Semi-supervised clustering guided by pairwise constraints and local density structures★

Zhiguo Long[a,c], Yang Gao[a], Hua Meng[b], Yuxu Chen[d,*], Hui Kou[d]

[a]*School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, 611756, China*
[b]*School of Mathematics, Southwest Jiaotong University, Chengdu, 611756, China*
[c]*Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, Chengdu, 611756, China*
[d]*School of Mathematics, Sichuan University, Chengdu, 610014, China*

Abstract

Clustering based on local density peaks and graph cut (LDP-SC) is one of the state-of-the-art algorithms in unsupervised clustering, which first divides the data set to be multiple local trees, and then aggregates these local trees to obtain the final clustering result. However, for complex data sets, there might exist data points from different classes in the same local tree. In this article, we use pairwise constraint information to resolve this issue and propose a semi-supervised local density peaks and graph cut based clustering algorithm (SLDPC). In particular, SLDPC proposes *intra-cluster conflict resolution* and *inter-cluster conflict resolution* steps to split the local trees which are inconsistent with the provided pairwise constraint information. Theoretically, we show that the two steps will finish in a finite number of operations and the split local trees will be consistent with the pairwise constraint information. Subsequently, *root node redirection* and *noise filtering* steps are designed to avoid the local trees becoming too fragmented. Finally, we exploit the E2CP algorithm to further improve the similarity matrix between local trees using the pairwise constraint information, and the spectral clustering algorithm is adopted to obtain the clustering result. Experiments on multiple widely used synthetic and real-world data sets show that SLDPC is superior to LDP-SC and several other semi-supervised prominent clustering algorithms for most of the cases.

Keywords: semi-supervised clustering, local density peaks, pairwise constraint propagation, inter-cluster conflict resolution

1. Introduction

Clustering [1] is the process of dividing a data set into multiple disjoint subsets while maximizing similarity within each subset and minimizing similarity between subsets [2, 3]. It is an important branch in the field of data mining and machine learning and has been widely used in scientific research and engineering applications [4, 5, 6] such as image segmentation [7], community discovery [8, 9] and environmental analysis [10].

Clustering by fast search and find of density peaks (DPC) [11, 12] is a widely used clustering algorithm based on density peaks, which performs well in clustering local structures. However, it has drawbacks in global clustering such as it cannot effectively deal with data sets with different densities in different parts and it is difficult to use Euclidean distance to fully exploit the manifold structure of the data. Spectrum clustering (SC) [13] is a graph cut clustering algorithm based on the similarity matrix. It is effective in the overall structural division but has shortcomings such as insufficient utilization of local information, impracticality for data sets with significant differences in point numbers between clusters, and sensitivity to changes in similarity measures and clustering parameter selection. Clustering based on local density peaks and graph cut (LDP-SC) [14] combines the advantages of DPC and SC, which first uses DPC to establish trees (called *family trees*) in local areas, and then uses the improved graph cut algorithm to aggregate these local trees to complete the clustering, thus taking into account both local and global information.

Although previous results have shown the superiority of LDP-SC over other similar methods, the construction of the family trees may lead to clustering errors on some complex data sets as shown in Figure 1, which displays the clustering results of LDP-SC and ground-truth clusters on the Compound and Pathbased data sets. For the Compound data set, at the stage of building family trees, LDP-SC makes mistakes for close clusters that have

⋆Accepted version. Formal version available at: https://doi.org/10.1016/j.patcog.2024.110751
∗Corresponding author.
Email address: yuxuchen1210@sina.com (Yuxu Chen)

significant differences in densities. The reason is that the high-density cluster is surrounded by the low-density cluster, and the principle of LDP-SC (and DPC variants) that "points with higher density and closest distance as the parent node" is no longer applicable in this case. The mistakes make it impossible to obtain desirable results for subsequent tree clustering. On the other hand, for the Pathbased data set, LDP-SC is basically correct in the stage of building family trees. However, due to the complex data distribution, the banded cluster is cut into pieces during the graph cut process, which again results in poor performance of clustering.

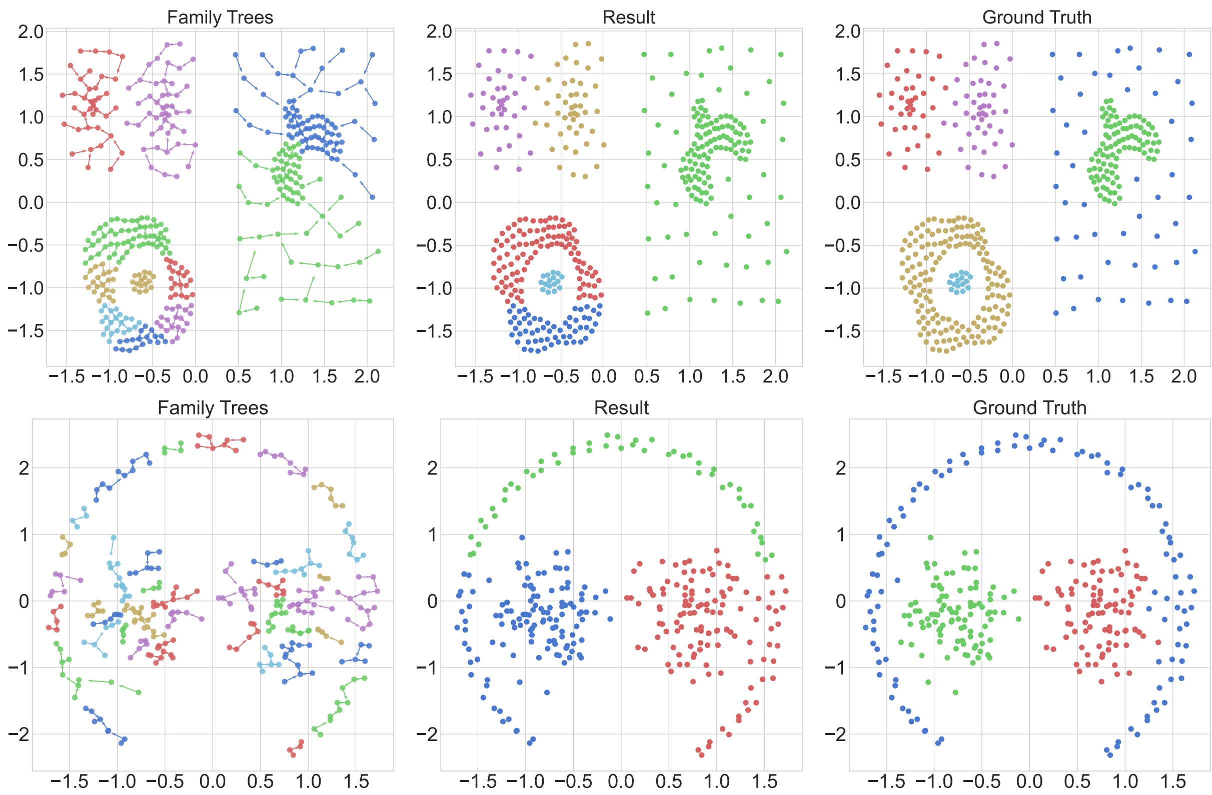

Figure 1: Problems of LDP-SC on Compound and Pathbased.

To solve these problems, a common method is to use semi-supervised learning [15], which utilizes a small amount of auxiliary information to help the algorithm to make decisions. In this article, we propose a semi-supervised local density peaks and graph cut based algorithm (SLDPC) builds on LDP-SC. SLDPC utilizes semi-supervised information in the form of pairwise constraints (i.e., must-links and cannot-links) and designs *intra-cluster con-*

flict resolution and *inter-cluster conflict resolution* steps to split and reorganize local trees. The two steps ensure that the constructed local trees are consistent with given constraint information, effectively preventing errors and ultimately enhancing the purity of the trees. Subsequently, two post-processing steps, i.e., *root node redirection* and *noise filtering*, are designed to avoid the family trees being too fragmented. Additionally, in the clustering step, SLDPC combines the classical pairwise constraint propagation algorithm E2CP [16] to further optimize the similarity matrix between trees, therefore improving the result of aggregating trees and thus of clustering.

The main contributions of this article are:

1. We proposed a semi-supervised version of LDP-SC. By utilizing semi-supervised information, we design intra-cluster conflict resolution, inter-cluster conflict resolution, root node redirection and noise filtering steps to improve the purity of local trees while avoiding them being fragmented.
2. We integrate the E2CP algorithm to optimize the similarity measure between local trees to make it more consistent with the given constraint information.
3. We validate the effectiveness of SLDPC experimentally on two synthetic data sets and twelve real-world data sets. The results show its superiority over several prominent baseline algorithms, including LDP-SC.

2. Related Work

In recent years, numerous semi-supervised clustering algorithms have been proposed and implemented in various scenarios. These algorithms can be mainly classified into two categories: those based on label information and those based on pairwise constraints.

2.1. Semi-supervised clustering based on label information

The most intuitive way to add supervision information is to directly give the labels of some samples. Basu et al. [17] regarded the provided labeled data as "seeds" and proposed the Seeded-KMeans and Constrained-KMeans algorithms. These algorithms choose the initial points of KMeans from the seed set to ensure the correctness of initial points, instead of randomly selecting from the whole data set. In the Constrained-KMeans algorithm, the label information of "seeds" will not be changed with the iteration of KMeans, which ensures that the final result does not violate the supervision information. PLCC [18] uses "category utility function" [19] as a regularization term in the loss function of KMeans or spectral clustering to improve the consistency between label values and given constraints in the objective function. SMKFC-ER [20] is proposed from the perspective of entropy and relative entropy. In this algorithm, the objective function is divided into a supervised part and an unsupervised part. Entropy coefficient and relative entropy divergence measure are applied rather than fuzzifier for the unsupervised part and the geometric distance measure for the semi-supervised part respectively. CSSC [21] is proposed based on the assumption that the cluster is compact, i.e., without low-density separation. It uses a top-down approach to iteratively refine the results of traditional clustering algorithms until each cluster is compact enough. CSSC is integrated into conventional clustering algorithms, leading to the emergence of CSSC-KMeans and CSSC-SC algorithms.

2.2. Semi-supervised clustering based on pairwise constraints

Pairwise constraint information is weaker than label information. The pairwise constraints are composed of *must-link* and *cannot-link* relations. The former indicates that two points belong to the same category, while the latter indicates that they belong to different classes. Obviously, label information can be converted into pairwise constraints, but not vice versa.

Since pairwise constraints provide information between two data points, an intuitive idea is to utilize them to adjust the similarity between points. Kamvar et al. [22] used supervision information to optimize the similarity matrix in spectral clustering by directly setting the similarity between *must-link* pairs to 1 and the similarity between *cannot-link* pairs to 0. However, this algorithm only adjusts the similarity for points with constraint information, which is relatively limited. Hence, the idea of propagating the similarity between constraint pairs to all samples (constraint propagation) is developed [23]. Constraint propagation has also been described as a semi-definite programming (SDP) problem in [24]. To further utilize constraint information and overcome the high time complexity of SDP,

E2CP algorithm [16] decomposes the constraint propagation problem into a set of independent semi-supervised classification sub-problems so that it can achieve excellent performance with lower time complexity.

In the PCOG [25] framework, a pairwise constraint based regularization method is integrated with their previous work [26]. This method ensures that the number of connected components (subgraphs) in a similar matrix is equal to the number of clusters while satisfying that cannot-link pairs are in different connected components (must-link pairs are in the same component). SSFPC [27] incorporates constraint information as a regularization term into the objective function of fuzzy clustering and solves non-convex problems using an improved expectation-maximization algorithm.

Taking advantage of the excellent performance of E2CP, ISSCE [28] is proposed for high-dimensional data clustering by combining clustering ensembles with random subspace techniques. It uses E2CP in different subspaces and obtains the final clustering result by a consensus function. In addition, based on the belief that members in the ensemble algorithm should have different contributions and not all constraints are useful, the DCECP [29] framework is proposed. The WECR k-means algorithm [30] uses random subspace and random sample techniques [31] to obtain base partitions, and then incorporates constraint information and Silhouette coefficient [32] to calculate similarity matrix for spectral clustering.

To address potential conflicts between different types of constraints from different sources such as pairwise constraints and label information, and prior knowledge from different domain experts, a multi-source fusion method is proposed and then the SC-MPI algorithm [33] based on spectral clustering is developed.

3. Preliminaries

3.1. LDP-SC

LDP-SC [14] first builds family trees based on the density and neighbors of the data points, so that the points can be clustered locally, and then the family trees are aggregated according to the similarity to complete the clustering globally.

For a data set $X = \{x_1, x_2, \cdots, x_N\}$, denote by $\mathsf{KNN}(x_i)$ the set of k nearest neighbors of the data point x_i (not including x_i). The density of an arbitrary point is defined by:

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

where dist is a $N \times k$ matrix, and dist_{ij} is the Euclidean distance between x_i and its j-th nearest neighbor. Then, the parent $P(x_i)$ of a data point x_i is defined as follows:

$$P(x_i) = \begin{cases} \underset{x_j \in \mathsf{higher}(x_i)}{\arg\min}\ \mathsf{dist}_{ij}, & \text{if } \mathsf{higher}(x_i) \neq \emptyset, \\ \mathsf{None}, & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathsf{higher}(x_i) = \{x_j | x_j \in \mathsf{KNN}(x_i), \rho(x_j) > \rho(x_i)\}$. If x_i has no parent ($P(x_i) = \mathsf{None}$), then x_i is defined as a root. Each root and its descendants form a family tree.

Definition 1. 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 $x_j = P(x_i)$.

For any root r_i, we use $T(r_i)$ to refer to the family tree with r_i as its root. For any point x, we denote by $\mathsf{root}(x)$ the root of the family tree that contains x. Next, the similarity between two family trees is decided together by the number of shared nearest neighbors, the separation measure and the distance between them.

Definition 2. The *similarity* between $T(r_i)$ and $T(r_j)$ is defined to be

$$\mathsf{sim}(T(r_i), T(r_j)) = \frac{|\mathsf{SNN}(T(r_i), T(r_j))|}{(1 + \sigma(T(r_i), T(r_j))) \cdot (1 + d(r_i, r_j))}, \tag{3}$$

where, $d(r_i, r_j) = \|r_i - r_j\|_2$, $\mathsf{SNN}(T(r_i), T(r_j))$ is the shared nearest neighbors of two trees, and $\sigma(T(r_i), T(r_j))$ is the separation between two trees (cf. [14]).

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 $W_{ij} = W_{ij} / \max(W)$. In order to maintain the connectedness of the graph, a Gaussian kernel distance matrix ${W_{ij}}' = \exp(-\frac{d(r_i,r_j)^2}{2})$ between roots is constructed, and W is updated to be $W + \theta W'$, where θ is set to be a small value 0.001.

To simplify notations, root r is used to represent the tree $T(r)$ in the connection graph. Let $\mathcal{T}$ be the set of family trees ($|\mathcal{T}| = p$), and set the number of target clusters to be C. LDP-SC adapts the Ncut loss function for family trees as follows:

$$\operatorname{Ntcut}(A_1, \ldots, A_C) = \sum_{i=1}^{C} \frac{\operatorname{cut}\left(A_i, \overline{A}_i\right)}{\operatorname{vol}(A_i) \cdot \sum_{r \in A_i} |T(r)|}, \tag{4}$$

where $A_i = \{r_{i_1}, \ldots, r_{i_s}\}$ is a set of family trees ($\bigcup_i A_i = \mathcal{T}$), $\operatorname{vol}(A_i) = \sum_{r_u \in A_i, r_j \in \mathcal{T}} W_{uj}$, and $\operatorname{cut}(A_i, \overline{A_i}) = \frac{1}{2} \sum_{r_u \in A_i, r_v \in \overline{A_i}} W_{uv}$. Finally, by finding a solution that minimizes the above loss function, LDP-SC gets the clustering result. For the specific solution process, see [3, 14].

3.2. E2CP

E2CP is a semi-supervised clustering algorithm based on pairwise constraints. The main idea is to adjust the initial similarity matrix according to the pairwise constraint information such that the similarity between data points is more consistent with the actual label distribution. It can be divided into three steps: (1) Calculating the initial similarity matrix; (2) Adjusting the similarity matrix by means of pairwise constraints; (3) Clustering according to the adjusted similarity matrix.

Given any data set $X = \{x_1, x_2, \ldots, x_m\}$, denote the pairwise constraints by $\mathcal{M}$ and $\mathcal{C}$, where $\mathcal{M}$ and $\mathcal{C}$ represent *must-link* and *cannot-link* separately: $(i,j) \in \mathcal{M}$ means that x_i and x_j belong to the same class; $(i,j) \in \mathcal{C}$ means that x_i and x_j belong to different classes. $\mathcal{M}$ and $\mathcal{C}$ can be denoted as a constraint matrix $Z = \{z_{ij}\}_{m \times m}$ as $z_{ij} = +1$ if $(i,j) \in \mathcal{M}$, $z_{ij} = -1$ if $(i,j) \in \mathcal{C}$, and $z_{ij} = 0$ otherwise.

E2CP first calculate the original similarity matrix $W = \{w_{ij}\}_{m \times m}$ as follows: if x_i is the k nearest neighbor of x_j (or vice versa), then define $w_{ij} = \frac{a(x_i,x_j)}{\sqrt{a(x_i,x_i)}\sqrt{a(x_j,x_j)}}$; otherwise, define $w_{ij} = 0$. Usually set $a(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / t)$, where t is a hyper-parameter. And then set $W = (W + W^T)/2$ to ensure the symmetry of the similarity matrix.

Then the pairwise constraint propagation matrix, denoted by $F = \{f_{ij}\}_{m \times m}$, is calculated by the following closed-form solution [34]:

$$F^* = (1-\alpha)^2 (I - \alpha \overline{L})^{-1} Z (I - \alpha \overline{L})^{-1}. \tag{5}$$

After calculating F^*, set $\tilde{f}^*_{ij} = f^*_{ij} / \max_{i',j'} \left| f^*_{i'j'} \right|$ to get a normalized matrix $\tilde{F}^*$. Then, use $\tilde{F}^*$ to adjust the original similarity matrix W in the following way:

$$\tilde{w}_{ij} = \begin{cases} 1 - \left(1 - \tilde{f}^*_{ij}\right)(1 - w_{ij}), & \tilde{f}^*_{ij} \geq 0; \\ \left(1 + \tilde{f}^*_{ij}\right) w_{ij}, & \tilde{f}^*_{ij} < 0. \end{cases} \tag{6}$$

Finally, the new similarity matrix $\tilde{W}$ is brought into the spectral clustering algorithm to obtain the final clustering result.

3.3. Analysis of the problems of LDP-SC

For LDP-SC, points in the same family tree will be grouped into the same cluster, and if a family tree contains points that should be in different clusters, then the algorithm will make mistakes that cannot be corrected in later stages. In other words, the purity of trees is crucial for the clustering performance of LDP-SC. Also, even if the trees have high purity, the stage of aggregating trees of LDP-SC might specify wrong similarities between trees when dealing with complex data. Two examples of such problems of LDP-SC on constructing trees and specifying similarities have been given in Figure 1.

The above two problems of LDP-SC remind us that the two steps of LDP-SC, i.e., building family trees and aggregating family trees, still need to be improved. In this article, we propose a novel algorithm, called SLDPC, that resolves the two problems of LDP-SC by improving the two steps with the help of semi-supervised constraint information. Particularly, the proposed SLDPC algorithm will first use semi-supervised constraint information to help select parent nodes of points to improve the purity of family trees. Additionally, by utilizing E2CP, SLDPC will optimize the established similarity matrix of family trees according to the constraint information to improve the aggregation effect, and thus to improve the overall clustering performance.

Similar to LDP-SC, SLDPC is also divided into two main stages: building the family trees and aggregating family trees. The differences are that we add *intra-cluster conflict resolution*, *inter-cluster conflict resolution*, *root node redirection* and *noise filtering* steps in the stage of building family trees, and add the similarity matrix adjustment step in the family tree aggregation stage based on pairwise constraints $\mathcal{M}$ and $\mathcal{C}$ (cf. Section 3.2). In the following, we will discuss this two main steps in sequence.

4. SLDPC stage 1: Building family trees based on pairwise constraints

4.1. Building initial family trees

We use the corresponding steps in LDP-SC (cf. Algorithm 1) to obtain the initial family trees. LDP-SC assumes that points in each family tree are highly similar and in a same class, and subsequently, the clustering of data points can be transformed into aggregating family trees. However, as shown in Introduction, this assumption is unreasonable for some complicated data distributions. We utilize *must-link* and *cannot-link* pairwise constraint information to correct the family trees.

Algorithm 1: BuildFamilyTree

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 $x_i \in X$ **do**
  Calculate $\mathsf{KNN}(x_i)$ and the distance matrix dist;
  Calculate $\rho(x_i)$;
root $\leftarrow \varnothing$;
foreach $x_i \in X$ **do**
  Compute $P(x_i)$ according to Eq. 2;
  if $P(x_i) =$ None **then**
    root $\leftarrow$ root $\cup \{x_i\}$;
return P, root.

4.2. Intra-cluster conflict resolution

Intra-cluster conflict resolution is the process of splitting family trees to increase the purity of each tree. Only the cannot-link information $\mathcal{C}$ is used at this step. For a family tree T, if there exist points $x_i, x_j \in T$ such that $(x_i, x_j) \in \mathcal{C}$, then it means that there is a conflict inside T, as points in a tree should be in the same class. We resolve this kind of conflicts by splitting the corresponding tree into multiple trees. To determine where to split a tree, we define the *difference degree* between a pair of parent-child nodes in a family tree to measure how different these nodes are.

Definition 3. If x_i and x_j are a pair of parent-child nodes, then the *difference degree* diff between them is defined as:

$$\mathsf{diff}_{i,j} = \sqrt{d_{ij}^2 + \lambda(\rho_i - \rho_j)^2} \tag{7}$$

where d_{ij} represents the Euclidean distance between x_i and x_j; ρ_i and ρ_j represent the density of x_i and x_j respectively; λ is a weighting parameter to balance the scale of the two terms. Since the data sets will undergo normalization, the density and distance will have roughly the same magnitude, and we will take an empirical value of λ as 1 in this paper.

For two conflicting points in a tree, there is a unique path between them. For each pair of parent-child nodes on the path, we calculate the diff value, and split the tree into two trees on the edge where the diff value is the largest. It is easy to see that by repeating the tree splitting operation finite times, there will be no *cannot-link* pairs in each family tree.

Figure 2 shows the process of splitting a tree. The left shows the family trees established by Algorithm 1, and the two pentagrams represent a pair of *cannot-link* points. The middle shows the path (red) between the constrained pairs, and the right shows the result after splitting a tree. We can see that the chosen position to split the family tree is consistent with our intuition, i.e., it is where the difference is visually the largest.

4.3. Inter-cluster conflict resolution

After intra-cluster conflict resolution, the points in the same tree will not have a conflict with each other. However, there may still exist conflicts between trees.

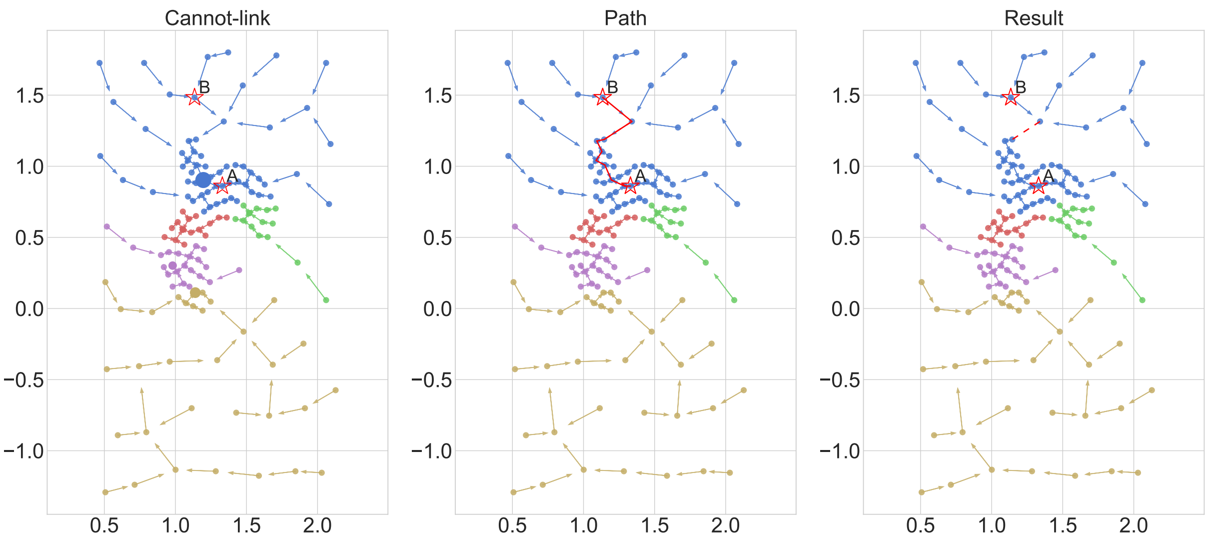

Figure 2: Process of splitting a tree.

Example 1. Consider the left picture in Figure 3, where $\mathcal{M} = \{(A, B), (C, D)\}$ and $\mathcal{C} = \{(E, F)\}$. By the assumptions that both points in a same family tree and points with must-link belong to the same class, we will have F and A, A and B, B and C, C and D, and D and E are in the same class, respectively. Then by transitivity, it is easy to conclude that E and F should also belong to the same class, which is a contradiction to the cannot-link constraint.

In the following, we propose two steps to solve this problem: detecting conflicts and resolving conflicts.

Note that conflicts between trees might be implicit, as the above example has shown. We need to introduce a closure operation for the must-link constraints to reveal all implicit must-link information and to detect conflicts.

First, let $G = (V, E)$ be an undirected graph, where $V = \{T_1, \ldots, T_p\}$ are the family trees and an edge exists between two family trees if there is a must-link relation between two points in the corresponding family trees. Then we construct a must-link matrix $M^{\mathcal{M}}$ and a cannot-link matrix $M^{\mathcal{C}}$ for family trees based on G. For the must-link matrix $M^{\mathcal{M}}$, the element at row i and column j $(i \neq j)$ is 1 iff T_i and T_j in G are in the same connected component of G. For the cannot-link matrix $M^{\mathcal{C}}$, the element at row i and column j $(i \neq j)$

is -1 iff there is a point x_i in some tree from the connected component containing T_i and a point x_j in some tree from the connected component containing T_j, and $(x_i, x_j) \in \mathcal{C}$. If $\exists i, j$ s.t. $M_{ij}^{\mathcal{M}} = 1$ and $M_{ij}^{\mathcal{C}} = -1$, then a conflict is detected, as $M_{ij}^{\mathcal{M}} = 1$ indicates that T_i and T_j are in the same component and all points from the same component should be in the same class, but $M_{ij}^{\mathcal{C}} = -1$ indicates that there are points from the component containing T_i and T_j that should not be in the same class.

Tree splitting is then exploited to resolve the detected inter-cluster conflicts. The general idea is as follows:

- When there is an inter-cluster conflict, it will occur between two points that are in two family trees that are in the same component of G.
- We can split a family tree into two new trees by removing an edge of it, and accordingly update the graph G of family trees to include the two new trees as vertices and to remove the vertex corresponding to the original tree.
- By repeatedly splitting the trees and updating the graph G, finally we can make two points that have cannot-link belong to two trees that are not in the same connected component of G, given that the must-link and cannot-link constraints are consistent (see Theorem 1).

To determine which tree to split and where to split, we devise a greedy strategy to find an edge in a tree to remove:

1. Identify all candidate points that are involved in a must-link or a cannot-link constraint;
2. Find and remove the edge that is on the path connecting two candidate points in the same tree and has the largest diff value.

After splitting a tree, the graph G will be updated accordingly as mentioned above. Note that the corresponding vertices of the resulting two trees may be still in the same connected component of G, because there might be a must-link constraint between points in these two trees, which means that the conflict still exists. Nevertheless, as observed in Theorem 1, by repeating the above steps finite times, an inter-cluster conflict will be correctly resolved.

Definition 4. A set $\mathcal{M}$ of must-link constraints and a set $\mathcal{C}$ of cannot-link constraints are said to be *consistent* iff for any two points x_i and x_j having a cannot-link constraint in $\mathcal{C}$, there is no sequence of points $x_i = x_{i_0}, x_{i_1}, \ldots, x_{i_t} = x_j$ s.t. x_{i_l} and $x_{i_{l+1}}$ $(l = 0, \ldots, t-1)$ are all connected by a must-link constraint in $\mathcal{M}$.

Theorem 1. *Suppose that the given set of must-link constraints and the set of cannot-link constraints are consistent. Let x_i and x_j be two points from two trees corresponding to two vertices in the same connected component of G. If there is a cannot-link constraint between x_i and x_j, then by following the steps described beforehand in finite times, the tree containing x_i and the tree containing x_j will no longer be in the same connected component of (the updated) G.*

Proof. Assume on the contrary that the tree containing x_i and the tree containing x_j are still in the same connected component of G and no tree splitting is possible. There are two cases:

- x_i and x_j are in the same tree.
- x_i and x_j are in two different trees.

For the first case, x_i and x_j are two candidate points in the same tree, and the edge that is on the tree-path connecting them and has the largest diff value can be removed, which is a contradiction to the assumption that no tree splitting is possible. For the second case, as the tree $T(x_i)$ containing x_i and the tree $T(x_j)$ containing x_j are still in the same component, there must be a path of must-link edges in G between $T(x_i)$ and $T(x_j)$, and for each tree on the path there are two points in it, each of are involved in a must-link constraint. If there are two different such points in a tree on the path, then an edge on the tree-path connecting them can be removed when no other edges can be removed, resulting again a contradiction to the assumption. If there are no such two points, i.e., each tree on the path contains only one point, then in this case the set of must-link constraints and the set of cannot-link constraints are not consistent, which is a contradiction to the given condition. $\square$

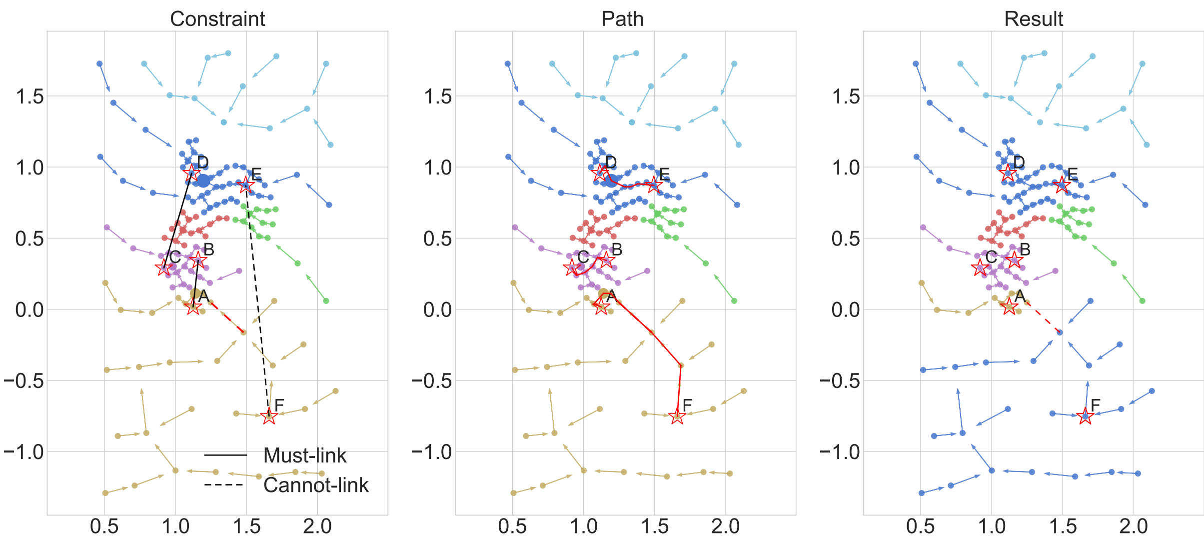

Figure 3: Process of inter-cluster conflict resolution.

Figure 3 illustrates the process of inter-cluster conflict resolution for the configuration in Example 1. Let T_1 be tree containing the points A and F, T_2 containing B and C, and T_3 containing D and E. Then in G, there are two edges, i.e., (T_1, T_2) and (T_2, T_3), and thus a connected component $\{T_1, T_2, T_3\}$. The corresponding must-link matrix and cannot-link matrix for T_1, T_2, T_3 are as follows:

$$M^{\mathcal{M}} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \quad M^{\mathcal{C}} = \begin{pmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}$$

From the above two matrices, we can see that there are conflicts in the component $\{T_1, T_2, T_3\}$, e.g., E and F are in the same class according to $M^{\mathcal{M}}$ but $(E, F) \in \mathcal{C}$. For tree splitting, the candidate points are $\{A, B, C, D, E, F\}$, and $(A, F), (B, C), (D, E)$ are the only pairs consisting of two candidate points in the same tree. The path connecting the points in each pair is drawn as a red curve in the middle of Figure 3. By comparing the diff values of the edges on the paths, an edge on the path connecting A and F is chosen to be removed. The result of splitting is shown on the right of Figure 3. It can be seen that the conflict has been resolved, as the tree containing E and the tree containing F are no longer in the same connected component.

4.4. Root nodes redirection

Both intra-cluster conflict resolution and inter-cluster conflict resolution adopt the idea of tree splitting, which resolves conflicts by making family trees smaller. However, continuously splitting trees may cause the trees to become too fragmented. Especially at the inter-cluster conflict resolution stage, the trees are split repeatedly until the conflicts are all resolved, which may result in a large number of fragmented trees. In extreme cases, each tree will contain only one point, which is not desired.

Figure 4 shows the result of intra-cluster conflict resolution (left) and inter-cluster conflict resolution (middle) for one instance of randomly generating $0.5n$ (n represents the number of samples) pairwise constraints. Points with red circles represent the root nodes of family trees. It can be seen in the middle that after inter-cluster conflict resolution, although the purity of the current family trees has become very high, there are many fragmented trees containing only a single node.

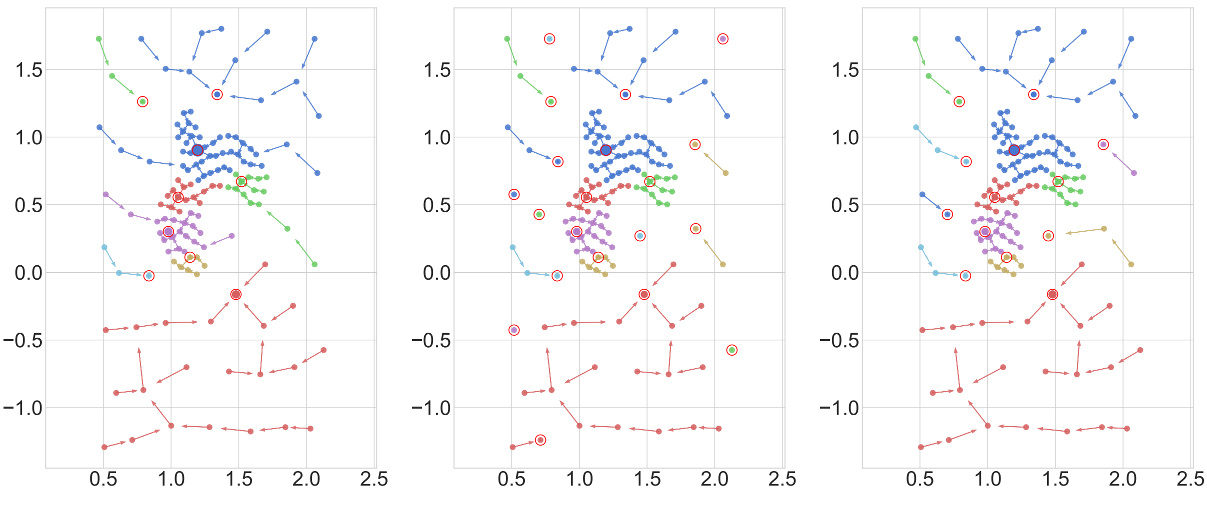

Figure 4: Results of family trees.

Thus, a further merging operation on the split family trees is needed. Similar to the definition of the parent of a point in Eq. 2, with pairwise constraint information, it is reasonable to consider that the parent of the root node of a family tree should also be consistent with the constraint information, i.e., for the tree T_i containing the parent and the tree T_j containing the root node, $M^{\mathcal{C}}_{ij} \neq -1$.

More specifically, for the family trees after conflict resolution, we check the root nodes in the order where densities are from high to low. For each root node x_i, we define the parent of it to be the closest point $\bar{p} \in \mathsf{KNN}(x_i)$ to x_i such that $\rho(\bar{p}) > \rho(x_i)$ and is consistent with x_i in terms of the cannot-link information as described above. We call this operation *root node redirection*, which will combine two trees. After combining two trees, we will also update the graph G of trees and the two matrices $M^{\mathcal{M}}$ and $M^{\mathcal{C}}$.

Figure 4 (right) shows the result of root node redirection for Figure 4 (middle). It can be seen that most of the fragmented trees have been merged while ensuring consistency with the constraint information. The number of family trees is reduced by more than half, and the purity is still maintained.

Algorithm 2 gives the whole process of building family trees based on pairwise constraints.

5. SLDPC stage 2: Aggregating family trees based on constraint propagation

5.1. Calculating similarity matrix

We first calculate the original similarity matrix of the constructed family trees and normalize it to obtain the matrix $W = \{w_{ij}\}_{p\times p}$ (p is the number of family trees), as in LDP-SC. Here, W is symmetric and $0 \leq w_{ij} \leq 1$. Then, in order to make full use of pairwise constraints information, we use the E2CP method (cf. Section 3.2) to optimize the similarity matrix by pairwise constraints. The constraint matrix Z required by E2CP is $M^{\mathcal{M}} + M^{\mathcal{C}}$ obtained from the graph G of family trees. Recall that the constraint propagation matrix $F^* = \{f^*_{ij}\}_{p\times p}$ can be obtained by:

$$F^* = (1-\alpha)^2(I-\alpha\overline{L})^{-1}Z(I-\alpha\overline{L})^{-1}, \tag{8}$$

where $\overline{L} = D^{-1/2}WD^{-1/2}$, D is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$. We set $\tilde{f}^*_{ij} = f^*_{ij}/\max_{i',j'}\left|f^*_{i'j'}\right|$ to obtain the standardized matrix $\tilde{F}^*$, and then use Eq. 6 to calculate the

Algorithm 2: FamilyTreeWithConstraint

Input: A data set X; the number of nearest neighbors k; pairwise constraints $\mathcal{M}$, $\mathcal{C}$
Output: Family trees represented by a parent-child relation P and the set of root nodes root.

$P, \mathsf{root} \leftarrow \text{BuildFamilyTree}(X, k)$;
// Intra-cluster conflict resolution;
foreach $(x_i, x_j) \in \mathcal{C}$ **do**
  if $\mathsf{root}(x_i) = \mathsf{root}(x_j)$ **then**
    Find the parent-child pair $a = P(b)$ with the largest diff degree on the path of x_i and x_j;
    $P(b) \leftarrow \mathsf{None}; \mathsf{root} \leftarrow \mathsf{root} \cup \{b\}$;
// Inter-cluster conflict resolution;
Construct the graph G of family trees;
$M^{\mathcal{M}}, M^{\mathcal{C}} \leftarrow$ *construct from* G;
while $\exists$ *a conflict in the constraint matrix* **do**
  $\{r_1, r_2, \cdots, r_t\} \leftarrow$ the component corresponding to the conflict;
  $\Psi \leftarrow \emptyset$;
  foreach $(x_i, x_j) \in \mathcal{C} \cup \mathcal{M}$ **do**
    if $\mathsf{root}(x_i), \mathsf{root}(x_j) \in \{r_1, r_2. \cdots, r_t\}$ **then**
      $\Psi \leftarrow \Psi \cup \{x_i, x_j\}$;
  foreach $x_i, x_j \in \Psi$ **do**
    if $\mathsf{root}(x_i) = \mathsf{root}(x_j)$ **then**
      Calculate diff degrees of all parent-child pairs on the path of x_i and x_j;
  Find the parent-child pair $P(b) = a$ with the largest diff degree;
  $P(b) \leftarrow \mathsf{None}; \mathsf{root} \leftarrow \mathsf{root} \cup \{b\}$;
  Update G;
  $M^{\mathcal{M}}, M^{\mathcal{C}} \leftarrow$ *update from* G;
// Root node redirection;
sorted_root $\leftarrow$ sort(root);
foreach $x_i \in$ sorted_root **do**
  if $\exists$ *closest point* $\bar{p} \in \mathsf{KNN}(x_i)$ *s.t.* $\rho(\bar{p}) > \rho(x_i)$, *and* $\bar{p}$ *as parent of* x_i *is consistent with the cannot-link matrix* **then**
    $P(x_i) \leftarrow \bar{p}$;
    $\mathsf{root} \leftarrow (\mathsf{root} \setminus \{x_i\}) \cup \{\bar{p}\}$;
    Update G;
    $M^{\mathcal{M}}, M^{\mathcal{C}} \leftarrow$ *update from* G;
return P, root.

final similarity matrix $\tilde{W}$.

In LDP-SC, to avoid the adverse effects of multiple connected components on the spectral clustering, W is updated to be $W + \theta W'$ to adjust the similarity matrix. Our algorithm is more conservative. Specifically, we first use E2CP method to obtain the adjusted similarity matrix $\tilde{W}$, and then calculate the number of connected components based on the similarity matrix (two nodes r_i, r_j are connected if $\tilde{W}_{ij}$ is positive). If the number of connected components is greater than the target number of clusters, then we adjust the similarity matrix for graph cut by $\tilde{W} = \tilde{W} + \theta W'$, where $W'_{ij} = \exp(-\left\|r_i - r_j\right\|^2/2)$.

5.2. Noise filtering

In practice, we find that after the root node redirection step, there are still some family trees that contain very few points, which may cause these trees to be clustered as separate categories in the graph cut stage and greatly affect the final clustering result. To solve this problem, a noise filtering step is adopted.

We first set a threshold τ. When the number of points contained in a family tree is less than or equal to τ, the root node is marked as noise. We traverse the set of all noise root nodes in the order where densities are from high to low, and for each noise root, check if there is a point in its k nearest neighbors having a higher density. If so, the closest such point is set as the parent of the noise root; otherwise, the closest non-noise root is set as its parent. By doing so, the trees containing noise roots are merged into other trees, thus avoiding clustering them as a single class.

After the trees containing noise roots are merged, the rows and columns corresponding to the noise points in $\tilde{W}$ are deleted, and then the final similarity matrix $\tilde{W}^*$ is obtained.

5.3. Graph cut and clustering of trees

After the noise filtering step, we obtain the final family trees and the corresponding similarity matrix $\tilde{W}^*$. The last step is to perform graph cut and aggregate the family trees to obtain the final clustering result, which is exactly the same as LDP-SC. The whole process of the SLDPC algorithm is shown in Algorithm 3.

5.4. Time complexity analysis

The time complexity of LDP-SC is $\mathcal{O}(n^2 + p^3)$ [14], where n is the number of data points, and p is the number of family trees. Assume that the number of constraints is m. The proposed SLDPC algorithm is based on LDP-SC with five additional steps.

(1) Intra-cluster conflict resolution (lines 3-6 in Algorithm 2): For each constraint, one need to find the path between them, and calculate the diff value between adjacent nodes in the path, so the time complexity is $\mathcal{O}(mn)$.

(2) Inter-cluster conflict resolution (lines 8-22 in Algorithm 2): For lines 8-9, the connected graph between trees are constructed and then the pairwise constraint matrix is calculated, the time complexity is $\mathcal{O}(m + p^2)$. Suppose the number of iterations of the while loop from line 10 to line 22 is s. For lines 11-12, one needs to find components corresponding to the conflict, and the time complexity is $\mathcal{O}(p^2)$. For lines 13-15, constraint pairs for the conflict is identified and the complexity is $\mathcal{O}(m)$. For lines 16-20, the complexity is the same as the intra-cluster stage with $\mathcal{O}(mn)$. For lines 21-22, The complexity is similar to Lines 8-9, with $\mathcal{O}(p^2)$. Thus, the total complexity in this stage is $\mathcal{O}(s(mn + p^2))$.

(3) Root node redirection (lines 24-31 in Algorithm 2): For line 24, the time complexity of sort function is $\mathcal{O}(p \log p)$. For lines 25-31, each root node will be checked if it should be assigned to another root, and the constraint matrix is updated, so the time complexity is $\mathcal{O}(p^3)$.

(4) E2CP propogation (line 10 in Algorithm 3): The time complexity is $\mathcal{O}(kmn)$ [16], where k is the number of nearest neighbors.

(5) Noise filtering (line 14 in Algorithm 3): A sort algorithm for the set of all noise root nodes is performed, and then parent nodes for them are identified in its k nearest neighbors, so the complexity is $\mathcal{O}(p \log p)$.

Therefore, the total time complexity for the additional operations in the proposed algorithm is $\mathcal{O}(mn + s(mn + p^2) + p^3 + kmn + p \log p)$. Usually, k and p are much smaller than n, and s is much smaller than m, so in most cases the actual time complexity of the overhead calculations w.r.t. LDP-SC is about $\mathcal{O}(smn + p^3)$.

Algorithm 3: SLDPC

Input: data set X; number of clusters C; number of nearest neighbors k; pairwise constraints $\mathcal{M}$ and $\mathcal{C}$

Output: Clustering result label.

$P, \mathsf{root} \leftarrow \text{FamilyTreeWithConstraint}(X, k, \mathcal{M}, \mathcal{C})$;

foreach $(r_i, r_j) \in \mathsf{root} \times \mathsf{root}$ *and* $r_i \neq r_j$ **do**

  $W_{ij} \leftarrow \mathsf{sim}(T(r_i), T(r_j))$ by Eq. 3;

$W \leftarrow W / \max(W)$ and set each W_{ii} to 1;

Construct the graph G of family trees;

$M^{\mathcal{M}}, M^{\mathcal{C}} \leftarrow$ *construct from* G;

$Z \leftarrow M^{\mathcal{M}} + M^{\mathcal{C}}$;

Calculate F^* by Eq. 8;

Using $\tilde{f}^*_{ij} = f^*_{ij} / \max_{i',j'} \left| f^*_{i'j'} \right|$ get standardized matrix $\tilde{F}^*$;

Use Eq. 6 to obtain similarity matrix $\tilde{W}$;

if *the number of connected components of* $\tilde{W}$ *is bigger than* C **then**

  Compute the Gaussian kernel distance matrix W';

  $\tilde{W} \leftarrow \tilde{W} + \theta W'$;

Obtain $\tilde{W}^*$ by noise filtering;

foreach $r_i \in \mathsf{root}$ **do** $D_{ii} \leftarrow \sum_j \tilde{W}^*_{ij}$;

$L \leftarrow D - \tilde{W}^*$;

foreach $r_i \in \mathsf{root}$ **do** $D^*_{ii} \leftarrow |T(r_i)|$;

$U \leftarrow DD^*$;

$E \leftarrow (t_1, \ldots, t_C)$, the C eigenvectors of $U^{-1/2} L U^{-1/2}$ corresponding to its C smallest eigenvalues;

foreach $(r_i, r_j) \in \mathsf{root} \times \mathsf{root}$ **do**

  $Y_{ij} \leftarrow E_{ij} / (\sum_{s=1}^{C} E^2_{is})^{1/2}$;

Apply K-means on Y to obtain cluster label c_i for each $T(r_i)$;

foreach $r_i \in \mathsf{root}$ **do**

  foreach $x_i \in T(r_i)$ **do**

    $\mathsf{label}(x_i) \leftarrow c_i$;

return label.

6. Empirical Evaluations

6.1. Experiment settings

We compare SLDPC with LDP-SC and several other prominent algorithms on both synthetic data sets and real-world data sets. The chosen prominent algorithms are displayed in Tabel 1.

Table 1: Comparison algorithms

Algorithm	Date, Publication	Parameters
SLDPC	Ours	$c, k, \tau, \alpha, \lambda$
LDP-SC [14]	2022, Information Sciences	c, k
E2CP [16]	2013, International Journal of Computer Vision	c, k, α
WECR k-means [30]	2019, IEEE Transactions on Knowledge and data engineering	c, γ, r_i, r_f
SSFPC [27]	2021, IEEE Transactions on Fuzzy Systems	α, β
A-SSC [35]	2023, IEEE Transactions on Neural Networks and Learning Systems.	-
TLRR [36]	2023, IEEE Transactions on Circuits and Systems for Video Technology	λ, β

* For multi-parameter algorithms, grid search is used to select the optimal parameters.

The parameter c (number of target clusters) of LDP-SC is set as the number of ground-truth clusters. The parameter k (number of nearest neighbors) is searched from 2 to 30. For SLDPC, parameters c, k are set to be the same as LDP-SC. The weighting parameter λ in Eq. 3 is set as 1. The noise filtering threshold τ is set as $\log_2(k)$, and the parameter α is set as 0.6 according to the original recommendation of E2CP algorithm. The implementation of E2CP is based on the descriptions of the original paper, using Eq. (5) to do pairwise constraint propagation.

For WECR k-means, the number of clusters c is set as the number of ground-truth clusters, γ is searched in $\{0.1, 0.2, \cdots, 1\}$, and the parameters r_i, r_f in the subspace stage is searched in $\{(0.7, 0.7), (0.7, 0.3), (0.3, 0.7)\}$. The source code is provided by the authors[1]. For SSFPC, α is searched from 0 to 0.3 with a step size of 0.02. β is searched in $\{0, 0.005\}$. The implementation is from the source code provided by the authors[2]. The source code of TLRR and A-SSC is provided by the authors, with parameters set to be same with those in the original papers.

[1] `https://codeocean.com/capsule/0235982`
[2] `https://github.com/gamer1882/FDC`

Table 2: Synthetic and real-world data sets.

Data set	Pathbased	Compound	orl	breast	brea	baobab	worldmap	mfeat-kar	segm	baldder	cic-ids	mnist	pendigits	letter-recognition
#Instance	300	399	400	683	699	900	935	2000	2100	2486	4219	10000	10992	20000
#Attribute	2	2	4096	9	9	892	899	64	19	512	77	768	16	16
#Cluster	3	6	40	2	2	3	3	10	7	4	12	10	10	26

In terms of data sets, we selected two synthetic data sets and twelve real-world data sets including image data sets (orl, mfeat-kar, segm, mnist, pendigits, and letter-recognition)[3], medical data sets (breast and brea)[4], bioinformation data sets (bladder)[5], cybersecurity data sets (cic-ids)[6], and two data sets (baobab and worldmap) by Microsoft[7]. The details are shown in Table 2. For non-image data sets, the z-score method is used for normalization. Pictures whose pixel values in range $[0, 255]$ is scaled to $[0, 1]$.

For each dataset, in order to show the results of the algorithms under different numbers of pairwise constraints, we generate pairwise constraints with five different numbers: $0.1n$, $0.2n$, $0.3n$, $0.4n$, $0.5n$ (n represents the number of samples). Must-link and cannot-link constraints are generated by randomly choosing two samples and check their ground-truth labels: if they have the same label, then they are taken as a must-link constraint; otherwise, they are considered as a cannot-link constraint. Ten groups of pairwise constraint information are randomly generated for each number. Note that for any data set containing n samples, the number of constraints can reach the order of n^2. In contrast, the constraint numbers selected by us are relatively small, which is consistent with the assumption of semi-supervised constraints and practical application scenarios. All data sets and groups of pairwise constraints are pre-generated and saved, so that all algorithms use the same constraint information.

We select Adjusted Rand Index (ARI), Normalized Mutual Information (NMI) and Accuracy (ACC) for performance evaluation. The values are scaled to display within the range $[0, 100]$. Since ten groups of pairwise constraints are generated for each amount of constraints, we calculate the optimal result for each parameter within each group, and then

[3] `https://archive.ics.uci.edu/dataset`

[4] `https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original`

[5] `https://figshare.com/articles/software/scBGEDA/19657911`

[6] `https://www.unb.ca/cic/datasets/ids-2017.html`

[7] `https://www.microsoft.com/en-us/research/tools/`

take the average of the optimal results from the ten groups as the final result and calculate their standard deviation.

6.2. Results and analysis on synthetic data sets

Since WECR k-means is based on subspace clustering, which is specifically designed for high-dimensional data sets, it is not involved in the comparison conducted on two 2-dimensional data sets in this section. For each data set, we calculate the mean and variance of the clustering indexes for each algorithm for different constraint numbers. Due to space limitations, only the result of the mean ARI is shown here (see Figure 5).

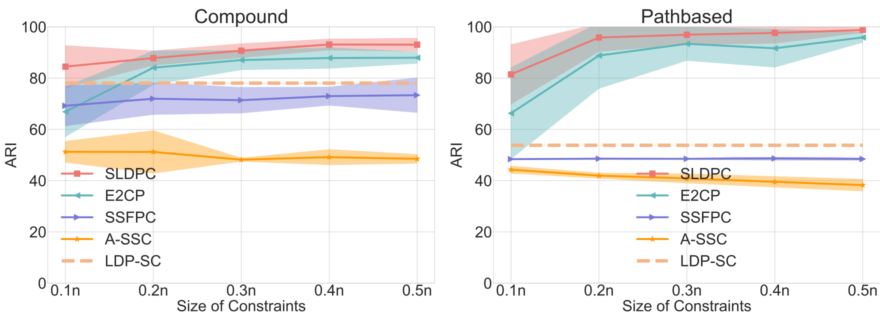

Figure 5: Results on synthetic data sets.

The left of Figure 5 shows that the mean ARI of SLDPC is higher than other comparison algorithms for all constraint numbers on the Compound dataset. Its performance gradually improves with the increase of the constraint number. When the constraint number reaches $0.4n$, its performance tends to be stable, and the mean value of ARI is about 93. The right of figure 5 shows the results of the Pathbased dataset. The mean ARI of SLDPC is also higher than other comparison algorithms globally. For SLDPC, when the constraint number is under $0.3n$, the performance improvement is evident with the increase of the constraint number, and the performance tends to be stable after exceeding this number. When the constraint number reaches $0.5n$, the mean value of ARI reaches 98. At the same time, E2CP reaches 95, which is slightly lower than SLDPC. However, when the constraint number is under $0.3n$, the performance of E2CP is far behind SLDPC.

Figure 6 shows the clustering results of LDP-SC and SLDPC under different numbers of pairwise constraints on the Compound dataset, where the results are selected from ex-

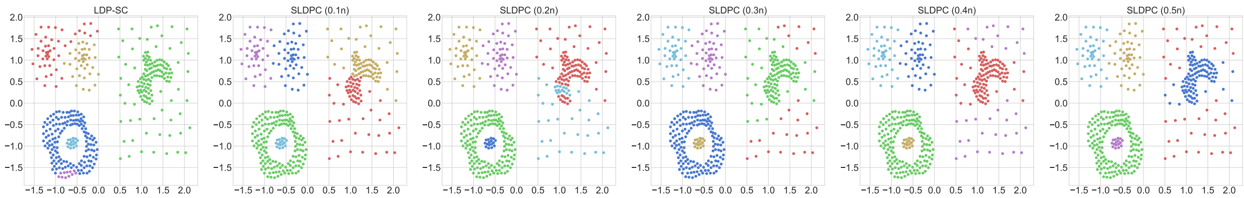

Figure 6: Visualization of results on Compound data set.

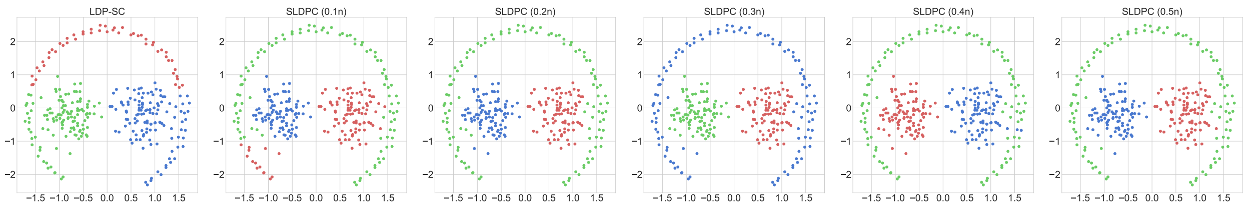

Figure 7: Visualization of results on Pathbased data set.

periments that exhibit outcomes closest to the mean value shown in Figure 5. On the right part of the data set, it can be seen that the original LDP-SC algorithm clusters the two types of data, i.e., data with higher density and data with low density, into one class. Due to the limitation of the number of clusters, a small part in the lower left corner is cut out as a single class. For SLDPC, when the constraint number is $0.1n$, the right part of the data set is incorrectly divided into two wrong pieces as well. With the increase of the constraint number, the division of the right area tends to be correct. Since good clustering results rely on the prerequisite of having no conflicts within the generated tree, we can infer that intra-cluster conflict resolution and inter-cluster conflict resolution have played a crucial role in enhancing the purity of the trees.

Figure 7 shows the clustering result of LDP-SC and SLDPC on the Pathbased data set under different constraint numbers. The banded area in the outermost layer of the data set cannot be correctly identified by DPC-SC. In SLDPC, when the constraint number is only $0.1n$, the banded area on the right side of the dataset is basically identified except for the data in the left corner. When the constraint number is above $0.2n$, the performance of SLDPC is already close to the optimal clustering result. For this data set, SLDPC can greatly enhance the performance of clustering with only a small amount of constraint information.

Table 3: Clustering results on smaller real-world data sets (ACC/NMI/ARI).

Data	Size	LDP-SC	SSFPC	WECR k-means	E2CP	A-SSC	TLRR	SLDPC
	$0.1n$	**64.75**/82.87/**50.92**	24.30/14.25/36.84	N/A	64.47/82.12/48.91	N/A	52.69/73.12/37.24	63.75/**83.09**/50.90
	$0.2n$	64.75/82.87/**50.92**	24.12/14.00/35.68	N/A	**65.95**/82.56/50.31	N/A	50.81/72.66/36.31	63.73/**83.09**/50.87
orl	$0.3n$	64.75/82.87/50.92	27.16/14.00/35.96	N/A	64.67/81.84/49.70	N/A	54.94/74.07/39.44	**65.12**/**83.11**/**51.26**
	$0.4n$	64.75/82.87/50.92	34.29/12.50/34.06	N/A	65.45/82.48/49.95	N/A	53.25/73.36/38.22	**65.65**/**83.34**/**51.61**
	$0.5n$	64.75/82.87/50.92	34.29/12.50/34.06	N/A	65.55/82.49/51.18	N/A	51.31/72.57/36.51	**66.25**/**83.52**/**52.56**
	$0.1n$	96.19/75.96/85.18	96.88/78.99/87.81	95.92/74.18/84.16	**97.07**/**79.84**/**88.53**	96.59/77.48/86.68	83.42/44.58/44.61	95.46/73.12/82.49
	$0.2n$	96.19/75.96/85.18	96.5/76.9/86.37	96.0/74.57/84.48	97.04/**79.67**/**88.41**	**97.13**/80.44/88.73	82.76/43.07/42.86	95.99/75.48/84.46
breast	$0.3n$	96.19/75.96/85.18	96.43/76.63/86.09	96.03/74.7/84.59	97.13/80.15/88.75	**97.61**/**83.07**/**90.59**	82.58/42.28/42.38	97.53/82.53/90.26
	$0.4n$	96.19/75.96/85.18	95.84/73.91/83.88	95.96/74.37/84.32	97.14/80.21/88.8	**97.69**/**83.35**/**90.88**	82.54/42.28/42.28	97.58/83.0/90.49
	$0.5n$	96.19/75.96/85.18	96.0/74.73/84.48	96.0/74.55/84.49	97.41/81.76/89.81	**98.10**/**85.94**/**92.46**	83.35/44.34/44.41	97.79/84.03/91.28
	$0.1n$	95.99/74.78/84.42	96.62/77.69/86.83	95.89/73.78/84.07	**96.75**/**78.19**/**87.31**	96.65/77.64/86.91	82.33/42.75/41.74	95.49/72.95/82.58
	$0.2n$	95.99/74.78/84.42	96.01/74.57/84.52	95.99/74.23/84.45	96.68/77.72/87.03	**96.78**/**78.43**/**87.39**	82.22/41.55/41.46	96.55/77.52/86.56
brea	$0.3n$	95.99/74.78/84.42	95.79/73.5/83.7	96.02/74.37/84.55	96.77/78.19/87.36	**97.68**/**83.27**/**90.85**	83.19/44.50/44.01	97.17/80.46/88.88
	$0.4n$	95.99/74.78/84.42	95.62/72.64/83.05	96.09/74.71/84.82	96.72/77.94/87.2	**97.88**/**84.46**/**91.62**	82.33/42.21/41.76	97.47/82.08/90.03
	$0.5n$	95.99/74.78/84.42	95.62/72.64/83.05	95.99/74.23/84.45	97.04/79.68/88.39	**98.11**/**85.76**/**92.51**	82.80/42.67/42.97	97.81/84.21/91.36
	$0.1n$	**65.22**/5.83/9.93	44.34/1.85/0.99	53.98/5.77/9.44	62.9/**8.48**/**18.57**	62.04/5.48/13.83	41.00/3.96/2.81	64.54/6.18/13.6
	$0.2n$	65.22/5.83/9.93	46.07/2.88/1.49	53.6/5.64/9.84	62.96/8.45/16.42	60.92/6.17/12.79	39.69/2.94/2.63	**66.61**/**10.51**/**20.95**
baobab	$0.3n$	65.22/5.83/9.93	51.38/1.64/0.95	53.61/5.53/8.72	64.69/8.99/18.36	61.11/6.80/13.66	41.64/3.61/4.14	**66.87**/**12.21**/**26.39**
	$0.4n$	65.22/5.83/9.93	47.93/2.84/1.61	54.39/5.82/10.47	66.6/11.05/20.86	62.02/7.70/15.00	45.97/2.76/4.34	**68.38**/**14.61**/**29.26**
	$0.5n$	65.22/5.83/9.93	48.77/4.78/5.41	53.79/5.49/9.37	67.18/12.37/24.43	61.76/7.54/15.29	48.11/2.30/3.53	**70.0**/**18.83**/**33.8**
	$0.1n$	61.71/6.89/11.93	47.34/2.54/0.99	57.54/6.61/13.93	**71.27**/**9.98**/**16.07**	61.65/0.61/3.87	41.42/3.84/2.00	62.65/7.93/14.24
	$0.2n$	61.71/6.89/11.93	46.96/2.83/1.34	58.82/6.54/14.38	**70.65**/**10.73**/18.16	60.83/0.82/4.47	40.96/3.74/1.80	66.51/10.65/**19.74**
worldmap	$0.3n$	61.71/6.89/11.93	47.6/2.17/1.36	57.85/6.87/14.35	71.56/10.59/16.12	62.46/1.52/6.54	41.60/4.28/1.67	**71.83**/**12.56**/**24.62**
	$0.4n$	61.71/6.89/11.93	49.36/1.76/0.64	58.24/6.98/14.68	68.45/10.07/16.17	61.78/1.31/5.83	41.10/3.94/1.26	**74.82**/**16.59**/**32.88**
	$0.5n$	61.71/6.89/11.93	53.21/1.99/1.6	58.0/6.92/14.59	72.05/12.16/17.36	62.18/1.65/6.68	41.20/4.68/1.90	**75.43**/**18.4**/**34.84**
	$0.1n$	84.6/84.39/**77.89**	26.64/20.67/10.31	78.57/78.88/70.83	81.6/**84.58**/75.95	69.88/70.33/59.10	71.30/69.62/60.89	**85.96**/82.23/77.51
	$0.2n$	84.6/84.39/**77.89**	26.9/20.5/10.14	78.52/78.79/70.76	81.32/**84.59**/75.76	72.06/71.64/61.11	80.49/72.78/66.96	**85.74**/82.74/77.84
mfeat-kar	$0.3n$	84.6/84.39/77.89	25.28/19.25/9.47	78.51/78.76/70.73	83.09/**86.29**/78.27	74.91/72.71/62.97	73.04/71.17/62.99	**86.88**/83.35/**78.97**
	$0.4n$	84.6/84.39/77.89	27.06/20.6/10.35	78.48/78.77/70.72	82.95/**86.63**/78.46	72.76/71.98/61.69	71.78/71.80/62.99	**88.6**/84.33/**80.62**
	$0.5n$	84.6/84.39/77.89	27.39/20.94/10.56	78.53/78.83/70.8	83.26/**87.0**/79.09	70.22/70.67/59.60	80.88/73.94/69.18	**89.99**/84.99/**81.89**
	$0.1n$	51.9/52.59/36.73	64.66/61.03/51.0	60.63/64.99/50.52	56.9/63.75/46.86	54.47/53.78/37.82	68.14/62.00/53.15	**68.35**/**67.62**/**56.06**
	$0.2n$	51.9/52.59/36.73	64.16/61.11/50.5	60.6/64.96/50.53	60.68/68.67/53.0	56.33/54.29/39.50	70.14/60.97/54.03	**75.55**/**72.84**/**64.47**
segm	$0.3n$	51.9/52.59/36.73	63.9/60.94/50.17	60.63/65.0/50.55	65.09/72.85/57.07	57.76/54.89/40.21	71.83/62.13/54.87	**77.9**/**76.45**/**68.38**
	$0.4n$	51.9/52.59/36.73	65.28/61.8/51.48	60.64/65.03/50.57	71.87/76.48/63.23	58.71/54.66/40.40	70.88/60.32/53.54	**83.0**/**78.17**/**72.62**
	$0.5n$	51.9/52.59/36.73	64.3/61.33/50.71	60.6/64.99/50.54	71.55/77.66/64.12	56.73/53.31/39.48	68.26/60.39/51.60	**86.04**/**80.34**/**75.69**
	$0.1n$	92.68/81.87/89.36	61.9/10.53 /7.63	41.28/10.65/10.62	49.74/7.95/2.81	71.30/54.34/54.37	82.18/76.45/75.28	**97.13**/**88.10**/**93.98**
	$0.2n$	92.68/81.87/89.36	62.43/9.19 /7.28	40.79/10.19/10.04	49.65/7.69/2.74	71.44/54.71/55.76	79.00/71.63/71.60	**96.44**/**87.32**/**93.16**
bladder	$0.3n$	92.68/81.87/89.36	50.93 / 8.41/ 6.28	39.84/9.63/9.39	49.73/8.56/3.07	71.62/54.99/56.71	79.32/67.64/70.36	**95.82**/**86.82**/**92.73**
	$0.4n$	92.68/81.87/89.36	48.15 /8.56 6.42	40.08/9.56/9.28	49.73/7.76/2.73	71.59/55.18/57.41	79.24/67.80/69.77	**95.26**/**87.04**/**92.20**
	$0.5n$	92.68/81.87/89.36	63.75/10.69/ 7.85	41.22/9.90/9.88	49.85/9.03/3.18	72.03/55.66/58.18	80.19/66.76/71.50	**95.38**/**86.33**/**92.41**

6.3. Results and analysis on real-world data sets

Table 3 and Table 4 show the mean of ACC, NMI and ARI of SLDPC and each comparison algorithm on world-real data sets. If the result cannot be obtained within 5 hours, it is marked as N/A.

From Table 3 and Table 4, we can see that SLDPC has the best performance in most cases. Compared with LDP-SC, SLDPC performs significantly better except for a few data sets, and this trend becomes more and more evident as the constraint number increases. The proposed algorithm further improves the clustering effect on the basis of LDP-SC. A-SSC on orl shows N/A because it cannot deal with this very high-dimensional data. The WERC k-means on orl shows N/A because the limited number of samples makes it infeasible to generate sub-labels.

For the first eight smaller data sets, a Friedman test was conducted on seven comparison methods with accuracy (ACC) as an example (the results for ARI and NMI are similar). The statistic value for this test was 130.16, and the P-value was 1.19×10^{-25}. This indicates that at a significance level of 0.05, there are statistically significant differences among these

Table 4: Clustering results on larger real-world data sets (ACC/NMI/ARI).

Data	Size	LDP-SC	WECR k-means	E2CP	A-SSC	SLDPC
cic-ids	$0.1n$	54.61/63.04/37.49	54.96/65.55/42.69	66.51/70.45/54.21	40.28/47.21/25.75	**70.61**/**74.26**/**62.02**
	$0.2n$	54.61/63.04/37.49	56.89/66.55/44.18	70.33/74.56/62.03	40.78/47.22/27.26	**72.26**/**75.86**/**62.57**
	$0.3n$	54.61/63.04/37.49	53.93/63.04/37.38	**76.44**/**77.75**/**67.93**	40.43/47.54/28.27	75.30/77.67/65.10
	$0.4n$	54.61/63.04/37.49	56.10/65.62/43.07	75.53/**78.54**/**68.40**	38.87/45.10/26.93	**75.62**/77.91/67.33
	$0.5n$	54.61/63.04/37.49	56.19/66.19/44.26	76.84/**79.07**/**69.10**	38.76/44.01/26.18	**78.15**/78.39/68.74
mnist	$0.1n$	77.22/75.63/68.5	65.34/68.51/56.35	74.13/77.16/66.63	54.56/50.54/38.67	**88.55**/**80.32**/**78.37**
	$0.2n$	77.22/75.63/68.5	64.96/68.39/56.14	80.61/**82.58**/75.25	56.55/51.36/40.21	**89.0**/79.9/**78.05**
	$0.3n$	77.22/75.63/68.5	64.52/68.28/55.95	85.08/**84.54**/79.46	55.89/51.30/39.78	**89.83**/80.97/**79.53**
	$0.4n$	77.22/75.63/68.5	64.59/68.33/56.04	88.1/**85.65**/**82.15**	55.47/50.95/39.31	**90.2**/81.7/80.36
	$0.5n$	77.22/75.63/68.5	64.66/68.38/56.12	**91.87**/**87.27**/**85.95**	57.16/52.21/41.04	91.03/82.44/81.65
pendigits	$0.1n$	89.3/86.31/80.04	73.18/75.86/60.45	88.41/84.68/78.3	69.41/66.70/53.88	**91.23**/**91.04**/**86.21**
	$0.2n$	89.3/86.31/80.04	73.19/75.88/60.48	87.17/92.54/84.69	69.41/66.35/53.82	**94.0**/**93.63**/**90.74**
	$0.3n$	89.3/86.31/80.04	73.17/75.85/60.43	95.49/94.41/92.67	69.51/66.21/53.92	**98.27**/**96.05**/**96.22**
	$0.4n$	89.3/86.31/80.04	73.17/75.85/60.44	94.12/94.74/91.88	69.52/66.02/53.89	**97.66**/**96.42**/**96.03**
	$0.5n$	89.3/86.31/80.04	73.17/75.85/60.44	95.37/95.26/93.42	69.53/65.84/53.83	**96.77**/**96.41**/**95.3**
letter-recognition	$0.1n$	**31.25**/**46.16**/6.66	27.41/40.34/11.80	N/A	24.65/35.86/**12.93**	31.06/45.05/12.49
	$0.2n$	31.25/46.16/6.66	27.18/40.33/11.76	N/A	24.39/35.82/12.84	**38.10**/**52.34**/**19.08**
	$0.3n$	31.25/46.16/6.66	26.42/40.00/11.41	N/A	24.66/35.86/13.11	**43.12**/**55.59**/**26.30**
	$0.4n$	31.25/46.16/6.66	27.29/40.39/11.91	N/A	25.25/35.78/13.29	**45.66**/**57.27**/**30.43**
	$0.5n$	31.25/46.16/6.66	27.37/40.15/11.71	N/A	25.39/35.55/13.49	**53.48**/**62.45**/**37.60**

methods. Further Nemenyi post-hoc tests showed that, compared pairwise, SLDPC demonstrated statistically significant advantages over WECR k-means, SSFPC, A-SSC, TLRR, and LDP-SC, with P-values all less than 0.01. However, for E2CP, the significance level was 0.66. This is mainly due to the fact that the Nemenyi test is primarily based on the rank of methods. In the first two datasets (breast and brea) where all methods performed relatively well, E2CP ranked higher, and in subsequent datasets, E2CP also maintained high rankings. As a result, the ranking-based analysis did not clearly show the advantage of SLDPC over E2CP.

Nevertheless, looking at the specific metric values, our method exhibits clear advantages over E2CP on segm, bladder, pendigits, and letter-recognition data sets. Also, for larger datasets, such as Pendigits, mnist and letter-recognition, E2CP tends to be notably slower. In particular, for the letter-recognition dataset, E2CP failed to produce results within 5 hours.

7. Discussion

7.1. Runtime comparison

Table 5 and Table 6 present the runtimes (in seconds) of the compared algorithms. In Table 5, the constraint number is set to be $0.3n$, and it compares the efficiency on data sets with a variety of sizes and dimensions. Table 6 demonstrates how the runtimes of these

Table 5: Runtime comparison on various data sets (in seconds).

Algorithms	**breast**	**mfeat-kar**	**cic-ids**	**pendigits**
#Constraint= $0.3n$	$n = 683$, $d = 9$	$n = 2000$, $d = 64$	$n = 4219, d = 77$	$n = 10992, d = 16$
LDP-SC	0.44	0.48	0.89	0.98
SSFPC	47.83	2714	N/A	N/A
WECR k-means	4.38	5.88	6.71	16.21
E2CP	0.3	1.13	23.7	70.75
A-SSC	0.07	37.93	272.67	153.59
TLRR	171.8	6906	N/A	N/A
SLDPC	0.66	1.81	2.7	2.81

Table 6: Runtime comparison on cic-ids with various numbers of pairwise constraints (in seconds).

Algorithms	$0.1n$	$0.2n$	$0.3n$	$0.4n$	$0.5n$
LDP-SC	1.17	1.17	1.17	1.17	1.17
WECR k-means	2.02	2.45	2.35	2.27	2.35
E2CP	28.85	25.65	22.51	15.32	9.8
A-SSC	276.77	544.48	372.67	441.64	559.9
SLDPC	1.19	1.25	1.33	2.28	3.57

algorithms vary with increasing constraint numbers. It can be seen that the increment in runtime of SLDPC is small compared to the original LDP-SC algorithm and is competitive compared to other algorithms. For example, for cic-ids, the runtime of SLDPC is 2.7 seconds, and that of LDP-SC is 0.89 seconds, while A-SSC is 272.67 seconds and SSFPC and TLRR take too long (more than 5 hours) to obtain results. A similar phenomenon can also be observed on the pendigits data set. As shown in Table 6, when the number of pairwise constraints increases, the runtime of SLDPC increases in an acceptable scale, i.e., from 1.19 seconds to 3.57 seconds, which is much shorter than that of A-SSC (over 200 seconds) and is consistently shorter than E2CP.

7.2. Robustness

For clustering algorithms, the performance evaluation criteria should not only consider the best state, but also refer to the stability under different parameters.

Figure 8 shows the mean and standard deviation of ARI of each data set under different constraint numbers when the nearest neighbor number k is taken from 2 to 30. It can be seen

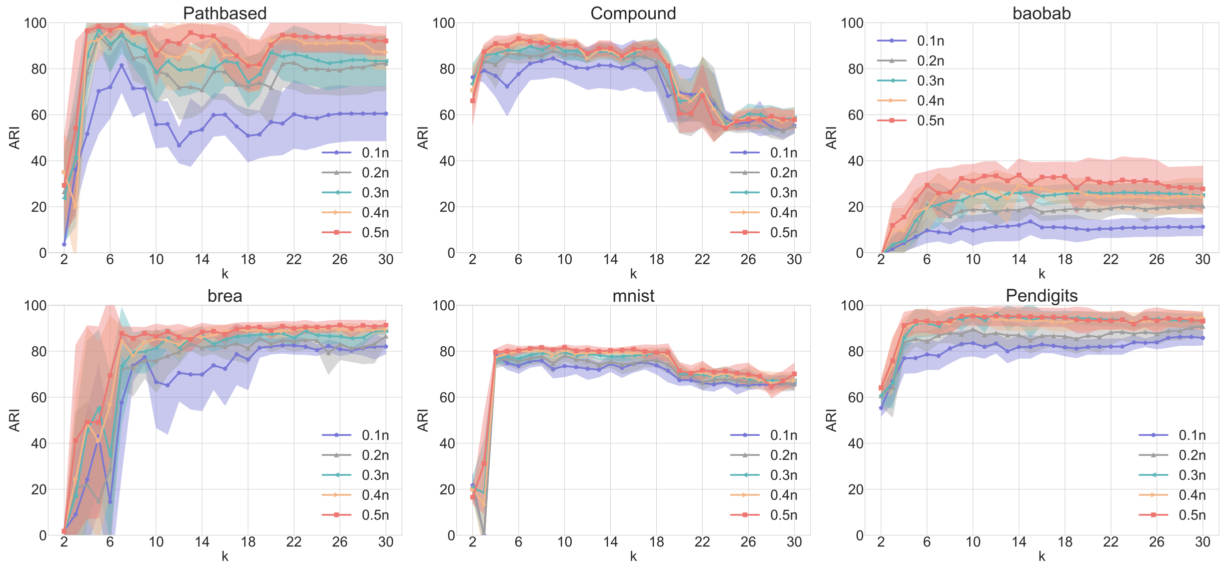

Figure 8: Robustness of SLDPC.

that k around 6 can be regarded as a cut-off point. When $k < 6$, the algorithm fluctuates greatly, and the overall result is not good. As k gradually increases from 2 to 6, the ARI shows an obvious increasing trend. When $k > 6$, the performance of the algorithm tends to be stable. It is worth noting that as the constraint number gradually increases from $0.1n$ to $0.5n$, the ARI shows an increasing trend on all data sets, and the ARI curves become smoother, especially on the Pathbased data set. In other words, as the constraint number increases, the algorithm becomes more robust.

7.3. Growth trend of performance

In this section, we select four data sets (baobab, worldmap, semg, mfeat-kar) that still have a large room for performance improvement, to demonstrate the trend of the performance with larger constraint sizes. The range of the constraint numbers is enlarged from the previous $0.1n \sim 0.5n$ to $0.1n \sim 3n$. For each constrain number, we randomly generate ten groups of pairwise constraints, and calculate the mean and standard deviation of ARI.

Figure 9 depicts the ARI curve of the experiment. The performance of SLDPC continues to improve and becomes relatively stable after around $1.5n$. This provides us an empirical experience that increasing the number of pairwise constraint information can effectively

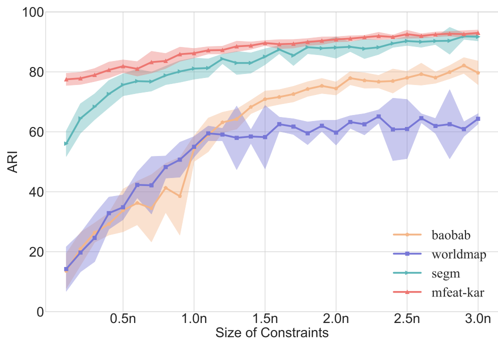

Figure 9: Growth trend of performance of SDLPC.

improve the clustering performance of SLDPC while maintaining good rate of improvement before reaching the constraint size of $1.5n$.

7.4. Analysis of the process of clustering

In this part, we visualize the clustering process of SLDPC on the Pathbased dataset to analyze how each step of the proposed algorithm improves the final result. The number of nearest neighbors is set as $k = 9$.

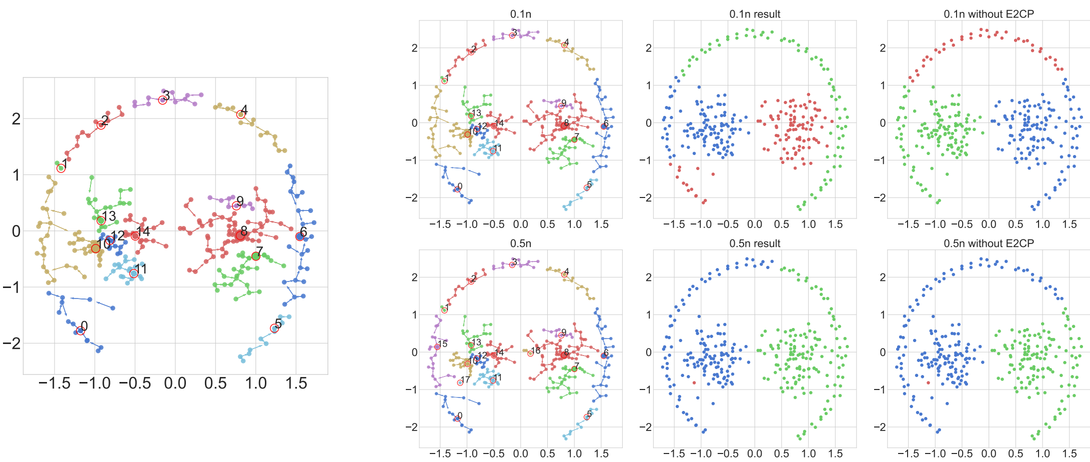

Figure 10: Initial family trees and effects of intra-cluster conflict resolution.

Figure 10 (left) shows the constructed initial family trees at the first step of SLDPC, which are numbered from 0 to 14. It can be clearly seen that there is a clustering error inside the No.10 tree: some points in the banded area and some points in the the block area are merged into a same tree.

Figure 10 (right) shows the result of family trees after intra-cluster conflict resolution with $0.1n$ and $0.5n$ constraint numbers. By observing the figure, we notice that as the constraint number increases, some errors are corrected and the clustering result becomes better, e.g., for $0.5n$ constraint number, the original No.10 tree is split into three subtrees with higher purity: the No.10, No.15 and No.17 trees.

Although the purity of the trees is not improved when the constraint number is $0.1n$, we can see from the middle column and the rightmost column that the constraint information has positive effects on the E2CP adjustment similarity stage. There are still some errors that are due to wrong trees and incorrect clustering of small subtrees. For example, the banded area on the right is correctly divided after E2CP, but due to the incorrect No.10 tree, the banded area on the left is still not identified correctly. In the $0.5n$ case, the No.17 subtree is a single sample point, which will be divided into a separate class at the stage of graph cut.

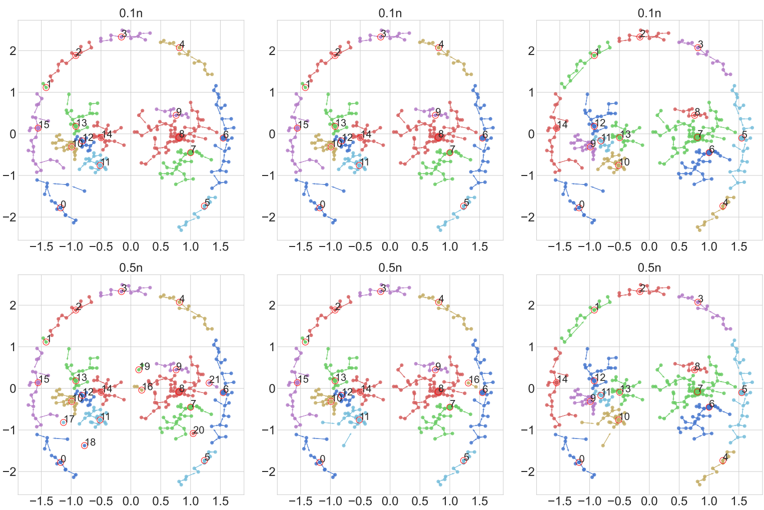

Figure 11: Effects of inter-cluster conflict resolution, root node redirection and noise filtering.

In SLDPC, the inter-cluster conflict resolution, root node redirection and noise filtering steps are performed after the intra-cluster conflict resolution. The left, middle and right columns of Figure 11 respectively show the results of the subsequent inter-cluster conflict resolution, root node redirection and noise filtering of SLDPC under $0.1n$ and $0.5n$ constraint number based on Figure 10.

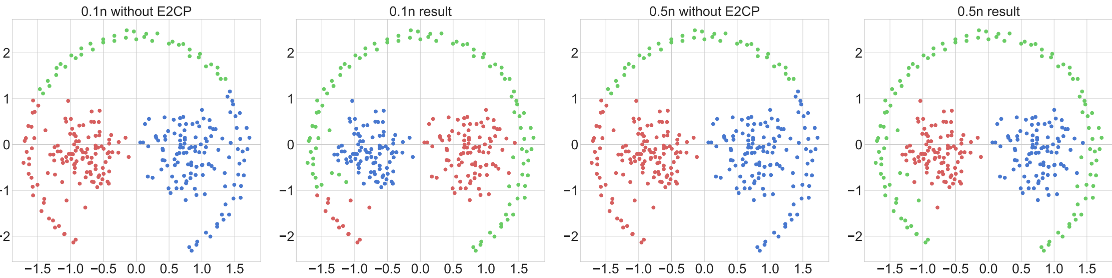

Figure 12: Clustering results with and without E2CP.

Compared with Figure 10, the purity of trees is further improved. For example, in the $0.5n$ case, the split that results in the No.18 and No.21 trees effectively resolved the conflicts, but more fragmented trees appeared. In the root node redirection stage, this situation has been greatly mitigated. Finally, after the noise filtering step, the fragmented trees no longer exist, and the purity of the obtained trees is maintained.

After completing all the steps illustrated in Figure 11, the subtree aggregation step is finally executed. Figure 12 shows the clustering result of with and without the similarity adjustment step by E2CP method based on Figure 11. It can be seen that for both the number of $0.1n$ and $0.5n$, E2CP improves the final result significantly, which has once again confirmed the effectiveness and necessity of this step.

8. Conclusion

We propose a semi-supervised clustering algorithm SLDPC. SLDPC utilizes pairwise constraint information to improve LDP-SC in two aspects: improving the purity of local trees and optimizing the original similarity matrix of local trees. SLDPC formulates the intra-cluster conflict resolution and inter-cluster conflict resolution steps to ensure that the established local trees are consistent with the pairwise constraint information. In order to avoid the trees being too fragmented, root node redirection and noise filtering steps are then designed to make the trees as large and pure as possible. In terms of similarity matrix adjustment, SLDPC employs the E2CP algorithm to optimize the similarity matrix. Experiments on twelve commonly used real-world datasets and two synthetic datasets show that the proposed algorithm is superior to other prominent clustering algorithms in most cases. The robustness analysis of hyper-parameters also shows that SLDPC has stable performance. While SLDPC demonstrates superior performance, it faces challenges when handling large datasets due to its complexity, and its density-based strategy limits its effectiveness for high-dimensional data. To explore a more efficient method of splitting trees for the inter-cluster conflict resolution stage and to devise similarity measures that characterized high-dimensional data more effectively may help further improve the algorithm.

Declaration of Competing Interest

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

Acknowledgment

This work was supported by the National Natural Science Foundation of China [grant numbers 12231007, 11871353, 61806170], and the Fundamental Research Funds for the Central Universities [grant numbers 2682022ZTPY082, 2682023ZTPY027].

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] G. Yang, S. Deng, X. Chen, C. Chen, Y. Yang, Z. Gong, Z. Hao, Reskm: A general framework to accelerate large-scale spectral clustering, Pattern Recognition 137 (2023) 109275.

[7] Y. Shi, Z. Chen, Z. Qi, F. Meng, L. Cui, A novel clustering-based image segmentation via density peaks algorithm with mid-level feature, Neural Computing and Applications 28 (2017) 29–39.

[8] L. Sun, T. Ye, J. Sun, X. Duan, Y. Luo, Density-peak-based overlapping community detection algorithm, IEEE Transactions on Computational Social Systems 9 (2021) 1211–1223.

[9] Z.-H. Deng, H.-H. Qiao, M.-Y. Gao, Q. Song, L. Gao, Complex network community detection method by improved density peaks model, Physica A: Statistical Mechanics and its Applications 526 (2019) 121070.

[10] B. Zhou, M. Shang, L. Feng, K. Shan, L. Feng, J. Ma, X. Liu, L. Wu, Long-term remote tracking the dynamics of surface water turbidity using a density peaks-based classification: A case study in the three gorges reservoir, china, Ecological Indicators 116 (2020) 106539.

[11] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496.

[12] Q. Zhang, Y. Dai, G. Wang, Density peaks clustering based on balance density and connectivity, Pattern Recognition 134 (2023) 109052.

[13] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, Advances in Neural Information Processing Systems (2002) 849–856.

[14] Z. Long, Y. Gao, H. Meng, Y. Yao, T. Li, Clustering based on local density peaks and graph cut, Information Sciences 600 (2022) 263–286.

[15] J. E. van Engelen, H. H. Hoos, A survey on semi-supervised learning, Machine Learning 109 (2020) 373–440.

[16] Z. Lu, Y. Peng, Exhaustive and efficient constraint propagation: A graph-based learning approach and its applications, International Journal of Computer Vision 103 (2013) 306–325.

[17] S. Basu, A. Banerjee, R. Mooney, Semi-supervised clustering by seeding, in: International Conference on Machine Learning, 2002, pp. 27–34.

[18] H. Liu, Z. Tao, Y. Fu, Partition level constrained clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (2017) 2469–2483.

[19] B. Mirkin, Reinterpreting the category utility function, Machine Learning 45 (2001) 219–228.

[20] F. Salehi, M. R. Keyvanpour, A. Sharifi, Smkfc-er: Semi-supervised multiple kernel fuzzy clustering based on entropy and relative entropy, Information Sciences 547 (2021) 667–688.

[21] Z. Jiang, Y. Zhan, Q. Mao, Y. Du, Semi-supervised clustering under a compact-cluster assumption, IEEE Transactions on Knowledge and Data Engineering 35 (2022) 5244–5256.

[22] K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, C. Christopher, Spectral learning, in: International Joint Conference of Artificial Intelligence, 2003, pp. 561–566.

[23] Z. Lu, M. A. Carreira-Perpinan, Constrained spectral clustering through affinity propagation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[24] Z. Li, J. Liu, X. Tang, Pairwise constraint propagation by semidefinite programming for semi-supervised classification, in: International Conference on Machine learning, 2008, pp. 576–583.

[25] F. Nie, H. Zhang, R. Wang, X. Li, Semi-supervised clustering via pairwise constrained optimal graph,

in: International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 3160–3166.

[26] F. Nie, X. Wang, M. I. Jordan, H. Huang, The constrained laplacian rank algorithm for graph-based clustering, in: AAAI Conference on Artificial Intelligence, 2016, pp. 1969–1976.

[27] Z. Wang, S.-S. Wang, L. Bai, W.-S. Wang, Y.-H. Shao, Semi-supervised fuzzy clustering with fuzzy pairwise constraints, IEEE Transactions on Fuzzy Systems 30 (2021) 3797–3811.

[28] Z. Yu, P. Luo, J. You, H.-S. Wong, H. Leung, S. Wu, J. Zhang, G. Han, Incremental semi-supervised clustering ensemble for high dimensional data clustering, IEEE Transactions on Knowledge and Data Engineering 28 (2015) 701–714.

[29] Z. Yu, P. Luo, J. Liu, H.-S. Wong, J. You, G. Han, J. Zhang, Semi-supervised ensemble clustering based on selected constraint projection, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 2394–2407.

[30] Y. Lai, S. He, Z. Lin, F. Yang, Q. Zhou, X. Zhou, An adaptive robust semi-supervised clustering framework using weighted consensus of random k k-means ensemble, IEEE Transactions on Knowledge and data engineering 33 (2019) 1877–1890.

[31] F. Yang, X. Li, Q. Li, T. Li, Exploring the diversity in cluster ensemble generation: Random sampling and random projection, Expert Systems with Applications 41 (2014) 4844–4866.

[32] P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.

[33] L. Bai, J. Liang, F. Cao, Semi-supervised clustering with constraints of different types from multiple information sources, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2020) 3247–3258.

[34] W. Xia, H. Lu, Q. Wang, A. Tripathi, Y. Huang, I. L. Moreno, H. Sak, Turn-to-diarize: Online speaker diarization constrained by transformer transducer speaker turn detection, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2022, pp. 8077–8081.

[35] A. M. Bagirov, S. Taheri, F. Bai, F. Zheng, Nonsmooth optimization-based model and algorithm for semisupervised clustering, IEEE Transactions on Neural Networks and Learning Systems 34(9) (2023) 5517–5530.

[36] Y. Jia, G. Lu, H. Liu, J. Hou, Semi-supervised subspace clustering via tensor low-rank representation, IEEE Transactions Circuits Systems for Video Technology 33 (2023) 3455–3461.