we first elaborately design a novel similarity measure for family trees, characterizing not only the distance between data points, but also the structure of trees. Then, we adapt graph cut for the corresponding connection graph to also take global structural information into account. Extensive experiments on both real-world and synthetic data sets show that the proposed algorithm can outperform several prominent clustering algorithms for most of the cases, including the DPC and spectral clustering algorithms and some of their latest variants. We also analyze the robustness of the proposed algorithm w.r.t. hyper-parameters and its time complexity, as well as the necessity of its components through ablation study.

Keywords: Clustering, Density peaks, Spectral clustering, Local density, Similarity between trees

1. Introduction

Clustering [1] is an important unsupervised problem in the field of data mining and machine learning. It aims to divide a data set into multiple disjoint subsets, where the data in each subset are as similar as possible, and the data between subsets are as dissimilar as possible [2, 3]. Clustering can reveal inherent or latent knowledge and rules in data and has been widely used in scientific research and engineering applications [4, 5].

In the past decades, a large number of clustering algorithms have been proposed and applied in different scenarios. Traditional clustering algorithms can be roughly divided into

*Accepted version. Formal version available at: https://doi.org/10.1016/j.ins.2022.03.091
*Corresponding author.
Email address: menghua@swjtu.edu.cn (Hua Meng)

five categories: division based [6], hierarchical based [7], density based [8], model based [9], and grid based [10]. In addition, there are also extensions like spectral clustering [11], multi-view clustering [12], clustering ensemble [13] and deep clustering [14]. Each of these algorithms has its own advantages and disadvantages for certain type of data or applications.

Clustering by fast search and find of density peaks (DPC; [15]) is an influential density based clustering algorithm. Although shown to perform well on a variety of data, it has several problems such as: it cannot properly process data sets with variant densities in different part; its usage of Euclidean distance for calculating densities and searching of density peaks is not appropriate for manifold structures; its allocation strategy of clusters can suffer from the domino effect that the misallocation of one point will result in misallocation of all subsequent points. A large number of improvements and extensions of the original DPC algorithm were proposed to overcome those problems. For example, KNN-DPC [16] and FDPC [17] try to improve DPC by optimizing the calculation of densities; FKNN-DPC [18] and SNN-DPC [19] further improve allocation strategy; DLORE-DP [20] and DPC-GD [21] introduce geodesic distance to adapt for manifold structures; FastDPeak [22] and DPCG [23] optimize computational efficiency.

On the other hand, in this article, instead of improving density calculation, peak selection, or allocation strategy of DPC, we try to make use of certain merit in the idea of DPC to devise a new clustering algorithm. In fact, although the allocation strategy of DPC is not always reliable globally, it can achieve high *purity* in small local subsets of data (cf. Fig. 2), i.e., locally, points that should be in the same cluster are also allocated to the same cluster. By restricting DPC to local neighborhoods, we can obtain tree structures (cf. Fig. 1) that reflect density domination relations in local neighborhood, and each of these trees has high purity. The aforementioned DLORE-DP considers this kind of local structures as natural accumulations of data points, and makes use of DPC to cluster these accumulations in the subsequent stage. FHC-LDP [24] uses these local trees as sub-clusters and combines them with the help of hierarchical clustering. Besides, ADBC-KNN [25] and CDMC-IA [26] exploit the idea of DPC to obtain sub-clusters, and propose strategies to achieve final clustering, i.e., combining sub-clusters according to density reachability and quantity affinity (in a hierarchical clustering way), respectively.

In this article, we further investigate the properties of such tree-based local structures, and propose a novel clustering algorithm (LDP-SC), by designing a more sophisticated similarity measure of these trees, and adopting an improved graph cut method. The use of graph cut method can better determine whether to combine or to separate these trees, because it takes into account more global structural information of data, than hierarchical clustering, which mainly considers local information of sub-clusters. This graph cut method also avoids the identification of cluster centers, which is an error-prone process, especially for complex data.

This article has the following contributions:

1. We study the properties of local tree-structured clusters (called *family trees*), and propose a novel similarity measure for such local clusters, which helps to capture complex structures of data.

2. We exploit the idea of spectral clustering on these local structures to obtain the final clustering result, which avoids the error-prone steps of the identification of cluster centers and the allocation of clusters.
3. We improve the loss function for graph cut to better suit local tree-structured clusters.
4. The proposed algorithm outperforms several prominent baseline algorithms, including the DPC and spectral clustering algorithms and some of their latest variants.

The rest of this article is organized as follows. In Section 2, research works on improving the DPC and spectral clustering algorithms in recent years are discussed, as well as work on the combination of DPC and spectral clustering. In Section 3, preliminaries for DPC and spectral clustering are introduced, as well as an analysis for the motivation of the improvements proposed in this article. Section 4 gives details on the proposed LDP-SC algorithm. Section 5 then presents a number of experiments for empirically evaluating the proposed algorithm against some benchmark algorithms from various aspects and on different data sets. Section 6 discusses the robustness of the LDP-SC algorithm, and performs ablation study to confirm the significance of each component in the algorithm. Section 7 concludes the article.

2. Related Work

In recent years, a large number of variants of DPC have been proposed to overcome its defects. Generally, DPC consists of three essential parts, i.e., the calculation of densities, the selection of cluster centers, and the cluster allocation strategy of non-center points.

KNN-DPC [16] uses K-nearest neighbors to replace the use of cutoff distance for the calculation of densities. FDPC [17] and FKNN-DPC [18] exploit fuzzy theory to characterize uncertainties introduced by the sampling of data points, and thus generalize the calculation of densities. FKNN-DPC also modifies the allocation strategy by making use of neighboring relations. Based on KNN-DPC and FKNN-DPC, DPC-DBFN [27] uses a fuzzy kernel to extract a specific kind of structural information, called *density backbone*, to improve the robustness of clustering. SNN-DPC [19] investigates local structures of data, and characterizes similarities between data points in a more sophisticated way by considering shared nearest neighbors. It further improves the calculation of densities by using the new similarity measure, and the allocation strategy by distinguishing inevitable and possible subordinate points. BPEC [28] replaces densities with beliefs and the allocation strategy with an optimization problem called *credal partition*, and can provide fuzzy or even rough partitions. CFDPC [29] proposes an automatic method to filter out density peaks and thus several initial clusters, conducts adaptive searching of core points, and finally fuses initial clusters based on these core points in a hierarchical way. Considering manifold structures of some data, DPC-GD [21] and DLORE-DP [20] replace Euclidean distance with geodesic distance to improve the applicability to manifold structures. ADBC-KNN [25] and CDMC-IA [26] select multiple centers (generally more than the number of target clusters), by setting a threshold on the value of γ of DPC, and then combine the sub-clusters determined by these centers to obtain the final clustering result, by density reachability and by quantity affinity in a

hierarchical way, respectively. Instead of selecting centers, LDP-MST [30] generates several minimum spanning trees and constructs connection relations between these trees. The final clustering result is obtained by removing some of the edges between trees. FHC-LDP [24] builds sub-clusters with local density structures, and then combines these sub-clusters by hierarchical clustering. It is worth noting that, identifying sub-clusters is a way to discover *local* structural information, and combining them with hierarchical clustering or density reachability emphasizes again on *local* information. Instead, here we propose to use graph cut method for the purpose of aggregating sub-clusters, which arguably better balances *local* and *global* information.

The spectral clustering algorithm [3] is a typical graph cut method, which is simple, effective and widely used. However, one of the most obvious disadvantages is its high time complexity $\mathcal{O}(m^3)$, where m is the number of data points, and in recent years, improvements of the spectral clustering algorithm have mainly focused on computational efficiency. To achieve the purpose of acceleration, Cheng et al. [31] randomly select p representative points to approximate the original data set and make use of approximate matrix decomposition. Cai et al. [32] propose the landmark-based spectral clustering algorithm (LSC), which obtains p representative points as landmarks by random selection or K-means pre-clustering, and then uses sparse linear combinations of these landmarks to approximate adjacency matrix. Huang et al. [33] use approximated K-nearest neighbors to construct a sparse adjacency submatrix and interpret connections between data points as a bipartite graph, which is then divided into clusters by the Ncut algorithm. This enables the spectral clustering algorithm to efficiently perform clustering on data sets with millions of data points. Our idea of performing graph cut on family trees obtained from local structural information improves both the computational efficiency and effectiveness of graph cut, because points in the same family tree will be considered as a whole and will never be separated.

There are also some researches on combining the ideas of the DPC algorithm with the spectral clustering algorithm. Li et al. [34] propose the DPSC algorithm, which replaces the K-means clustering with DPC in the second stage of spectral clustering, so that the decision graph of DPC could be used to determine the number of clustering and filter noises. Liu and Zhao [35] use changes of densities to improve the DPC, making it more effective for data with uneven density. They also apply the improved DPC algorithm to the second stage of spectral clustering.

3. Preliminaries

3.1. DPC

The idea of DPC to cluster data is first to identify several density peaks as cluster centers and then to allocate the other data points to the clusters of the corresponding centers. For cluster centers, DPC makes the following assumptions:

1. A cluster center has a higher density than the surrounding data points;
2. The distance between cluster centers is far.

Based on these assumptions, DPC calculates ρ_i and δ_i to determine which data points can be cluster centers, where ρ_i represents the density of a data point x_i and δ_i represents the distance between x_i and its nearest neighbor with higher density, defined as:

$$\rho_i = \sum_{i \neq j} \chi\left(d_{ij} - d_{\mathrm{c}}\right) \ , \text{ where } \chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \ , \text{ and} \tag{1}$$

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho_i} d_{ij} & \text{if } \exists j \text{ s.t. } \rho_j > \rho_i, \\ \max\limits_{j} d_{ij} & \text{otherwise.} \end{cases} \tag{2}$$

Here, d_{ij} is the Euclidean distance between points x_i and x_j, and d_{c} refers to the cutoff distance, which is a given hyper-parameter. The original paper of DPC [15] suggests that d_{c} should be selected to cover 1-2% of the total number of points in data sets.

Alternatively, ρ_i can also be defined using a Gaussian kernel as follows:

$$\rho_i = \sum_{i \neq j} \exp\left[-\left(\frac{d_{ij}}{d_{\mathrm{c}}}\right)^2\right]. \tag{3}$$

After obtaining ρ_i and δ_i, there are two commonly used methods to identify C cluster centers, where C represents the number of target clusters. The first method is to draw a decision graph with δ_i and ρ_i as the x-axis and the y-axis, respectively, and then to manually select C cluster centers. The second method is to calculate $\gamma_i = \rho_i \times \delta_i$ and select the C data points with the first C largest values of γ_i as the cluster centers.

Once cluster centers are identified, DPC allocates each center to a cluster, and then each of the other data points is allocated to the cluster that its nearest neighbor with higher density belongs to.

The assumptions and strategies of DPC have been shown to be effective in many cases. However, DPC fails to find correct cluster centers sometimes. For example, when data have various densities [19], data points with highest γ_i could lie in the same ground-truth cluster but their distance could be large enough, which makes DPC choose cluster centers in the same ground-truth cluster and then fail with any allocation strategy. Also, since DPC is based on Euclidean distance, it performs poorly on data sets with complex structures [19], e.g., as illustrated in experiments of this article, it is not suitable for data sets with manifold structure. Finally, the choice of its parameter d_{c}, the cutoff distance, is also not an easy task.

3.2. Spectral Clustering

Spectral clustering is another widely used clustering algorithm. The idea of spectral clustering is: 1) to construct a connection graph with data points as nodes and their similarity measures as weights of edges; 2) to optimize the division of the connection graph into subgraphs s.t. the connection (similarity) between subgraphs is low and the connection of nodes within each subgraph is high; 3) to find an approximate solution to the former

optimization problem and obtain a new representation of data; 4) to perform clustering on the new representation.

Given a data set $X = \{x_1, x_2, \ldots, x_m\}$, where m is the number of data points and the dimension of each data point is n, an adjacency matrix W is used to represent the connection graph, where W_{ij} represents the similarity value between x_i and x_j. A commonly used loss function in spectral clustering is the Ncut function [36]. Suppose C is the number of target clusters. Let $\mathrm{cut}(A, \overline{A}) = \frac{1}{2}\sum_{x_i \in A, x_j \in \overline{A}} W_{ij}$ (where $\overline{A} = X \setminus A$) and $\mathrm{vol}(A) = \sum_{x_i \in A, x_j \in X} W_{ij}$, for some $A \subseteq X$, then

$$\mathrm{Ncut}\left(A_1, \ldots, A_C\right) = \sum_{i=1}^{C} \frac{\mathrm{cut}\left(A_i, \overline{A_i}\right)}{\mathrm{vol}\left(A_i\right)}. \tag{4}$$

Finding optimal $A_1, \ldots, A_C$ for this objective function is an NP-hard problem and can be approximated by a relaxed problem, as follows. Let $h_j = (h_{1j}, h_{2j}, \cdots, h_{mj})^{\mathrm{T}}$, where

$$h_{ij} = \begin{cases} 0 & x_i \notin A_j \\ \frac{1}{\sqrt{\mathrm{vol}(A_j)}} & x_i \in A_j \end{cases}. \tag{5}$$

Denote by D the diagonal matrix s.t. $D_{ii} = \sum_j W_{ij}$ and by L the Laplacian matrix $L = D - W$, then

$$\begin{aligned} {h_i}^{\mathrm{T}} L h_i &= \frac{1}{2}\sum_{u,v} W_{ij}(h_{ui} - h_{vi})^2 \\ &= \frac{1}{2}\sum_{x_u \in A_i, x_v \in \overline{A_i}} W_{uv}\frac{1}{\mathrm{vol}\left(A_i\right)} + \frac{1}{2}\sum_{x_u \in \overline{A_i}, x_v \in A_i} W_{uv}\frac{1}{\mathrm{vol}\left(A_i\right)} \\ &= \frac{1}{2}\mathrm{cut}(A_i, \overline{A_i})\frac{1}{\mathrm{vol}\left(A_i\right)} + \frac{1}{2}\mathrm{cut}(\overline{A_i}, A_i)\frac{1}{\mathrm{vol}\left(A_i\right)} \\ &= \frac{\mathrm{cut}(A_i, \overline{A_i})}{\mathrm{vol}\left(A_i\right)}. \end{aligned} \tag{6}$$

In this way, the problem of minimizing the Ncut loss function is converted to:

$$\min_F \mathrm{tr}\left(F^T D^{-1/2} L D^{-1/2} F\right) \qquad \text{s.t.} \quad F^T F = I, \tag{7}$$

where $F = D^{1/2}H$ and the matrix $H = (h_1, \ldots, h_C)$.

This optimization problem is a typical Rayleigh quotient problem [37] and the optimal F consists of the C eigenvectors corresponding to the first C smallest eigenvalues of $D^{-1/2}LD^{-1/2}$. The optimal solution H to the original problem can be obtained by calculating $H = D^{-1/2}F$. The final clustering result is obtained by K-means clustering on H, taking each row of H as a feature vector. Another commonly used method comes from Ng et al. [3], where instead of calculating H, it just standardizes F by row obtaining $Y_{ij} = F_{ij}/(\sum_{j'} F_{ij'}^2)^{1/2}$. Again, to obtain the final clustering result, K-means clustering is conducted on Y, taking each row of Y as a feature vector. This article will also adopt the latter strategy.

Spectral clustering does not make any assumptions about the structure of the data and, as a result, has good performance on data of various structures. However, spectral clustering has several significant problems: 1) it needs to calculate eigenvectors of the matrix and the time complexity is $\mathcal{O}(m^3)$ which is too high for large data sets; 2) a good similarity measure to form the adjacency matrix is crucial for its performance.

3.3. Analysis

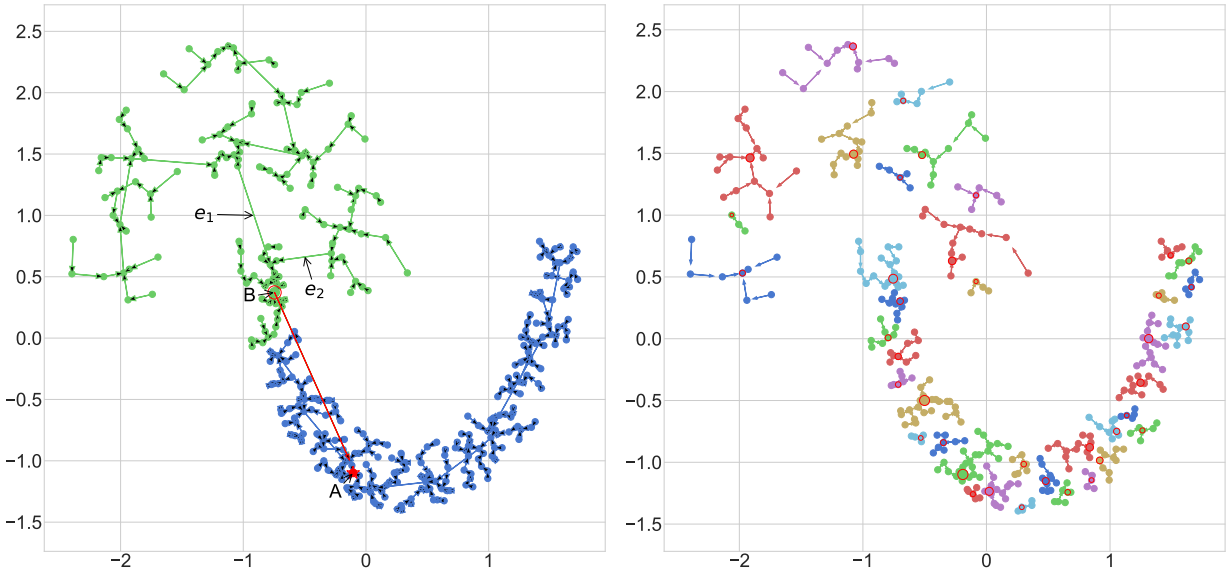

Figure 1: DPC clustering (left) and results with family trees (right).

DPC and many of its variants identify cluster centers based on the criteria that the centers should be points with globally higher density that are far apart from each other in terms of Euclidean distance. As densities could vary significantly across the whole data set and the data set might have a complex structure, this criteria is sometimes problematic. Consider Fig. 1 for example, where an arrow points from a data point to its nearest neighbor with higher density. On the left of the figure, point A is identified as the first cluster center by DPC; point B is a point with high density and is far enough from A, and thus DPC identifies it as the second cluster center, which is wrong. Moreover, as DPC uses Euclidean distance to measure the distance between a data point and its nearest neighbor with higher density, DPC allocates data points from the other cluster to the same cluster of B, e.g., through arrows e_1 and e_2. Both of the above make DPC have a poor clustering performance on this data set.

However, it is interesting to note that the strategy of DPC can have better clustering performance when restricted to local neighborhood. To illustrate this, we vary the number of target clusters from 2 to 29 for each of the four data sets, and measure the correctness of clusters by *purity*. The purity of a clustering result on a data set is defined as the $1 - \sum_i \frac{|c_i|}{m} \mathsf{entropy}\,(c_i)$, where m is the number of data points, c_i stands for the i-th cluster in

the result, and $\mathsf{entropy}(c_i)$ represents the information entropy of the real label distribution corresponding to i-th cluster. This measures the proportion of data points that are correctly clustered together. Fig. 2 shows the results. With the increase of the number of target clusters, or in other words, the decrease of the range of local neighborhood, the purity of clustering results increases constantly. This means that DPC is likely to correctly cluster data points when restricted to a local neighborhood.

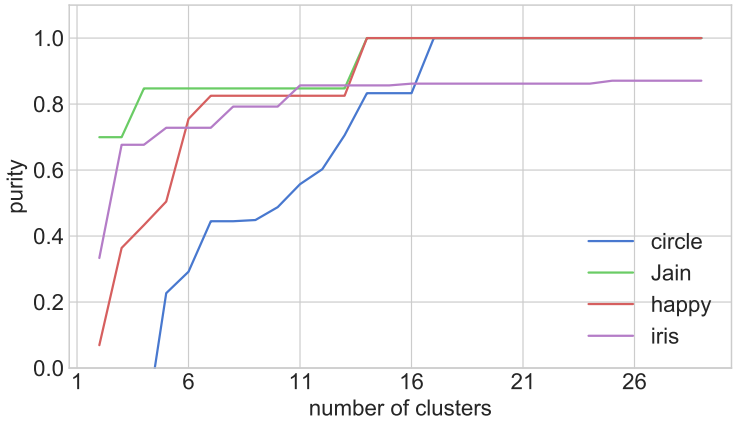

Figure 2: Purity change of DPC with respect to the number of target clusters.

The above observation inspires us to consider applying the idea of DPC in local proximity of data points with some necessary modifications, to first partition a data set into smaller local clusters. Here, we consider finding the nearest neighbor with higher density for a data point only from its k nearest neighbors, and if none is found then we treat this data point as a local density peak. The figure on the right of Fig. 1 shows the result of this idea. In the figure, the data set is partitioned into smaller local clusters (called *family trees*), where points with locally highest density are marked by circles with red boundary. Note that the incorrect inter-cluster connections disappear in the result.

The next important question is then how to combine these intermediate partitions into final target clusters. To this end, we exploit the idea of spectral clustering on family trees. In particular, we devise a novel similarity measure for these family trees to construct the adjacency matrix for spectral clustering. Unlike the original spectral clustering algorithm, this similarity measure also takes into account the contribution of the structure of family trees. In addition, we adapt the Ncut loss function for these family trees. As a result, we propose an algorithm combining the idea of local density peaks and spectral clustering (LDP-SC), and this algorithm retains several advantages of both DPC and spectral clustering.

4. LDP-SC Algorithm

The LDP-SC algorithm consists of two stages: constructing family trees so that all the data points are *locally* clustered, and clustering of family trees so that all the data points are *globally* clustered.

4.1. Construction of Family Trees

As we have mentioned in the previous section, DPC performs well in identifying local clusters. The reason is that for local clusters, the sample space can be approximately regarded as an Euclidean space, which is exactly what DPC is good at. Therefore, instead of obtaining the final clustering result directly by finding cluster centers and allocating other points, we propose to first construct local clusters in the form of tree structures, known as *family trees*.

The idea to construct family trees for a data set is to find the parent for each data point in its local proximity in the sense of k nearest neighbors, and to identify the ones without parent as the root nodes of trees. To be more specific, let $\mathsf{KNN}(x_i)$ denote the set of k nearest neighbors of a data point x_i (excluding x_i itself). The density of a data point x_i, denoted by $\rho(x_i)$, is given by

$$\rho(x_i) = \sum_{j=1}^{k} \exp(-\mathsf{dist}_{ij}^2), \tag{8}$$

where dist is an $N \times k$ matrix and dist_{ij} is the Euclidean distance between x_i and its j-th nearest neighbor. In this article, dist is also normalized by $\mathsf{dist} \leftarrow \mathsf{dist}/\max(\mathsf{dist})$ to reduce rounding errors. Compared with the definition of density in DPC (cf. Eq. 3), by exploiting k nearest neighbors, we avoid the cutoff distance d_c that is hard to determine and sensitive to specific distributions of data.

The *parent* $P(x_i)$ of a data point x_i can then be defined and the family trees of a data set will be automatically constructed with this information for all x_i.

Definition 1. The *parent* $P(x_i)$ is determined by

$$P(x_i) = \begin{cases} \underset{x_j \in \mathsf{higher}(x_i)}{\arg\min} \ \mathsf{dist}_{ij} & \text{if } \mathsf{higher}(x_i) \neq \emptyset \\ \mathsf{None} & \text{otherwise} \end{cases}, \tag{9}$$

where $\mathsf{higher}(x_i) = \{x_j | x_j \in \mathsf{KNN}(x_i), \rho(x_j) > \rho(x_i)\}$.

In other words, if there are points with higher density in the k nearest neighbors of x_i, then the nearest point x_j is selected as the *parent* of x_i (if there are multiple such x_j, we will select the first met one); otherwise, x_i has no parent ($P(x_i) = \mathsf{None}$), and x_i is defined as a root and added to the set root. For any root $r_i \in \mathsf{root}$, we will use $T(r_i)$ to refer to the family tree with r_i as its root.

Definition 2 (Family tree). A *family tree* is a tree with some data points x_i as its nodes, and has a directed edge (x_i, x_j) if $x_j = P(x_i)$.

Fig. 3 is an illustration of the family trees constructed for the data set Aggregation (see Section 5 for details) with the number of nearest neighbors $k = 20$. The data points with red boundary are root nodes. This data set has 788 data points while there are only 17 family trees. From the figure, it is also worth noting that the local clusters given by these

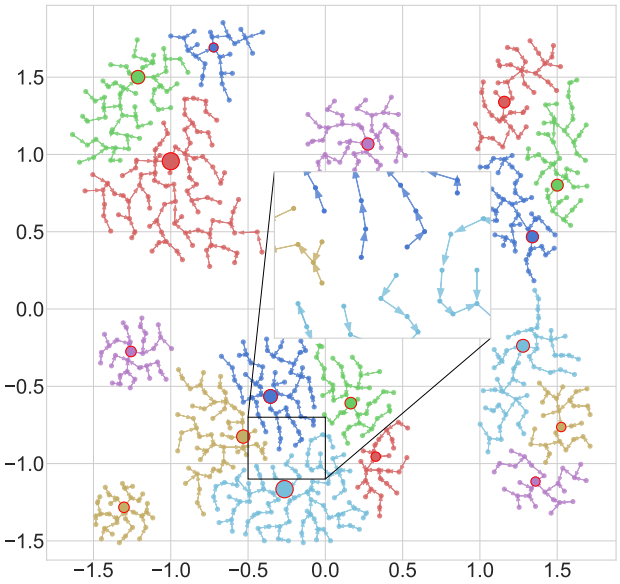

Figure 3: Illustration of family trees.

family trees have high purity, that is, the data points in the same tree are usually also in the same ground-truth cluster. A process to construct family trees for a data set is given in Algorithm 1.

Finally, the following theorem characterizes the correctness of Algorithm 1 and the representation power of family trees.

Theorem 1. *Given k the number of nearest neighbors and X a finite data set to be considered, the returned P by Algorithm 1 forms a set of family trees as defined in Definition 2, and each of the data points in X is in exactly one of the family trees.*

Proof. To show that P returned by Algorithm 1 forms a set of family trees, it is enough to show that it forms a set of trees and each tree is a rooted tree. In fact, there is no cycle path in the graph induced by the parent-child relation P, so it forms a set of trees. Assume on the contrary that there is a cycle path $x_{i_1}, \ldots, x_{i_s}, x_{i_1}$. Then by the construction of P, $\rho(x_{i_j}) > \rho(x_{i_{j+1}})$ for $1 \leq j \leq s$, which implies $\rho(x_{i_1}) < \rho(x_{i_s})$, and $\rho(x_{i_s}) < \rho(x_{i_1})$, which is a contradiction. In addition, each data point x_i has either 1 or 0 out-edge, and each tree has exactly one x_i with 0 out-edge, because otherwise there will be a data point having at least 2 out-edges. This means each tree is a rooted tree.

To see that each data point x_i is in exactly one of the family trees, note that each family tree is a connected component if the edges are taken as undirected. If a data point is in two trees, then these two trees will be in a same connected component, which is not possible. □

In other words, any finite data set can be represented as a unique set of family trees and such a set forms a partition of the data set. In practice, as we will see from the experimental results, this kind of partitions conform well with the ground-truth clusters of a data set, and the number of trees is much smaller than the number of data points. To obtain the final

Algorithm 1: FamilyTree

Input: A data set X; the number of nearest neighbors k.
Output: Family trees represented by a parent-child relation P and a set of root nodes root.

```
foreach x_i ∈ X do
    Calculate KNN(x_i) and the distance matrix dist;
    Calculate ρ(x_i);
root ← ∅;
foreach x_i ∈ X do
    Compute P(x_i) according to Eq. 9;
    if P(x_i) = None then
        root ← root ∪ {x_i};
return P, root.
```

clustering result, we then need to combine the trees into larger clusters in a way it considers both the root nodes and the structure of the trees, which will be discussed subsequently.

4.2. Family Tree Aggregation

In order to aggregate family trees into final clusters, it is necessary to consider not only the roots of trees but also the characteristics of the tree structure. To this end, the strategy of graph cut through spectral clustering is adopted, by considering each tree as a new data point, and devising a novel similarity measure between trees and a more appropriate loss function. The general steps of tree aggregation involve constructing an adjacency matrix, conducting graph cut to extract abstract features for each tree, and finally performing a K-means clustering on the new features to get the aggregation result of trees. In the following, we will explain each of the steps in details.

4.2.1. Similarity Measure for Family Trees

An adjacency matrix for family trees represents the similarity between each pair of them. A trivial solution is to calculate the distance between each pair of roots, which however, cannot correctly reflects the similarity between trees, as two roots might be close to each other but the trees as a whole are not. Therefore, we propose a more sophisticated similarity measure for trees.

First, the *average distance* between trees is worth considering, which can reflect the relative distance between trees to some extent.

Definition 3. The *average l-distance* between two trees $T(r_i)$ and $T(r_j)$ is

$$d_l(T(r_i), T(r_j)) = \frac{\sum_{t=1}^{l} \delta_t}{l}, \tag{10}$$

where $(\delta_1, \ldots, \delta_s)$ is an ascendingly sorted list of all $d(a, b)$ for $a \in T(r_i)$ and $b \in T(r_j)$, $d(a, b) = \|a - b\|_2$, l is a hyper-parameter, and if $s < l$ then l is directly set as s.

The structural information of a tree itself can also be interesting, and here we consider a measure that reflects the looseness of the distribution of data points in a tree. The reason is that when considering the distance between two trees, we need to take into account the internal distance for data points in each of the trees. For example, a distance δ between two trees in a case where the internal distance for them is much smaller than δ would be more significant than a case where that is larger than or similar to δ. Consider Fig. 4 as an example. Although the gap between the tree A and the tree B is larger than the gap between A and C, intuitively A is more similar to B than to C.

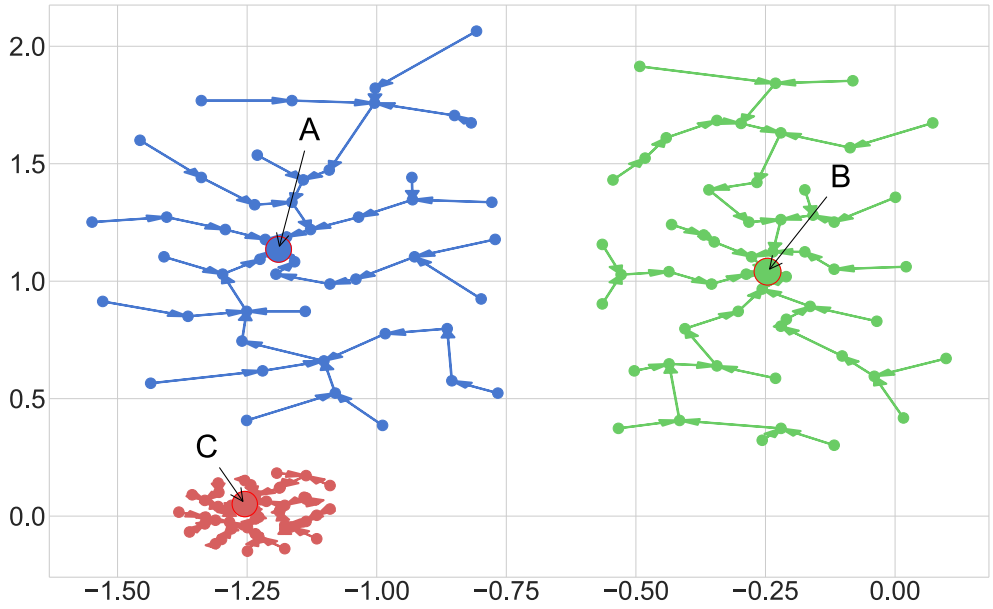

Figure 4: Illustration of the effect of looseness of trees on their similarities.

Definition 4. The *looseness of a tree* $T(r_i)$ is measured by

$$\iota(T(r_i)) = \sum_{x_j \in T(r_i)} g(x_j, r_i) \cdot \omega(x_j, r_i), \tag{11}$$

where $g(x_j, r) = 0$ if $T(r) = \{x_j\}$, and $g(x_j, r) = \min\{d(x_j, x_{j'}) | x_{j'} \neq x_j, x_{j'} \in T(r)\}$ otherwise; $\omega(x_j, r) = \rho(x_j) / \sum_{x_{j'} \in T(r)} \rho(x_{j'})$.

This measure of looseness can be considered as the sum of the distances between x_j and its nearest neighbor in the tree for all x_j, weighted by the relative density of x_j in the tree.

Then, a measure of *separation* between two trees can be proposed as follows.

Definition 5. The *separation* between two trees $T(r_i)$ and $T(r_j)$ can be measured by $\sigma(T(r_i), T(r_j)) = d_l(T(r_i), T(r_j)) / \sqrt{\iota(T(r_i)) \cdot \iota(T(r_j)) + \epsilon}$, where ϵ is a small number to avoid division by zero.

It is easy to verify in Fig. 4 that this separation measure conforms to intuition. In fact, when calculating the separation degrees for trees A, B and C, we get $d_l(A,B) > d_l(A,C)$ and $\iota(A) \approx \iota(B) \gg \iota(C)$, which leads to $\sigma(A,B) < \sigma(A,C)$.

Finally, we would also consider the number of *shared nearest neighbors* of two trees as a contribution to the similarity measure of trees, which can be helpful for processing data with manifold structure.

Definition 6. The *shared nearest neighbors* of trees $T(r_i)$ and $T(r_j)$ $\mathsf{SNN}(T(r_i),T(r_j)) = A(T(r_i)) \cap A(T(r_j))$, where $A(T(r_i))$ is the *affiliate set* of $T(r_i)$, which is $\bigcup_{x_j \in \mathrm{T}(r_i)} \mathsf{KNN}(x_j)$, i.e., the set of k nearest neighbors of data points in the tree.

Note that this definition of shared nearest neighbors is generalized from [19]. Particularly, the above definition is about the shared nearest neighbors of trees, instead of points as in [19]. Fig. 5 shows the shared nearest neighbor connections with the number of nearest neighbors being 5 on the Jain data set, where a red line connects two roots r_i and r_j if $|\mathsf{SNN}(T(r_i),T(r_j))| > 0$. It can be seen from this example that this measure respects the manifold structure of the data.

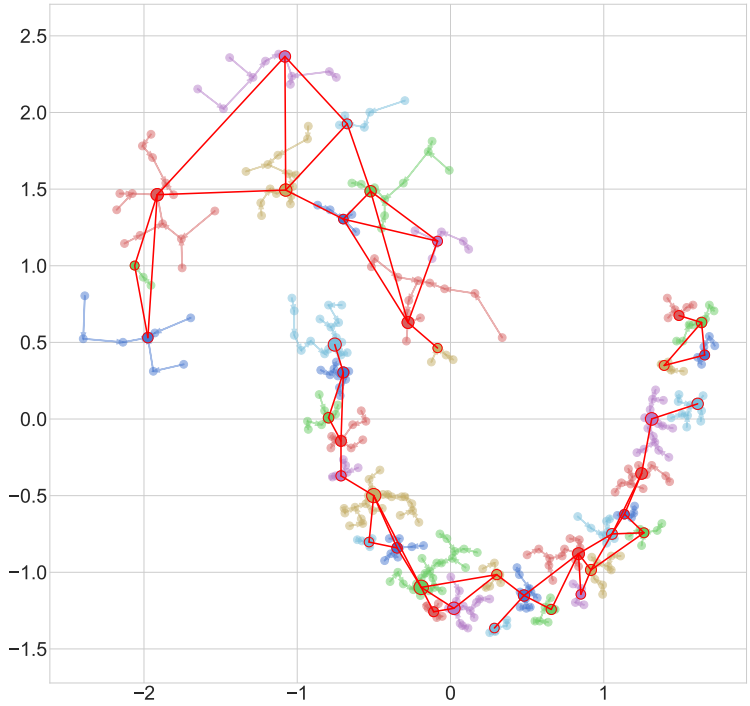

Figure 5: Illustration of shared nearest neighbor connections between trees.

As a result, the *similarity* between two trees can be obtained by synthesizing all of the above.

Definition 7. The *similarity* between two trees $T(r_i)$ and $T(r_j)$ is given by

$$\mathsf{sim}(T(r_i),T(r_j)) = \frac{|\mathsf{SNN}(T(r_i),T(r_j))|}{(1+\sigma(T(r_i),T(r_j))) \cdot (1+d(r_i,r_j))}, \tag{12}$$

where $d(r_i, r_j) = \|r_i - r_j\|_2$.

The above similarity measure for two trees is determined together by the number of shared nearest neighbors, the separation measure between them, and the distance between roots. Although this measure makes use of shared nearest neighbors, it is significantly different from the similarity measure defined in [19]. Specifically, in addition to the difference in the use of shared nearest neighbors, we also take the distance between roots and the separation between trees into consideration. Moreover, the above similarity measure will be used for graph cut, whereas that in [19] is used for the calculation of densities, as a subroutine for a procedure similar to DPC.

For two trees $T(r_i)$ and $T(r_j)$, when their number of shared nearest neighbors is non-zero and fixed, if the densities of them have large difference (i.e., the separation degree is high), or if the distance between the roots is large, then the similarity will be low. Otherwise, the separation degree $\sigma(T(r_i), T(r_j))$ will be a balance against the distance between roots $d(r_i, r_j)$: even when the distance is relatively large, if the separation degree is relatively low, then the similarity measure can still be relatively large. If both the separation degree and the distance between roots are close to zero (i.e., two trees are very similar), then the similarity value will be close to $|\mathsf{SNN}(T(r_i), T(r_j))|$. Finally, when two trees have similar densities and similar distances between roots, then the similarity value will be mainly determined by the number of shared nearest neighbors. It can be easily seen that the value of this similarity measure is between 0 and m (m is the number of data points in a data set).

The adjacency matrix W for the family trees of a data set is then calculated as $W_{ij} = \mathsf{sim}(T(r_i), T(r_j))$ and normalized by $W_{ij} = W_{ij}/\max(W)$. Note that after normalization, $W_{ij} \leq 1$, and that the similarity between a tree and itself should be the highest. We thus set each W_{ii} to be 1. It is worth noting that the connection graph induced by the adjacency matrix (i.e., an edge exists if the corresponding entry in the matrix is nonzero) might not be connected (see Fig. 5 for example), and the number of connected components could be larger than the number of target clusters, which is not desirable [11]. Therefore, we construct a Gaussian kernel distance matrix W' between roots, which is ${W_{ij}}' = \exp(-\frac{d(r_i,r_i)^2}{2})$, and update W to be $W + \theta W'$. Because W' is added only to maintain the connectivity of the whole graph, usually θ is set to a small value, e.g., 0.001.

4.2.2. Graph Cut and Clustering of Trees

The idea to perform graph cut on the connection graph of trees is similar to that of spectral clustering. To simplify notations, in the connection graph, we use root r to represent the tree $T(r)$. Suppose $\mathcal{T}$ is the set of family trees ($|\mathcal{T}| = p$) and the number of target clusters is C. We adapt the original Ncut loss function (Eq. 4) for trees as follows.

$$\mathrm{Ntcut}\left(A_1, \ldots, A_C\right) = \sum_{i=1}^{C} \frac{\mathrm{cut}\left(A_i, \overline{A}_i\right)}{\mathrm{vol}\left(A_i\right) \cdot \sum_{r \in A_i} |T(r)|}, \tag{13}$$

where $A_i = \{r_{i_1}, \ldots, r_{i_s}\}$ is a set of family trees ($\bigcup_i A_i = \mathcal{T}$), $\mathrm{cut}(A_i, \overline{A_i}) = \frac{1}{2}\sum_{r_u \in A_i, r_v \in \overline{A_i}} W_{uv}$, and $\mathrm{vol}(A_i) = \sum_{r_u \in A_i, r_j \in \mathcal{T}} W_{uj}$. Note that an important difference between the above equa-

tion and the original Ncut loss function is that we also take the size of trees into consideration, in order to balance the sizes of the final clusters of data points.

To find a solution that minimizes the above loss function, let $h_j = (h_{1j}, h_{2j}, \cdots, h_{pj})^{\mathrm{T}}$, where

$$h_{ij} = \begin{cases} 0 & v_i \notin A_j \\ \frac{1}{\sqrt{\mathrm{vol}(A_j) \cdot \sum_{r \in A_j} |T(r)|}} & v_i \in A_j \end{cases} . \tag{14}$$

Write $U = DD^*$, where D^* and D are diagonal matrices, $D^*_{ii} = |T(r_i)|$, and $D_{ii} = \sum_j w_{ij}$. Similar to the derivation of Eq. 6, the final optimization problem becomes

$$\min_H \mathrm{tr}\left(H^T L H\right) \qquad \text{s.t.} \quad H^T U H = I, \tag{15}$$

where $H = (h_1, \ldots, h_C)$ and $L = D - W$ is the Laplacian matrix. Again, we can convert the above problem to the following one, by setting $F = U^{1/2} H$.

$$\min_F \mathrm{tr}\left(F^T U^{-1/2} L U^{-1/2} F\right) \qquad \text{s.t.} \quad F^T F = I. \tag{16}$$

The above problem can be solved by computing the eigenvectors $t_1, \ldots, t_C$ corresponding to the first C smallest eigenvalues of $U^{-1/2} L U^{-1/2}$.

Following a similar fashion of spectral clustering, let $E = (t_1, \ldots, t_C) \in \mathbb{R}^{p \times C}$ and $Y = (Y_{ij})_{p \times C}$ s.t. $Y_{ij} = E_{ij} / (\sum_{s=1}^{C} E_{is}^2)^{1/2}$. We can take each row of Y as a feature vector of each family tree, and the clustering result of the family trees can then be obtained by applying the K-means clustering algorithm on these feature vectors.

The effectiveness of the revised objective function is visually illustrated in Fig. 6. In these figures, the clustering results of several trees are shown, where the left figure is generated using the original Ncut loss function and the right is from the revised loss function for trees. In the results, each circle represents a tree and the size of a circle is proportional to the size of the tree. It can be seen that with the original Ncut function, there are three trees incorrectly clustered (tagged as "error"), while with the revised function the three errors are corrected. By looking closer, the incorrectly clustered tree at the top has only a few nodes and has relatively low connection weights with other trees, so it is identified as an individual cluster; the other two incorrectly clustered trees at the bottom should be belong to the same cluster but are identified as parts of two separated clusters, because they have relatively high connection weights with other trees; the sizes of the trees are never considered as a decision factor in the original Ncut function.

4.3. The LDP-SC Algorithm and Its Time Complexity

The LDP-SC algorithm is shown in Algorithm 2. The main steps include: 1) constructing family trees by Algorithm 1; 2) calculating the adjacency matrix W for graph cut; 3) solving the graph cut problem to extract new features; 4) conducting K-means clustering on the new features to obtain the final clustering result.

Suppose the number of data points in a data set is m, the dimension of each data point is n, the number of target clusters is C, the number of nearest neighbors is k, and the number of family trees is p.

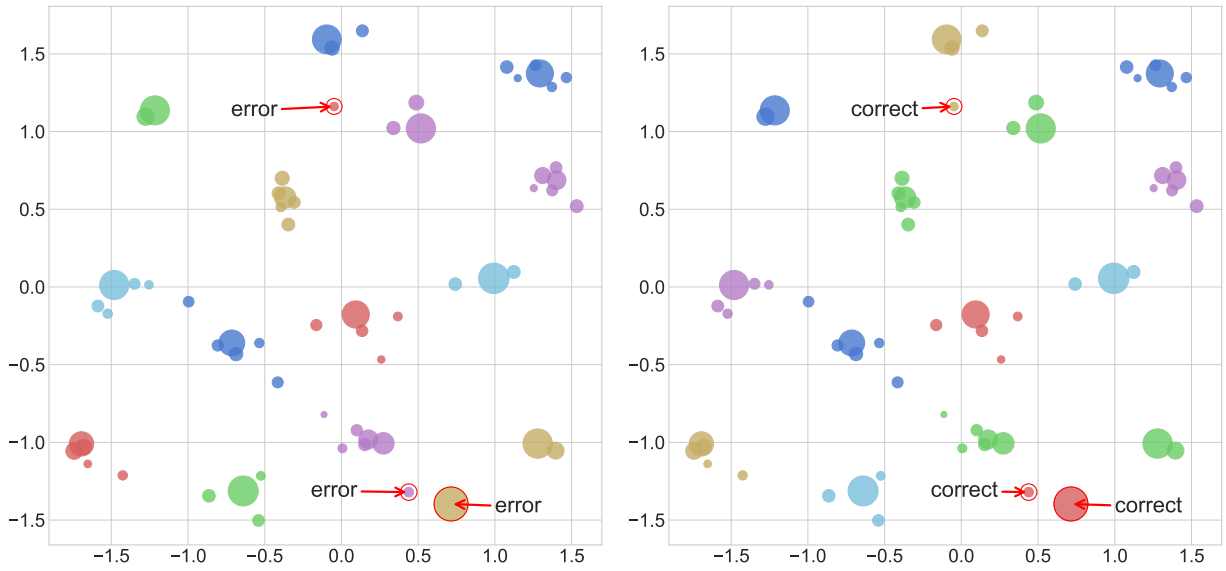

Figure 6: Illustration of the effectiveness of the revised objective function (right), compared with the original loss function (left).

In Algorithm 1, to get the distance matrix dist for the distances between points and their k nearest neighbors, one can make use of a KD-tree and the time complexity is $\mathcal{O}((n+k)m\log m)$. The computation of densities is only related to the k nearest neighbors of each data point, so the time complexity is $\mathcal{O}(km)$. The family trees can then be constructed by calculating $P(x_i)$ for all x_i in a total of time $\mathcal{O}(km)$, as to calculate $P(x_i)$ or to determine root, one needs to check every k nearest neighbor of x_i. So Algorithm 1 requires $\mathcal{O}((n+k)m\log m)$ time.

For the rest of the algorithm LDP-SC, in lines 2-3, when calculating the similarity measure, there are three parts in sequence.

For the part of $|\mathsf{SNN}(T(r_i),T(r_j))|$ for all i,j, one needs first to get the k nearest neighbors of each data point, which takes $\mathcal{O}(km)$ time in total; then the intersections of affiliate sets of each pair of trees can be obtained in time $\mathcal{O}(\sum_{i=1}^{p}|T(r_i)|(p-1)) = \mathcal{O}(pm)$. Therefore, for all $|\mathsf{SNN}(T(r_i),T(r_j))|$, it costs $\mathcal{O}((k+p)m)$ time in total.

For the part of $\sigma(T(r_i),T(r_j))$ for all i,j, one needs $\mathcal{O}(n\cdot|T(r_i)|\cdot|T(r_j)|)$ time to get each $d_l(T(r_i),T(r_j))$, by first calculating $d(a,b)$ for all $a\in T(r_i), b\in T(r_j)$ and then scanning through the list of these values to get the smallest l values. So in total one needs $\mathcal{O}(\sum_{i,j} n\cdot|T(r_i)|\cdot|T(r_j)|) = \mathcal{O}(n\cdot\sum_i|T(r_i)|\cdot m) = \mathcal{O}(nm^2)$ time to get all $d_l(T(r_i),T(r_j))$. To get $\iota(T(r_i))$ for all i, it can be completed in $\mathcal{O}(km)$, as for each $\iota(T(r_i))$ one needs $\mathcal{O}(|T(r_i)|)$ time to get all $\omega(x_j,r_i)$ and $\mathcal{O}(m^2)$ time to get all $g(x_j,r_i)$. So, all $\sigma(T(r_i),T(r_j))$ can be obtained in time $\mathcal{O}(nm^2)$.

The third part $d(r_i,r_j)$ for all i,j can be obtained in a total time of $\mathcal{O}(np^2)$.

Therefore, lines 2-3 in the algorithm LDP-SC take $\mathcal{O}((k+p)m+nm^2+np^2)$ time.

Lines 4-6 take $\mathcal{O}(p^2)$ time.

Lines 7-14 of Algorithm 2 take the same time as a spectral clustering algorithm which is

$\mathcal{O}(p^3)$ [36].

Lines 15-17 can be completed in time $\mathcal{O}(m)$ as one only needs to go through all the data points once.

Algorithm 2: LDP-SC

Input: A data set X; the number of nearest neighbors k; the number of clusters C.
Output: Clustering results label.

1 $P, \mathsf{root} \leftarrow \text{FamilyTree}(X, k)$;
2 **foreach** $(r_i, r_j) \in \mathsf{root} \times \mathsf{root}$ *and* $r_i \neq r_j$ **do**
3 $\quad W_{ij} \leftarrow \mathsf{sim}(T(r_i), T(r_j))$ by Eq. 12;
4 $W \leftarrow W/\max(W)$ and set each W_{ii} to 1;
5 Compute the Gaussian kernel distance matrix W';
6 $W \leftarrow W + \theta W'$;
7 **foreach** $r_i \in \mathsf{root}$ **do** $D_{ii} \leftarrow \sum_j W_{ij}$;
8 $L \leftarrow D - W$;
9 **foreach** $r_i \in \mathsf{root}$ **do** $D^*_{ii} \leftarrow |T(r_i)|$;
10 $U \leftarrow DD^*$;
11 $E \leftarrow (t_1, \ldots, t_C)$, the C eigenvectors of $U^{-1/2}LU^{-1/2}$ corresponding to its C smallest eigenvalues;
12 **foreach** $(r_i, r_j) \in \mathsf{root} \times \mathsf{root}$ **do**
13 $\quad Y_{ij} \leftarrow E_{ij}/(\sum_{s=1}^{C} E_{is}^2)^{1/2}$;
14 Apply K-means on Y to obtain cluster label c_i for each $T(r_i)$;
15 **foreach** $r_i \in \mathsf{root}$ **do**
16 $\quad$ **foreach** $x_i \in T(r_i)$ **do**
17 $\quad\quad \mathsf{label}(x_i) \leftarrow c_i$;
18 **return** label.

In summary, the worst time complexity of the LDP-SC algorithm is $\mathcal{O}((n+k)m \log m + (k+p)m + nm^2 + np^2 + p^3)$ and is bounded by $\mathcal{O}(km \log m + nm^2 + p^3)$. Usually, the value of dimension n is considered to be bounded by a much smaller value than m and the time complexity can be simplified as $\mathcal{O}(m^2 + p^3)$. A comparison of the time complexities of different algorithms is shown in Table 1 (without considering dimension). In practice, the time needed for LDP-SC is usually much lower than the worst case and it is actually quite efficient. For example, LDP-SC takes about 3 seconds for data sets with about 9000 data points (the Pendigits and USPS data sets), 160 seconds for data sets with about 70 000 data points (MNIST and Fashion MNIST), and 3300 seconds for a data set with about 580 000 data points (covertype). As an intuition on the efficiency of LDP-SC compared to others, DPC needs about 140 seconds for Pendigits and USPS, and cannot finish in 5 hours for MNIST, Fashion MNIST, and covertype.

5. Empirical Evaluations

Table 1: Running time complexity comparison.

Algorithm	Date, Publication	Time Complexity
LDP-SC	Ours	$O(m^2 + p^3)$
SNN-DPC	2018, Information Sciences	$O(m^2)$
DPC	2014, Science	$O(m^2)$
LDP-MST	2021, IEEE Transactions on Knowledge and Data Engineering	$O(m \log m)$
FHC-LDP	2021, Neurocomputing	$O(m \log m)$
DPC-DBFN	2020, Pattern Recognition	$O(m^2)$
U-SPEC	2020, IEEE Transactions on Knowledge and Data Engineering	$O(mk(k+C) + p_1^3)^{**}$
LEC-K	2014, IEEE Transactions on Cybernetics	$O(p_1^3 + mp_1^2)$
SC	2000, IEEE Transactions on Pattern Analysis and Machine Intelligence	$O(m^3)$

* m is the number of data points for all the algorithms; p is the number of family trees ($p \ll m$).
** p_1 is the number of landmarks ($k \ll p_1 \ll m$).

5.1. Experiment Settings

To evaluate the performance of the LDC-SC algorithm, we will compare it with several baseline algorithms on both synthetic data sets and real-world data sets. Because the idea of the algorithm is based on local density peaks and graph cut through spectral clustering, the following prominent baseline algorithms are chosen:

1. DPC [15]: the original DPC algorithm.
2. SNN-DPC [19]: it improves density measurement, distance calculation, and allocation strategy of the DPC algorithm.
3. LDP-MST [30]: it improves the DPC algorithm by generating some local tree structures (minimum spanning trees) and constructing connection relations between these trees.
4. FHC-LDP [24]: it improves the DPC algorithm by building sub-clusters with local density structures, and then combining these sub-clusters by hierarchical clustering.
5. DPC-DBFN [27]: it improves the DPC algorithm by exploiting a fuzzy kernel to extract a specific kind of structural information to improve the robustness of clustering.
6. SC [36]: the original spectral clustering algorithm based on Ncut loss function.
7. U-SPEC [33]: an accelerated spectral clustering algorithm using approximated K-nearest neighbors and bipartite graphs.
8. LSC-K [32]: a variant of spectral clustering algorithms by using K-means to select landmarks.

The implementation of SC are from the scikit-learn library, the implementations of SNN-DPC, LDP-MST, FHC-LDP, and DPC-DBFN are from the source code provided by the authors, the implementations of DPC and LSC-K are based on the descriptions of the original paper, and the U-SPEC implementation is from [38].

Three commonly used clustering performance metrics, i.e., Adjusted Rand Index (ARI [39]), Normalized Mutual Information (NMI [40]) and Accuracy (ACC [41]) are selected for performance evaluation. All results are averaged over ten repeated runs of the algorithms.

For fair comparison, the hyper-parameters of each algorithm are tuned beforehand. Specifically, all algorithms use the number of ground-truth clusters as the number of target

clusters. The number k of nearest neighbors for LDP-SC, SNN-DPC, SC, U-SPEC, FHC-LDP, DPC-DBFN, and LSC-K is searched from 2 to 50 with a step size of 1. DPC-DBFN has two ways to calculate densities and we try both and select the best one. For DPC, we search for an optimal value of the ratio of the number of neighbors against the number of all data points, in a loop with a step size of 0.1 from 0.2 to 5. For identifying cluster centers of SNN-DPC and DPC, this article selects the C data points with the first C largest values of γ (cf. Section 3.1), which is more general and fairer compared with manual selection with a decision graph. For LDP-MST, it recommends that data should have dimensionalities lower than 10, so we search for an optimal dimensionality from 2 to $\min(50, n)$ by using PCA. The adjacency matrix of SC is calculated based on k nearest neighbors. U-SPEC has a parameter p and we set the value p as 1 000 if the number of data points $m > 10\,000$ and as $\lfloor (m-1)/10 \rfloor$ otherwise. For LSC-K, the parameter p is set to 500 if $m > 1000$ and to $\lfloor m/2 \rfloor$ otherwise.

For LDP-SC, the parameter l for average l-distance is set as 4, the weight θ of Gaussian kernel distance matrix is set as 0.001, and ϵ in Definition 5 is set as 10^{-6}.

For data sets, we utilize 8 widely used synthetic data sets, 8 real-world data sets, and 6 image data sets. The details of them are shown in Table 2, Table 3, and Table 4. Fig. 7 gives an illustration of the image data sets. Data sets are all preprocessed in certain ways: for non-image data sets, the SNN-DPC and FHC-LDP algorithms apply min-max standardization to the data, and all the other algorithms applies z-score standardization; for image data sets, if the pixel value is in range $[0, 255]$, then it is scaled to $[0, 1]$.

The experimental environment is: Python 3.9, Ubuntu 18.04, CPU i7-8700, and 64GB RAM.

Table 2: Synthetic data sets.

Data set	**#Instances**	**#Attributes**	**#Clusters**
Aggregation	788	2	7
R15	600	2	15
S2	5000	2	15
Flame	240	2	2
Jain	373	2	2
Spiral	312	2	3
happy	266	2	3
circle	299	2	3

5.2. Results and Analysis on Synthetic Data Sets

Table 5 shows the results and the corresponding hyper-parameters, for LDP-SC and the other eight comparison algorithms on the synthetic data sets. The best results are marked in bold.

From Table 5, we can see that LDP-SC has the best or close to the best performance on all of these data sets. On five out of the eight data sets, LDP-SC has 100 scores for all of ARI, NMI, and ACC metrics. The cases where LDP-SC is slightly weaker than the optimal

Table 3: Real-world data sets.

Data set	**#Instances**	**#Attributes**	**#Clusters**
pengleukEW	72	7070	2
parkinsonsEW	195	22	2
seeds	210	7	3
spect	267	22	2
mfeat-fac	2000	216	10
landsatEW	6435	36	6
Pendigits	10992	16	10
convertype	581012	54	7

Table 4: Image data sets.

Data set	**#Instances**	**#Attributes**	**#Clusters**
MNIST	70000	784	10
Fashion MNIST	70000	784	10
COIL20	1440	1024	20
USPS	9298	256	10
Yale	2414	1024	38
ORL	400	4096	40

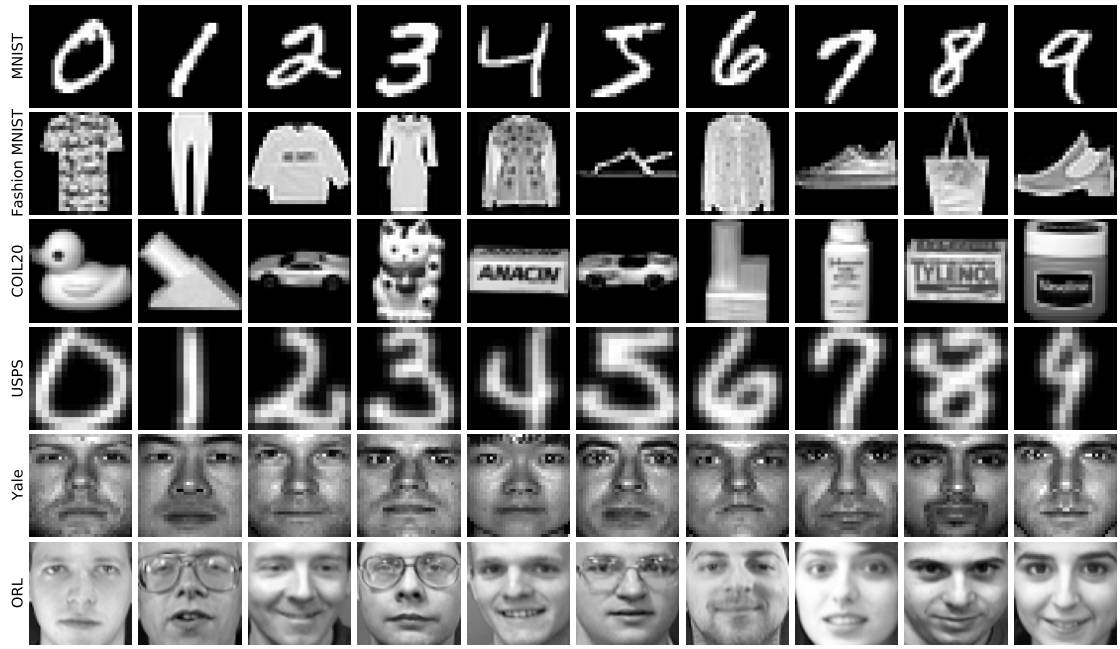

Figure 7: Examples of image data sets.

Table 5: Results on synthetic data set (%).

Data set		LDP-SC	SNN-DPC	DPC	LDP-MST	FHC-LDP	DPC-DBFN	U-SPEC	LEC-K	SC
Aggregation	ARI	99.56	95.94	99.56	99.56	**100**	99.27	90.57	99.20	98.98
	NMI	99.24	95.55	99.24	99.24	**100**	98.83	92.55	98.84	98.51
	ACC	99.75	97.84	99.75	99.75	**100**	99.62	85.41	99.62	99.49
	Par.	7	15	3.5	2	6	30	3	8	14
R15	ARI	99.28	99.28	99.28	98.91	99.28	**99.64**	98.91	99.28	99.28
	NMI	99.42	99.42	99.42	99.13	99.42	**99.71**	99.13	99.42	99.42
	ACC	99.67	99.67	99.67	99.50	99.67	**99.83**	99.50	99.67	99.67
	Par.	6	10	0.2	2	7	39	5	23	35
S2	ARI	92.94	91.91	92.49	91.88	92.86	**93.61**	93.57	93.52	93.41
	NMI	93.76	93.14	93.52	93.54	93.71	**94.29**	94.25	94.19	94.27
	ACC	96.58	96.08	96.36	96.04	96.54	**96.92**	96.90	96.88	96.82
	Par.	34	35	0.5	2	36	4	22	10	47
Flame	ARI	**100**	95.02	**100**	93.39	**100**	96.66	98.33	95.01	91.76
	NMI	**100**	89.94	**100**	87.52	**100**	92.69	96.35	89.91	85.19
	ACC	**100**	98.75	**100**	98.33	**100**	99.17	99.58	98.75	97.92
	Par.	23	5	2.7	2	7	6	4	5	5
Jain	ARI	**100**	59.35	71.46	**100**	**100**	80.08	**100**	**100**	**100**
	NMI	**100**	55.91	65.22	**100**	**100**	71.88	**100**	**100**	**100**
	ACC	**100**	88.74	92.49	**100**	**100**	95.17	**100**	**100**	**100**
	Par.	14	39	0.3	2	7	43	2	9	8
Spiral	ARI	**100**	**100**	**100**	**100**	**100**	14.87	2.64	**100**	**100**
	NMI	**100**	**100**	**100**	**100**	**100**	15.75	2.85	**100**	**100**
	ACC	**100**	**100**	**100**	**100**	**100**	55.45	44.87	**100**	**100**
	Par.	4	5	1.6	2	3	4	2	3	3
happy	ARI	**100**	**100**	71.79	**100**	**100**	52.21	**100**	**100**	**100**
	NMI	**100**	**100**	78.71	**100**	**100**	60.94	**100**	**100**	**100**
	ACC	**100**	**100**	89.85	**100**	**100**	81.58	**100**	**100**	**100**
	Par.	4	5	4.6	2	8	2	2	3	7
circle	ARI	**100**	**100**	21.41	**100**	**100**	46.06	**100**	**100**	**100**
	NMI	**100**	**100**	34.50	**100**	**100**	63.96	**100**	**100**	**100**
	ACC	**100**	**100**	56.86	**100**	**100**	73.91	**100**	**100**	**100**
	Par.	10	33	0.2	2	7	17	2	9	17

ones (FHC-LDP and DPC-DBFN) are the Aggregation, R15, and S2 data sets. It is worth noting that on the data set Jain, the SNN-DPC algorithm performs much worse than in the original paper, because in this article it identifies cluster centers automatically with γ values, whereas in the original paper it was done manually.

We also visualize the results on synthetic data sets in Figs. 8 to 10. For the sake of space limitation, we omit the visualization of the results of DPC, SC, and LSC-K. In all of these figures, the cluster centers of SNN-DPC and DPC-DBFN are represented by pentagrams.

Figs. 8, 9, and 10 show the clustering results on three data sets with high density clusters. SNN-DPC correctly finds the cluster centers and sometimes makes mistakes on borders due to its allocation strategy. Note that U-SPEC sometimes puts two subsets of data points that are far apart into one cluster. On the other hand, LDP-SC correctly separates the boundaries and adapts well for clusters of complex shapes.

Figs. 11 to 15 show clustering results on five data sets with irregular shapes and manifold structure. It can be seen that LDP-SC performs well in all cases. For Fig. 11, all the other algorithms also perform well, except for some errors on the boundary. For Fig. 12, LDP-SC, LDP-MST, FHC-LDP, and U-SPEC all correctly detect the manifold structure; SNN-DPC and DPC-DBFN make mistakes on the boundary; the errors of SNN-DPC and DPC-DBFN

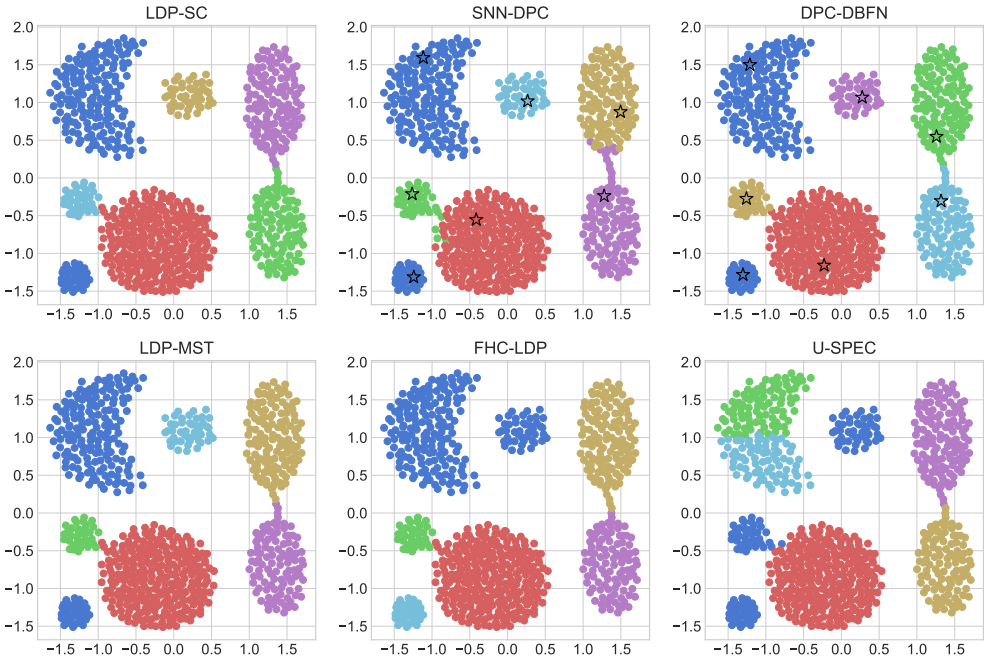

Figure 8: Aggregation

are due to large values for the number of nearest neighbors. For Fig. 13, LDP-SC, SNN-DPC, and LDP-MST perform well, while DPC-DBFN, FHC-LDP, and U-SPEC have poor performances on this data set. For Fig. 14, DPC-DBFN incorrectly includes data points from the curved strip for the clusters in the middle, and for Fig. 15, it identifies a wrong cluster center which makes it perform poorly.

5.3. Results and Analysis on Real-World Data Sets

Table 6 shows the results on eight commonly used real-world data sets, where the numbers of data points, clusters, and features vary in a large range, which allows us to evaluate our algorithm from different aspects. The proposed LDP-SC algorithm has the best performance in terms of all three metrics for five out of the eight data sets. Note that in four out of these five data sets, the performance of LDP-SC is significantly better than the second best. There are two data sets, i.e., spect and covertype, for which LDP-SC has the best scores for some of the three metrics. It is worth noting that for the covertype data set, the results of SNN-DPC, DPC, LDP-MST, FHC-LDP, DPC-DBFN, and SC cannot be obtained (shown as N/A in the table), because they exceed memory or time limit (5 hours). For the seeds data set, LDP-SC is slightly weaker than the best one. These results illustrate the superiority of LDP-SC on complex real-world data.

5.4. Results and Analysis on Image Data Sets

Table 7 shows the results on six commonly used image data sets. Note that for the MNIST and Fashion MNIST data sets, due to large number of data points (70,000), SNN-DPC, FHC-LDP, DPC-DBFN, and DPC exceed memory or time limit and have no results.

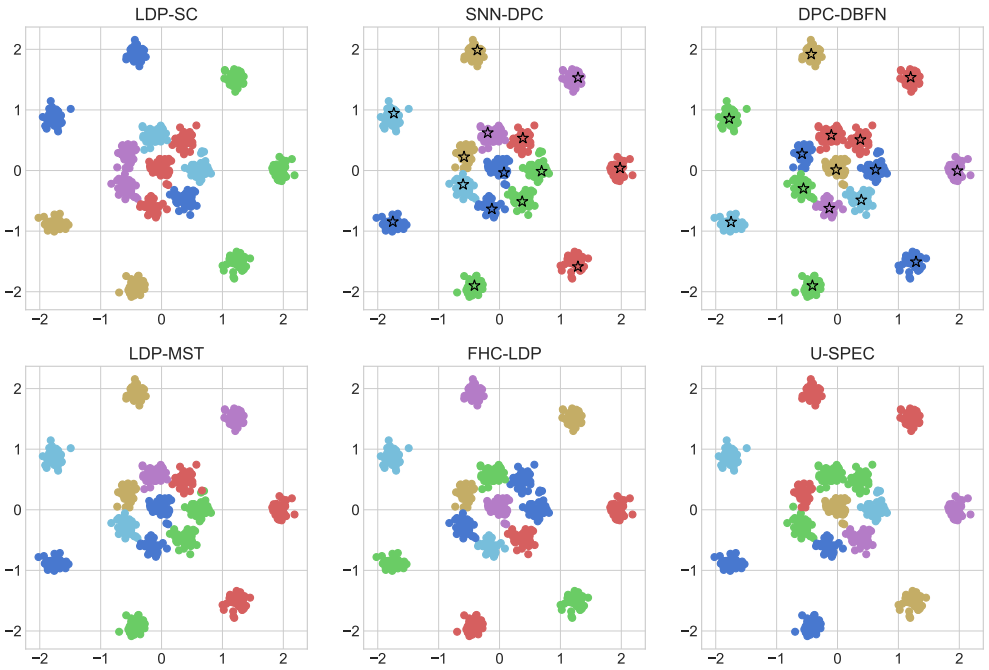

Figure 9: R15

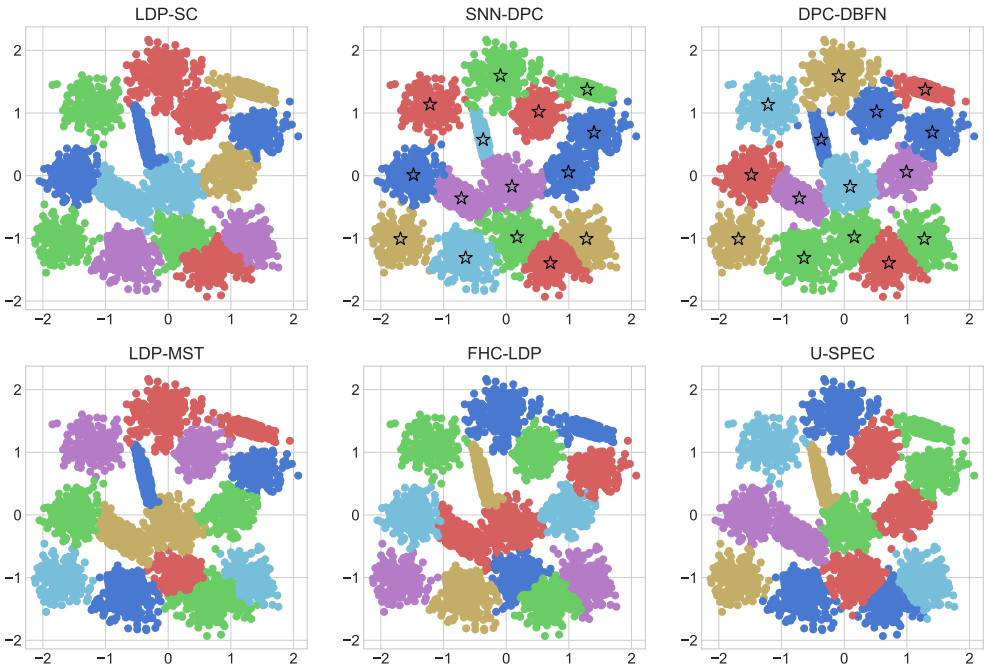

Figure 10: S2

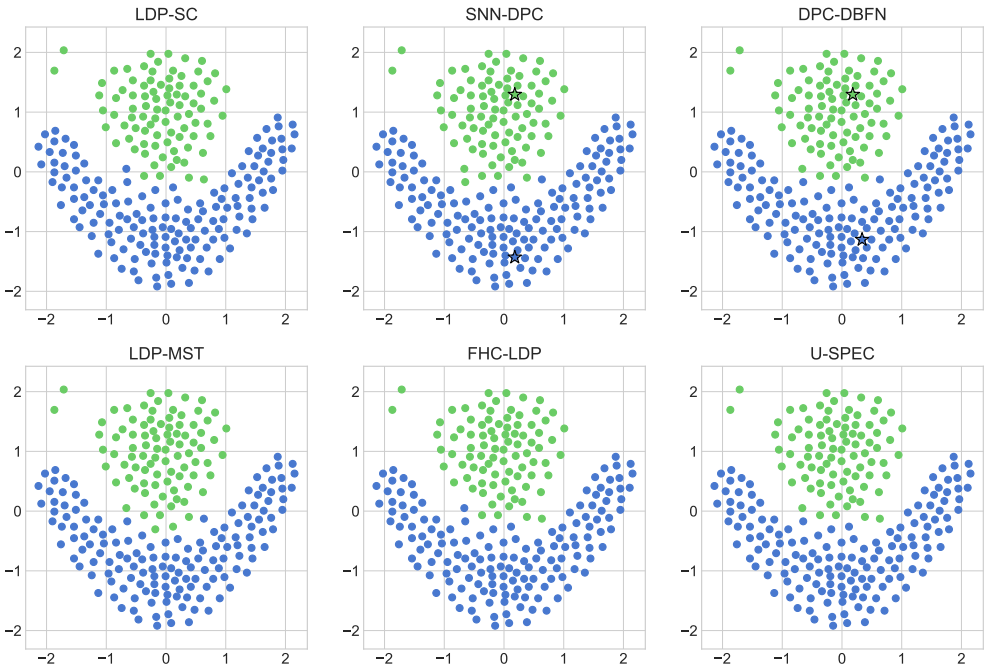

Figure 11: Flame

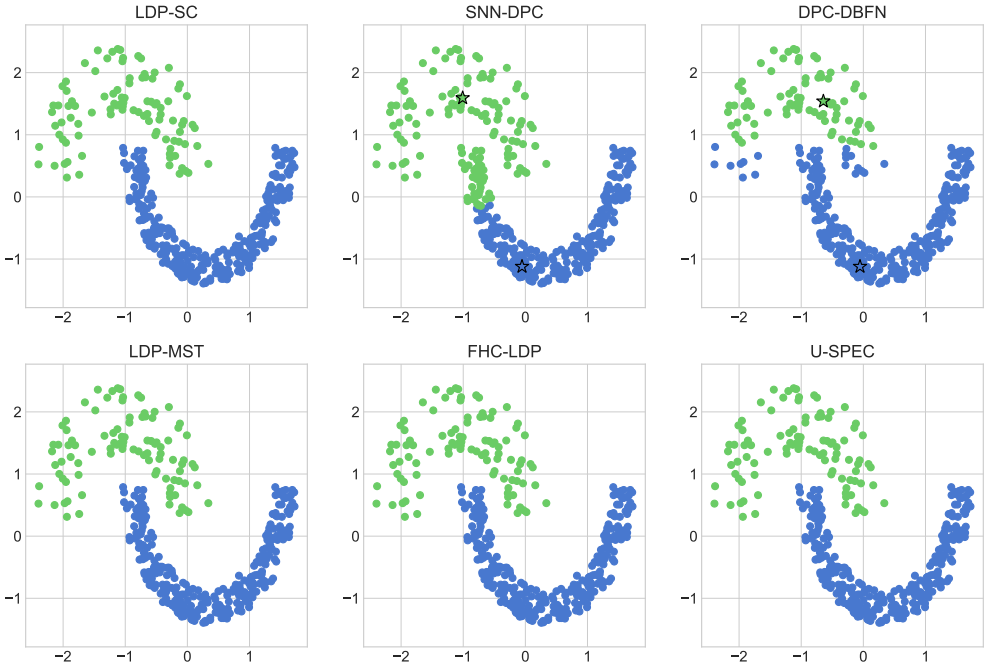

Figure 12: Jain

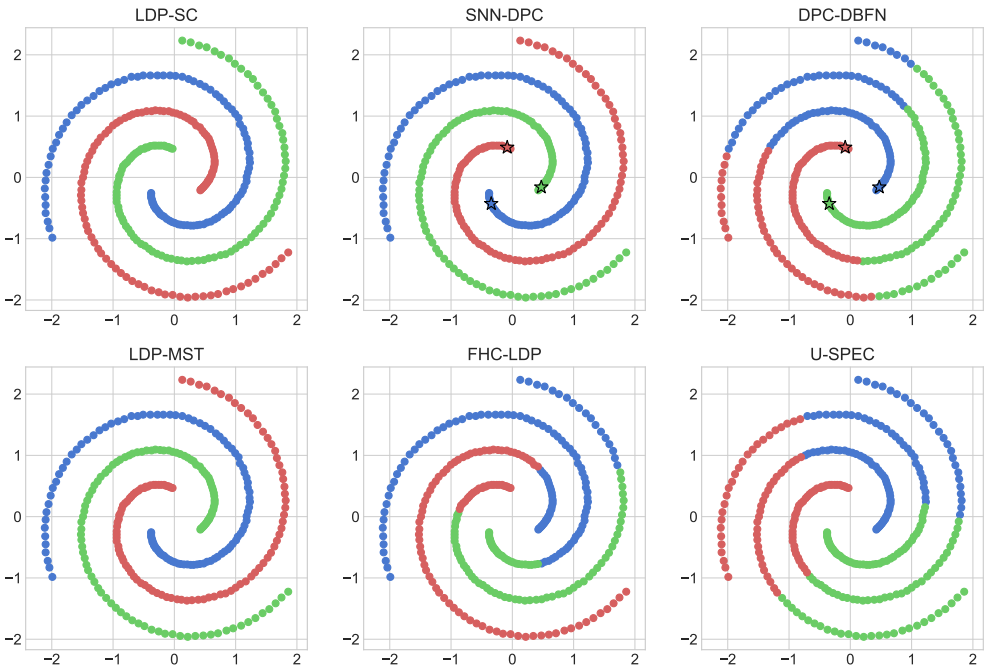

Figure 13: Spiral

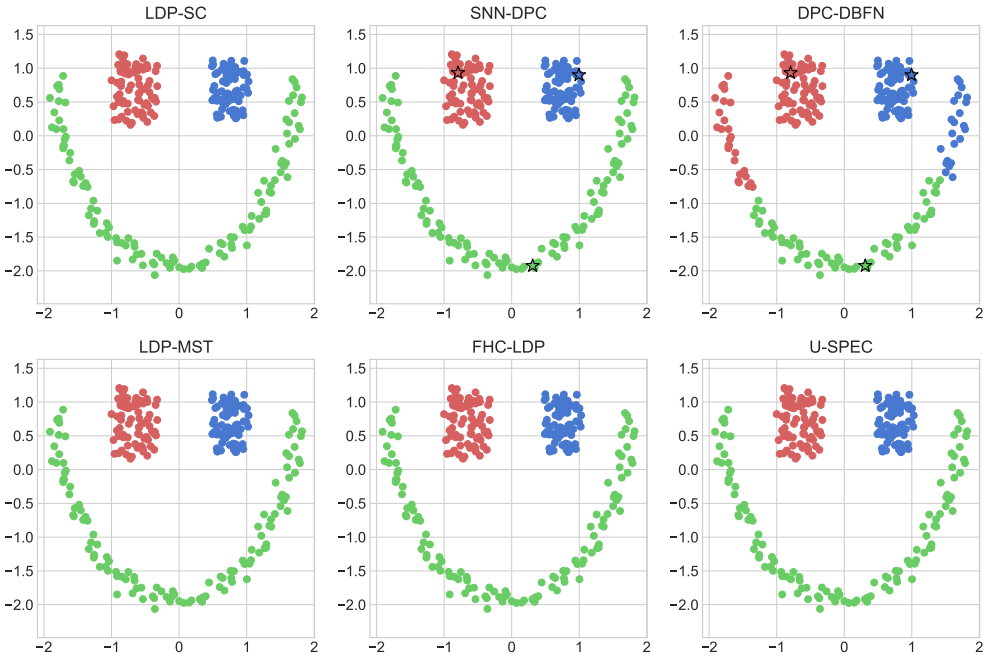

Figure 14: happy

Table 6: Results on real-world data sets (%).

Data set		LDP-SC	SNN-DPC	DPC	LDP-MST	FHC-LDP	DPC-DBFN	U-SPEC	LEC-K	SC
pengleukEW	ARI	**50.41**	36.21	13.48	29.22	38.17	18.30	13.48	29.61	46.46
	NMI	**42.35**	26.38	17.47	19.63	32.50	12.16	17.47	20.94	36.52
	ACC	**86.11**	80.56	72.22	77.78	81.94	72.22	72.22	77.78	84.72
	Par.	4	5	1	26	9	27	3	8	4
parkinsonsEW	ARI	**46.23**	45.21	40.28	41.71	39.10	9.77	-1.80	11.05	39.10
	NMI	**35.30**	31.87	30.38	33.60	35.14	5.15	15.65	11.70	35.14
	ACC	**86.67**	86.15	85.13	85.64	85.13	76.92	56.41	78.00	85.13
	Par.	7	25	0.4	13	5	8	4	2	5
seeds	ARI	78.48	77.76	79.97	57.02	72.89	76.64	**82.31**	82.25	82.18
	NMI	72.42	74.23	75.64	63.07	69.46	73.43	77.3	77.78	**77.86**
	ACC	92.38	91.90	92.86	81.90	90.00	91.43	**93.81**	**93.81**	**93.81**
	Par.	23	6	0.6	5	12	2	2	8	7
spect	ARI	30.62	**35.98**	-1.36	28.83	32.18	32.18	-1.99	-8.29	20.51
	NMI	**20.59**	19.33	0.06	16.36	19.93	19.93	13.03	5.03	12.72
	ACC	**84.64**	83.52	64.42	79.03	**84.64**	**84.64**	55.06	71.16	74.16
	Par.	5	15	0.20	3	23	24	4	6	4
mfeat-fac	ARI	**88.38**	71.86	64.15	55.97	81.69	48.85	83.48	85.01	85.90
	NMI	**89.13**	80.05	78.14	69.51	85.58	69.16	85.65	86.46	87.57
	ACC	**94.60**	78.40	69.95	63.65	90.95	59.25	92	92.88	93.25
	Par.	30	35	0.30	38	29	17	3	7	13
landsatEW	ARI	**62.80**	52.56	54.79	50.01	0.25	16.92	59.50	52.58	45.13
	NMI	**65.02**	57.80	58.24	59.83	1.64	30.40	63.61	62.83	59.04
	ACC	**76.02**	69.43	75.04	64.21	24.49	42.46	72.79	68.38	64.55
	Par.	6	8	0.20	6	3	2	6	3	7
Pendigits	ARI	**80.04**	64.17	65.10	70.11	68.36	54.50	71.93	64.31	77.67
	NMI	**86.31**	78.54	76.10	81.66	82.20	69.79	80.22	79.72	84.25
	ACC	**89.30**	74.15	76.72	78.02	79.08	69.92	84.95	74.65	88.05
	Par.	12	42	0.9	7	48	17	27	5	25
covertype	ARI	**16.88**	N/A	N/A	N/A	N/A	N/A	14.42	8.06	N/A
	NMI	**18.24**	N/A	N/A	N/A	N/A	N/A	12.21	12.12	N/A
	ACC	44.39	N/A	N/A	N/A	N/A	N/A	47.04	**48.33**	N/A
	Par.	21	-	-	-	-	-	3	6	-

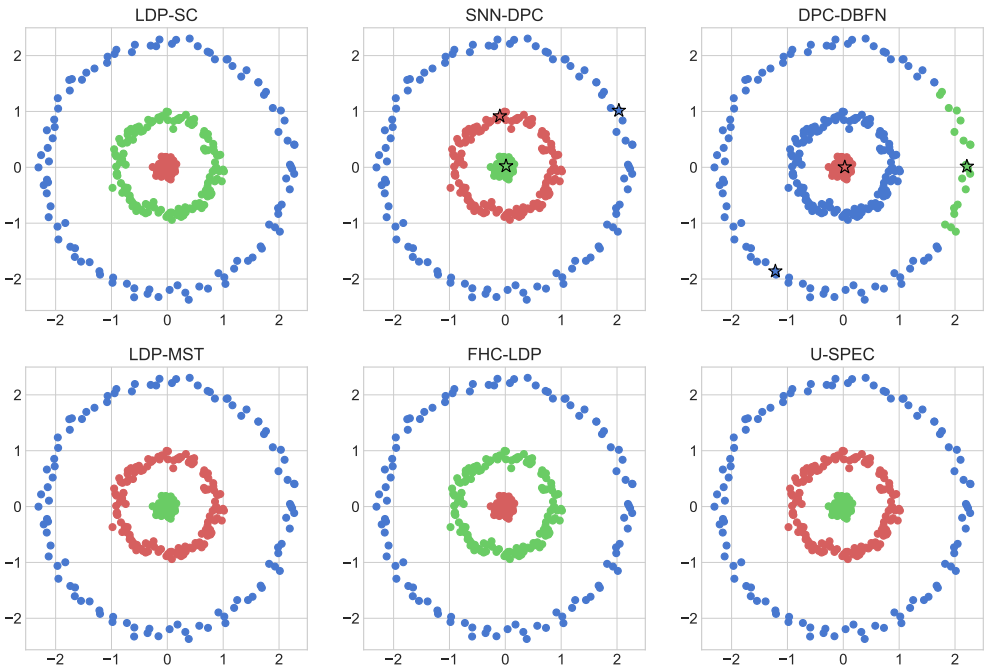

Figure 15: circle

For the Yale, COIL20, USPS, and MNIST data sets, LDP-SC has the best performance for all three metrics. For ORL, LDP-SC has the highest ARI and NMI scores, and has a only slightly lower ACC score than the best one. For Fashion MNIST data set, although LDP-SC is not the best, its performance is very close to the best. Again, for the data sets where LDP-SC performs best, the scores of it are significantly higher than the others, especially for the USPS data set. These results also show that LDP-SC is very promising in high dimensional and complex real-world data.

6. Discussion

6.1. Robustness

In the previous section, we discussed the optimal results of each algorithm with known data labels. However, unsupervised clustering can only be performed without labels in practice. Therefore, a good clustering algorithm should also be robust w.r.t. different values of parameters, so that its performance remains good enough without tuning parameters based on labels. Note that LDP-SC constructs a parent-child relationship by looking for the data point with higher density that is closest to a given data point. This relationship is expected to be insensitive to the only parameter k, i.e., the number of nearest neighbors: for a non-root point, when k changes within a reasonable range, its parent is likely to be the same one as before (because its parent is close to it and the relative ordering of their densities is likely to be unchanged); for a root point, k would need to become quite large to have a higher density point. To verify this empirically, Fig. 16 shows the fluctuation of ARI

Table 7: Results on image data sets (%).

Data sets		LDP-SC	SNN-DPC	DPC	LDP-MST	FHC-LDP	DPC-DBFN	U-SPEC	LEC-K	SC
ORL	ARI	**52.76**	31.36	38.59	28.69	37.33	28.87	31.66	45.96	49.66
	NMI	**83.72**	76.26	77.13	67.52	79.59	67.69	70.06	79.89	82.35
	ACC	65.00	52.50	50.50	49.2	58.75	47.25	45.50	63.05	**66.50**
	Par.	4	5	0.4	18	5	28	2	3	5
Yale	ARI	**14.17**	4.06	6.24	2.09	6.44	4.40	4.40	4.56	9.50
	NMI	**47.04**	36.00	37.83	12.46	37.65	31.50	20.81	26.34	41.74
	ACC	**33.51**	24.15	28.38	8.53	28.54	24.19	15.86	19.61	29.25
	Par.	3	6	3	42	9	2	2	2	3
COIL20	ARI	**84.33**	61.68	60.86	69.54	79.17	54.51	76.74	82.56	71.11
	NMI	**96.30**	82.07	84.19	89.42	92.76	75.59	89.69	95.38	86.24
	ACC	**87.29**	69.44	69.65	77.29	83.47	61.75	80.94	85.00	79.38
	Par.	3	41	2.6	13	9	26	2	3	13
USPS	ARI	**91.91**	61.82	35.65	80.29	68.91	51.16	80.72	75.20	68.02
	NMI	**90.03**	76.94	52.29	81.95	77.44	64.70	82.45	83.05	82.21
	ACC	**95.81**	66.41	46.86	86.47	73.13	63.49	84.76	78.92	68.36
	Par.	4	16	4.2	28	15	4	5	2	4
MNIST	ARI	**75.75**	N/A	N/A	62.66	N/A	N/A	64.95	63.16	62.95
	NMI	**83.11**	N/A	N/A	69.58	N/A	N/A	74.35	74.27	76.47
	ACC	**79.75**	N/A	N/A	68.57	N/A	N/A	74.57	73.72	68.31
	Par.	8	-	-	21	-	-	2	2	6
Fashion MNIST	ARI	46.92	N/A	N/A	40.36	N/A	N/A	**47.01**	40.16	46.8
	NMI	62.96	N/A	N/A	58.06	N/A	N/A	62.54	56.22	**65.89**
	ACC	58.69	N/A	N/A	47.08	N/A	N/A	**58.85**	58.13	58.33
	Par.	9	-	-	11	-	-	3	35	4

values of LDP-SC, LSC-K, SNN-DPC, U-SPEC, FHC-LDP and DPC-DBFN w.r.t. k on the data sets spect, seeds, mfeat-fac and Pendigits. For DPC based algorithms, $d_c = k/10$. Note that U-SPEC performs clustering on the specified core points, which might be smaller than the given k and thus makes the algorithm have no result in that case (spect and seeds data sets). From the results, it can be seen that LDP-SC algorithm is very stable w.r.t. changes of k and can be considered as one of the most stable ones among the six algorithms.

6.2. Ablation Study

In order to verify the significance of each component in the proposed algorithm, an ablation study was conducted here on seven real-world data sets. The results are optimal by searching for the best parameters, and are shown in Table 8, where columns a, b, c, d represents the following different settings, respectively:

a. Graph cut is changed to be on the roots only, using the original spectral clustering algorithm. This is to verify the significance of the tree structures and the corresponding similarity measure (Eq. 12) considered in this article.

b. The term of separation degree $(1 + \sigma(T(r_i), T(r_j)))$ in similarity measure (Eq. 12) is removed. This is to verify that separation degree is useful for complex structures of real-world data.

c. The term $|\mathsf{SNN}(T(r_i), T(r_j))|$ in the similarity measure is replaced with the constant 1. This is to verify that the neighboring information of trees is useful for clustering.

d. The total number of nodes in trees is removed from the loss function Ntcut (Eq. 13). This is to verify the significance of tree sizes for clustering.

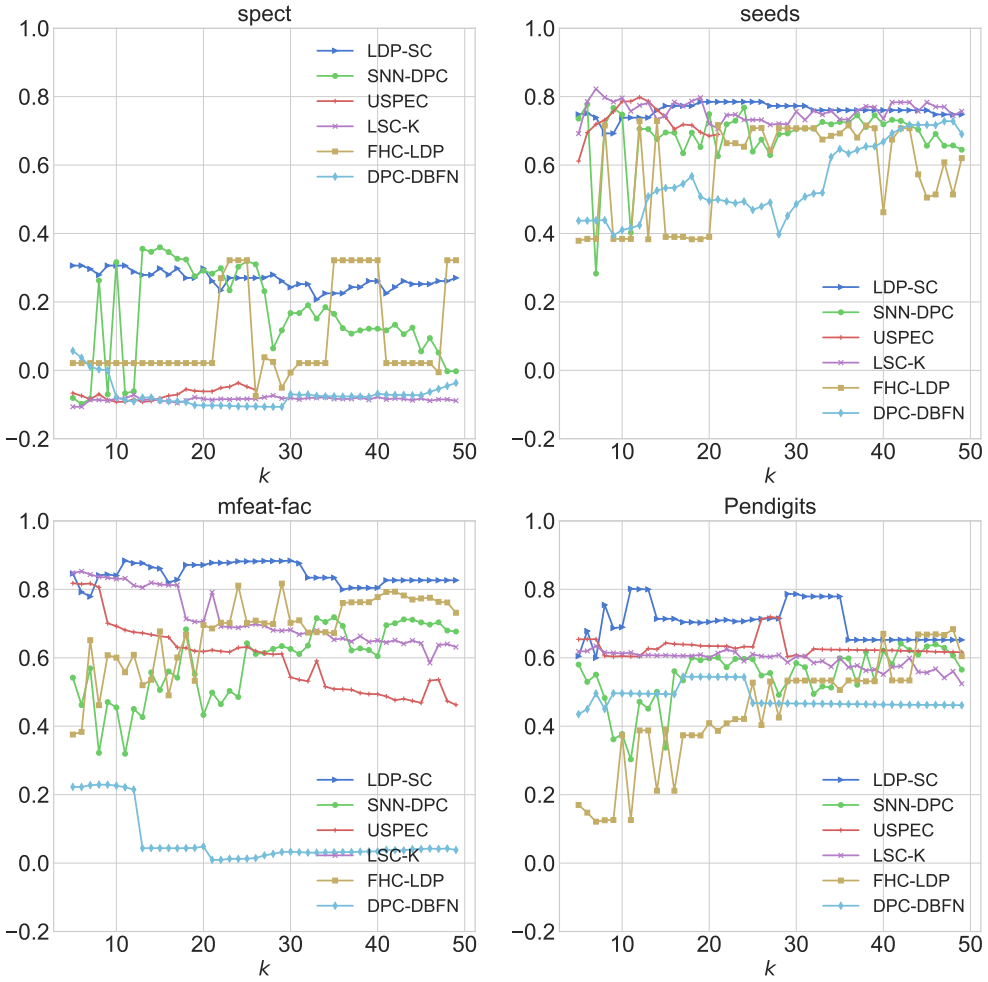

Figure 16: Performance changes with respect to the number of nearest neighbors k.

According to Table 8, as there will be a decrease in performance when a component is removed or changed in most data sets, it can be seen that each component of the proposed algorithm is useful.

7. Conclusion

This article proposes a new clustering algorithm based on the idea of local density peaks and graph cut. In particular, this algorithm takes the advantages of the DPC algorithm in local neighborhoods to capture latent structures of data to construct family trees, devises a novel similarity measure between family trees, and adapts the Ncut objective function to perform graph cut on the connection graph of these trees. In this way, a data set can be clustered by allocating data points in a family tree to the cluster of the tree. We performed a variety of empirical evaluations of the proposed algorithm on 22 challenging synthetic data sets and real-world data sets. The results demonstrated the superiority of our algorithm over

Table 8: Results of ablation study (in ARI, %).

Data set	LDP-SC	a	b	c	d
pengleukEW	**50.41**	0	**50.41**	13.76	**50.41**
spect	**30.62**	16.19	23.07	**30.62**	**30.62**
landsatEW	**62.80**	36.43	60.04	51.48	58.34
mfeat-fac	**88.38**	73.87	**88.38**	84.78	80.88
parkinsonsEW	**46.23**	39.10	39.10	39.10	**46.23**
Pendigits	**80.04**	56.23	79.16	72.68	79.50
seeds	**78.48**	**78.48**	**78.48**	**78.48**	**78.48**

several prominent clustering algorithms, in terms of three commonly used metrics (ARI, NMI, and ACC). The analysis of its robustness w.r.t. hyper-parameters shows that the proposed algorithm has stable performance with a large range of the hyper-parameter k, the number of nearest neighbors. This new algorithm is also very efficient compared to several algorithms including SNN-DPC, LDP-MST, DPC-DBFN, FHC-LDP, DPC, and SC on large data sets, and is competitive compared to accelerated spectral cluster algorithms. Ablation study also proves that each of its components has a contribution to the overall performance of the algorithm. Nevertheless, the current algorithm still has some defects, e.g., the efficiency of the algorithm can be further improved, the number of target clusters can only be determined manually, and the ability to detect and remove noise data is still missing. All of these will be interesting to explore in the future.

Acknowledgment

This work was supported by the National Natural Science Foundation of China [grant number 61806170]; the Humanities and Social Sciences Fund of Ministry of Education [grant number 18XJC72040001]; and the National Key Research and Development Program of China [grant number 2019YFB1706104].

References

[1] A. K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognition Letters 31 (2010) 651–666.
[2] J. A. Hartigan, Clustering algorithms, John Wiley & Sons, Inc., 1975.
[3] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Advances in Neural Information Processing Systems, 2002, pp. 849–856.
[4] B. J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972–976.
[5] U. von Luxburg, R. C. Williamson, I. Guyon, Clustering: Science or art?, in: International Conference on Machine Learning: Workshop on Unsupervised and Transfer Learning, 2012, pp. 65–79.
[6] J. MacQueen, others, Some methods for classification and analysis of multivariate observations, in: Berkeley Symposium on Mathematical Statistics and Probability, volume 1, 1967, pp. 281–297.
[7] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: An efficient data clustering method for very large databases, volume 25, 1996, pp. 103–114.
[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: International Conference on Knowledge Discovery and Data Mining, volume 96, 1996, pp. 226–231.

[9] D. Alahakoon, S. K. Halgamuge, B. Srinivasan, Dynamic self-organizing maps with controlled growth for knowledge discovery, IEEE Transactions on Neural Networks 11 (2000) 601–614.
[10] W. Wang, J. Yang, R. Muntz, STING: A statistical information grid approach to spatial data mining, in: International Conference on Very Large Data Bases, volume 97, 1997, pp. 186–195.
[11] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (2007) 395–416.
[12] C. Xu, D. Tao, C. Xu, A survey on multi-view learning, arXiv preprint arXiv:1304.5634 (2013).
[13] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, Journal of Machine Learning Research 3 (2002) 583–617.
[14] J. R. Hershey, Z. Chen, J. Le Roux, S. Watanabe, Deep clustering: Discriminative embeddings for segmentation and separation, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2016, pp. 31–35.
[15] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496.
[16] M. Du, S. Ding, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, Knowledge-Based Systems 99 (2016) 135–145.
[17] Z. Bian, F.-L. Chung, S. Wang, Fuzzy density peaks clustering, IEEE Transactions on Fuzzy Systems 29 (2021) 1725–1738.
[18] J. Xie, H. Gao, W. Xie, X. Liu, P. W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors, Information Sciences 354 (2016) 19–40.
[19] R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, Information Sciences 450 (2018) 200–226.
[20] D. Cheng, S. Zhang, J. Huang, Dense members of local cores-based density peaks clustering algorithm, Knowledge-Based Systems 193 (2020) 105454.
[21] M. Du, S. Ding, X. Xu, Y. Xue, Density peaks clustering using geodesic distances, International Journal of Machine Learning and Cybernetics 9 (2018) 1335–1349.
[22] Y. Chen, X. Hu, W. Fan, L. Shen, Z. Zhang, X. Liu, J. Du, H. Li, Y. Chen, H. Li, Fast density peak clustering for large scale data based on kNN, Knowledge-Based Systems 187 (2020).
[23] X. Xu, S. Ding, M. Du, Y. Xue, DPCG: An efficient density peaks clustering algorithm based on grid, International Journal of Machine Learning and Cybernetics 9 (2018) 743–754.
[24] J. Guan, S. Li, X. He, J. Zhu, J. Chen, Fast hierarchical clustering of local density peaks via an association degree transfer method, Neurocomputing 455 (2021) 401–418.
[25] L. Yaohui, Z. Ma, Y. Fang, Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy, Knowledge-Based Systems 133 (2017) 208–220.
[26] G. Wang, Y. Wei, P. W. Tse, Clustering by defining and merging candidates of cluster centers via independence and affinity, Neurocomputing 315 (2018) 486–495.
[27] A. Lotfi, P. Moradi, H. Beigy, Density peaks clustering based on density backbone and fuzzy neighborhood, Pattern Recognition 107 (2020) 107449.
[28] Z. Su, T. Denoeux, BPEC: belief-peaks evidential clustering, IEEE Transactions on Fuzzy Systems 27 (2019) 111–123.
[29] F. Fang, L. Qiu, S. Yuan, Adaptive core fusion-based density peak clustering for complex data with arbitrary shapes and densities, Pattern Recognition 107 (2020) 107452.
[30] D. Cheng, Q. Zhu, J. Huang, Q. Wu, L. Yang, Clustering with local density peaks-based minimum spanning tree, IEEE Transactions on Knowledge and Data Engineering 33 (2021) 374–387.
[31] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E. Y. Chang, Parallel spectral clustering in distributed systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2010) 568–586.
[32] D. Cai, X. Chen, Large scale spectral clustering via landmark-based sparse representation, IEEE Transactions on Cybernetics 45 (2014) 1669–1680.
[33] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, C.-K. Kwoh, Ultra-scalable spectral clustering and ensemble clustering, IEEE Transactions on Knowledge and Data Engineering 32 (2020) 1212–1226.
[34] Y. Li, W. Liu, Y. Wang, D. Tao, Co-spectral clustering based density peak, in: International Conference on Communication Technology, 2015, pp. 925–929.

[35] J. Liu, C. Zhao, Density gain-rate peaks for spectral clustering, IEEE Access 9 (2021) 46000–46010.
[36] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 888–905.
[37] C. F. Van Loan, G. Golub, Matrix computations, Johns Hopkins Press, 1996.
[38] A. Villar-Corrales, V. I. Morgenshtern, Scattering transform based image clustering using projection onto orthogonal complement, in: Workshop on Intelligent Cross-Data Analysis and Retrieval, 2021, pp. 24–32.
[39] D. Steinley, Properties of the Hubert-Arable Adjusted Rand Index, Psychological Methods 9 (2004) 386–396.
[40] W. Xu, X. Liu, Y. Gong, Document clustering based on non-negative matrix factorization, in: Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003, pp. 267–273.
[41] Y. Yang, D. Xu, F. Nie, S. Yan, Y. Zhuang, Image clustering using local discriminant models and global integration, IEEE Transactions on Image Processing 19 (2010) 2761–2773.