RESEARCH ARTICLE

A Density Peaks Clustering Algorithm With Sparse Search and K-d Tree

YUNXIAO SHAN [1], SHU LI [1,2], FUXIANG LI [1], YUXIN CUI [1], SHUAI LI [3], MING ZHOU [1], AND XIANG LI [1]

[1] School of Science, Harbin University of Science and Technology, Harbin 150080, China
[2] Key Laboratory of Engineering Dielectric and Applications (Ministry of Education), School of Electrical and Electronic Engineering, Harbin University of Science and Technology, Harbin 150080, China
[3] School of Materials Science and Engineering, Harbin University of Science and Technology, Harbin 150080, China

Corresponding authors: Shu Li (lishu@hrbust.edu.cn) and Fuxiang Li (lifx2013@163.com)

ABSTRACT Density peaks clustering has become a nova of clustering algorithm because of its simplicity and practicality. However, there is one main drawback: it is time-consuming due to its high computational complexity. Herein, a density peaks clustering algorithm with sparse search and K-d tree is developed to solve this problem. Firstly, a sparse distance matrix is calculated by using K-d tree to replace the original full rank distance matrix, so as to accelerate the calculation of local density. Secondly, a sparse search strategy is proposed to accelerate the computation of relative-separation with the intersection between the set of k nearest neighbors and the set consisting of the data points with larger local density for any data point. Furthermore, a second-order difference method for decision values is adopted to determine the cluster centers adaptively. Finally, experiments are carried out on datasets with different distribution characteristics, by comparing with other six state-of-the-art clustering algorithms. It is proved that the algorithm can effectively reduce the computational complexity of the original DPC from $O(n^2K)$ to $O(n(n^{1-1/K} + k))$. Especially for larger datasets, the efficiency is elevated more remarkably. Moreover, the clustering accuracy is also improved to a certain extent. Therefore, it can be concluded that the overall performance of the newly proposed algorithm is excellent.

INDEX TERMS Density peaks clustering, sparse search strategy, K-d tree, computational complexity, second-order difference method.

I. INTRODUCTION

Cluster analysis as an important exploration technology of data mining, is committed to reveal the inherent attributes and laws hidden behind the seemingly disorganized unknown data [1]–[3]. It provides support for decision-making and has been successfully applied in many fields such as image pattern recognition, social network mining, market statistical analysis, medical research and engineering systems [4]–[13]. With the extremely strong penetration and rapid development of the Internet, many professional fields are

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Wang.

faced with explosive growth of data storage. This leads to high computational complexity and difficulty in mining valuable information. In 2014, *Science* published clustering by fast search and find of density peaks (DPC) [14]. Due to its novel design idea and robust performance, DPC instantly became the topic center of scholars in related fields. Compared with classical clustering algorithms [15], [16], DPC possesses several advantages. Firstly, the cluster centers can be identified directly through the decision graph, which consists of local density and relative-separation for all data. Secondly, it can handle non-convex datasets well and no iterative process is required. Furthermore, it is insensitive to outliers. However, there are still some shortages for DPC to be further

improved, including the sensitivity to cut-off distance, the high computational complexity as well as the determination of cluster centers manually.

For the original DPC algorithm published by *Science* [14], the local structure of data [17] is not considered and it is sensitive to the cut-off distance. Some scholars try to solve this problem by integrating the k nearest neighbors with DPC in various ways. Du *et al.* [18] introduced the idea of k nearest neighbors into DPC and redefined the calculation method of local density. Thus, the so-called DPC-KNN algorithm was successfully established. Xie *et al.* [19] also gave another measure of local density using k nearest neighbors and designed a new allocation strategy for non-center points. In order to elevate the possibility of selecting the correct cluster centers, Liu *et al.* [20] modified the distance calculation method by considering the distance factors and neighbor information simultaneously. In addition, a two-step allocation strategy was also adopted to allocate non-center points. Recently, Liu *et al.* [21] have suggested a mixed density clustering method by defining two types of local density. One is based on k nearest neighbors, while the other is determined by local spatial position deviation. Local density description derived from k nearest neighbors supplies a novel way for DPC algorithm. Under this condition, the sensitivity to the hyper-parameter k, i.e. the number of nearest neighbors, is obviously depressed, compared with the sensitivity to cut-off distance used in the original DPC algorithm [14]. However, for these DPC algorithms based on k nearest neighbors, mentioned-above, searching neighbors mainly resorts to violent means. Therefore, these algorithms still have the same high computational complexity as original DPC.

For DPC algorithms, the high computational complexity commonly comes from the distance calculation, which is related with any two points among all data points. With the increase of dataset size, as groundwork, calculating distance between any two points is time-consuming and even impossible for DPC to be implemented. In order to reduce the computational complexity and lift the clustering efficiency, some improvement strategies have been proposed. Gong *et al.* developed the efficient distributed density peaks clustering algorithm (EDDPC) [22], which eliminates unnecessary distance calculation and data shuffling by Voronoi diagram, data replication and data filtering. Bai *et al.* [23] tried to save a part of the computational effort required for distance by combining DPC and K-means. Xu *et al.* introduced the idea of replacing all data points with non-empty grid into DPC [24], [25]. Two prescreening strategies were proposed to determine cluster centers by screening points with the feature of higher local density. Xu *et al.* [26] also proposed a fast density peaks clustering algorithm with sparse search (FSDPC), which introduced the idea of third-party random points to measure the distance. These optimization algorithms have shown good ability in reducing complexity and improve clustering efficiency. However, they either sacrifice accuracy,

resulting in poor clustering effect, or decrease the clustering stability, making the clustering result for each run vary obviously. Therefore, it is not satisfying for these algorithms to get reliable and reasonable clustering results.

As for determining cluster centers, it is difficult for the original DPC algorithm to identify the correct cluster centers manually, if the dividing line between the center points and the non-center points on the decision graph is not very clear. Tong *et al.* [27] obtained the initial clusters by a designed pre-clustering method, and then gave the final clusters according to the Scale Space Theory. Lv *et al.* [28] proposed a method to determine the cluster centers automatically according to the decision value defined by the product of local density and relative-separation. Flores *et al.* [29] proposed a strategy to find cluster centers adaptively by searching the gaps among data points on the one-dimensional decision graph mapped. Lin *et al.* [30] introduced a hyper-parameter, neighbor radius, to select a group of possible density peaks, as preliminary clustering results. Then the final clustering results were formed by merging the clusters with single-bond clustering method. However, some of these algorithms reached the goal of adaptive determination of cluster centers, at the cost of introducing an additional hyper-parameter. Meanwhile, some algorithms became more time-consuming and inefficient, especially for large or complex dataset.

To sum up, various DPC algorithms have been proposed, but very few of these algorithms could achieve satisfactory clustering accuracy, computational complexity as well as the adaptive determination of cluster centers simultaneously. Therefore, the main purpose of the present work is to develop an improved DPC algorithm, in which the computational complexity is reduced significantly and determining cluster centers could be carried out automatically, under the condition that the clustering accuracy is guaranteed without any additional hyper-parameter introduced. As a result, a density peaks clustering algorithm with sparse search and K-d tree (SKTDPC) was proposed. As a type of data structure, K-d tree [31] was adopted to find k nearest neighbors and thus the computational complexity of local density could be firstly reduced remarkably. Secondly, a strategy of sparse search was proposed to accelerate the calculation of relative-separation significantly. Furthermore, the method for automatic determination of cluster centers was described based on second-order difference of the decision value. Besides the main work, the applicability of SKTDPC algorithm is analyzed briefly on higher dimensional datasets.

The rest of this paper is organized as follows. Section II reviews the works related with DPC algorithms. Section III depicts the SKTDPC algorithm newly developed, including fast search for k nearest neighbors, dual acceleration for local density and relative-separation, and determining cluster centers adaptively etc. Section IV verifies the validity of the SKTDPC algorithm and makes a comparison with other state-of-the-art clustering algorithms thoroughly on various

datasets. Finally, Section V is the concluding remarks of the paper.

II. RELATED WORKS

Due to the simple and clear principle of DPC algorithm, it shows great performance on dataset with any shape or dimension. DPC has become a new favorite in the field of clustering algorithm research. The currently proposed SKT-DPC algorithm is designed on the basis of the original DPC, the idea of k nearest neighbors and sparse search. Therefore, this section briefly reviews the original DPC algorithm. The role of the idea for k nearest neighbors and sparse search in DPC are also analyzed.

A. THE ORIGINAL DPC ALGORITHM

In original DPC algorithm [14], there are two important variables, the local density ρ_i and the relative-separation δ_i for any data point x_i in dataset labeled by $S = \{x_1, x_2, \ldots, x_n\}$ with n data.

One of the important variables, the local density ρ_i, is defined as follows:

$$\rho_i = \sum_{j \neq i} \mathcal{X}(d_{i,j} - d_c), \ \mathcal{X}(a) = \begin{cases} 1, & a < 0 \\ 0, & a \geq 0 \end{cases} \quad (1)$$

where d_c is the cut-off distance, which is regarded as a hyper-parameter for the algorithm, and $d_{i,j}$ is the Euclidean distance between data points x_i and x_j.

Another important variable, the relative-separation δ_i, could be determined by searching for the nearest data point x_j, which has a relatively larger local density compared with x_i:

$$\delta_i = \min_{j: \rho_j > \rho_i} (d_{i,j}) \quad (2)$$

This is the condition that the point x_i is a non-maximum local density point. Especially, for the data point with the maximum local density, the relative-separation δ_i is marked specifically by $\delta_{\max}$ and is given as:

$$\delta_{\max} = \max_j (d_{i,j}) \quad (3)$$

DPC selects cluster centers based on the core idea, that the cluster centers are surrounded by these data points with relatively lower ρ_i as well as the cluster centers possess relatively larger distance from points with local density higher than the cluster centers. That is, the data points x_i with relatively large ρ_i and δ_i have a higher probability of being identified as cluster centers. This is the main characteristic of the original DPC algorithm different from other density clustering algorithms.

Based on the density ρ_i and distance δ_i, two critical tasks are performed by the DPC. Firstly, cluster centers are determined. By drawing decision graph in the two-dimensional coordinate system with ρ_i and δ_i as the abscissa and ordinate, respectively, each data point corresponds to a position in the graph and these points in the upper right corner of the decision

graph are selected as cluster centers [32]–[34]. Secondly, non-cluster center points are assigned [35], [36]. The allocation principle is that each non-center point and its nearest point with a higher local density have the same cluster. For the determination of cluster centers, if the boundary between the center points and the non-center points on the decision graph is obvious, the cluster centers could be identified quickly. Otherwise, it is not easy to distinguish the ideal cluster centers manually. This is the problem for determining cluster centers adaptively, which will be solved in the Section III.D.

For the computational complexity of the original DPC, it depends on the Euclidean distance calculation between any two data points in dataset S [37]. These distances construct the symmetric full-rank matrix D:

$$D = \left[d_{i,j} \right]_{n \times n} \quad (4)$$

This is groundwork for further determine both the local density ρ_i and the relative-separation δ_i of each data point x_i. For the dataset $S = \{x_1, x_2, \ldots, x_n\}$ with n data, the computational complexity of DPC is mainly determined by two parts together. One is the computational complexity required to compute the distance matrix, which is $O(n^2)$, while another is the complexity required to allocate the remaining points, which approximately is $O(n)$. Then, the overall computational complexity of DPC is $O(n^2)$.

B. THE IDEA OF K NEAREST NEIGHBORS AND SPARSE SEARCH

From Equation (1), we can see that for the data point x_i the local density ρ_i is defined by the number of the nearest neighbors which are located in the circular area with the cut-off distance d_c as the radius. Thus, Equation (1) can also be reformulated by searching for the set $\mathrm{NN}(x_i)$ which consists of the nearest neighbors of the data point x_i:

$$\mathrm{NN}(x_i) = \{x_j | d_{i,j} < d_c, j \neq i\}$$
$$\rho_i = |\mathrm{NN}(x_i)| \quad (5)$$

where $|\mathrm{NN}(x_i)|$ indicates the number of elements in the set $\mathrm{NN}(x_i)$.

In order to achieve better clustering results, for the original DPC algorithm to be implemented, the cut-off distance d_c should be adjusted as a hyper-parameter. This method defining the local density ρ_i ignores the local distance information of data points and makes the algorithm more sensitive to hyper-parameter. For solving this problem, inspired by the idea of k nearest neighbors, the hyper-parameter d_c could be replaced by the hyper-parameter k, the number of the nearest neighbors. Some scholars have proposed various forms to define the local density ρ_i with k nearest neighbors. In this way, not only the algorithmic sensitivity to hyper-parameter can be depressed, but also the local distance information can be integrated into the local density definition to describe the data density more reasonably. More specifically, d_c has a wide

range of adjustment and has a great impact on algorithm performance. On the contrary, the value of k increases with a step size of 1, and the algorithm can achieve a satisfactory state relatively when k does not exceed 20 generally. Therefore, this strategy to define the local density is still maintained by the present work, but a type of data structure, the K-d tree is adopted to accelerate search for the k nearest neighbors. This will be depicted in Sections III.A and III.B.

It can be found that the calculation of the distance between any two points among all data points is time-consuming at the beginning for DPC algorithms to be implemented. Thus, DPC algorithms have high computational complexity, no matter how to define ρ_i, with the hyper-parameter d_c or the hyper-parameter k, discussed in previous section. In view of this, some scholars have introduced the idea of sparse search into DPC algorithm to reduce the complexity. As the name implies, sparse search refers to a class of methods that can reduce internal friction effectively by replacing a large number of calculations with as little computation as possible. [22]–[26] proposed different sparse methods to speed up the calculation process of DPC. Inspired by this idea, we design a new and effective sparse search strategy to reduce the complexity of DPC.

It should be stressed that the present analysis on computational complexity does not take into account the influence of feature dimension of data, which will be discussed in the Section III.F. In the next section, an algorithm called SKTDPC will be developed, which can significantly reduce the computational complexity by the important strategies of sparse search and K-d tree.

III. THE DENSITY PEAKS CLUSTERING ALGORITHM WITH SPARSE SEARCH AND K-D TREE

In this section, the density peaks clustering algorithm with sparse search and K-d tree (SKTDPC) is proposed. Aiming to reduce the computational complexity, the present algorithm adopts K-d tree to find k nearest neighbors quickly and uses a strategy of sparse search to accelerate the calculation of relative-separation significantly. Moreover, a method for automatic determination of cluster centers is developed. Finally, the process of the algorithm and the complexity are depicted and discussed, respectively.

A. FAST SEARCH OF K NEAREST NEIGHBORS BASED ON K-D TREE

As discussed above, introducing the hyper-parameter k, the number of the nearest neighbors, to replace the hyper-parameter d_c, the cut-off distance, could reduce the algorithmic sensitivity to hyper-parameter obviously. However, for the k nearest neighbors of each data point to be found out, the distance between any two points in dataset S should also be calculated. Therefore, there is still the problem of high computational complexity. To accelerate the search for k nearest neighbors and save a large part of the computational

effort required for distance, as a type of data structure, the classical K-d tree is adopted in the present work.

K-d tree is a typical binary tree, which stores data points in K dimensional space for quick retrieval. It represents the division of K dimensional space and constitutes a series of K dimensional hyperrectangular regions. The fast determination of k nearest neighbors based on K-d tree is divided into two steps. The first part is the K-d tree construction process, the second part is the process of search for k nearest neighbors with K-d tree, the specific process described by Algorithm 1 is as follows.

Note that in order to reduce the cost of backtracking and achieve the best balance of data segmentation, this algorithm selects the dimension with the largest variance and the median in this dimension as dividing criterion each time. The larger the variance is, the more scattered the data points are in this dimension. In the whole paper, the dataset S takes the form of $S = \{x_1, x_2, \ldots, x_n\} \in \mathcal{R}^{n \times K}$, where n and K are the number and spatial dimension of data points respectively. In addition, k represents the number of the nearest neighbors for any data point x_i.

The computational complexity of K-d tree construction is $O(nK \log n)$, and the computational complexity of searching k nearest neighbors of n data points is $O(n(n^{1-1/K} + k))$ in Algorithm 1. Therefore, the overall computational complexity of searching k nearest neighbors by K-d tree is the larger one from $O(nK \log n)$ and $O(n(n^{1-1/K} + k))$, that is, $O(n(n^{1-1/K} + k))$, $K \ll n$. Based on K-d tree, the original computational complexity of searching k nearest neighbors is significantly reduced from $O(n^2)$ to $O(n(n^{1-1/K} + k))$. Thus, a lot of unnecessary calculations for distance are saved. This will further result in the acceleration for calculation of the local density ρ_i (Section III.B).

B. THE ACCELERATION FOR CALCULATION OF THE LOCAL DENSITY BY K-D TREE

Searching for k nearest neighbors is the groundwork for DPC algorithms to be implemented. By using the K-d tree method proposed in Section III.A, the computational complexity could be reduced essentially. As a result, the distance matrix D used in the original DPC (Section II.A), a symmetric full-rank matrix, is changed to the symmetric sparse matrix $\widetilde{D}$. For example, assuming there is a dataset $S = \{x_1, x_2, \ldots, x_{12}\}$ containing 12 elements, the form of the sparse matrix $\widetilde{D}$ may be as follows:

$$\widetilde{D} = \begin{bmatrix} 0 & d_{1,2} & [] & [] & d_{1,5} & d_{1,6} & [] & [] & [] & d_{1,10} & [] & [] \\ d_{1,2} & 0 & [] & [] & [] & d_{2,6} & [] & d_{2,8} & [] & [] & [] & [] \\ [] & [] & 0 & d_{3,4} & [] & [] & d_{3,7} & [] & [] & [] & d_{3,11} & [] \\ [] & [] & d_{3,4} & 0 & d_{4,5} & [] & [] & [] & d_{4,9} & [] & [] & d_{4,12} \\ d_{1,5} & [] & [] & d_{4,5} & 0 & [] & [] & [] & d_{5,9} & [] & [] & [] \\ d_{1,6} & d_{2,6} & [] & [] & [] & 0 & [] & [] & [] & d_{6,8} & [] & d_{6,11} & [] \\ [] & [] & d_{3,7} & [] & [] & [] & 0 & [] & [] & d_{7,10} & [] & d_{7,12} \\ [] & d_{2,8} & [] & [] & [] & d_{6,8} & [] & 0 & [] & d_{8,10} & [] & d_{9,11} & [] \\ [] & [] & [] & d_{4,9} & d_{5,9} & [] & [] & [] & 0 & [] & d_{9,11} & [] \\ d_{1,10} & [] & [] & [] & [] & [] & d_{7,10} & d_{8,10} & [] & 0 & [] & d_{10,12} \\ [] & [] & d_{3,11} & [] & [] & d_{6,11} & [] & [] & d_{9,11} & [] & 0 & [] \\ [] & [] & [] & d_{4,12} & [] & [] & d_{7,12} & [] & [] & d_{10,12} & [] & 0 \end{bmatrix}$$

where $d_{i,j}$ represents the Euclidean distance between the i-th data point and the j-th data point calculated by K-d tree in

---

Algorithm 1: The K-d Tree Determines for k Nearest Neighbors

Input: The dataset $S = \{x_1, x_2, \ldots, x_n\}$, where $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(K)})^T$, $i = 1, 2, \ldots, n$; The number of the nearest neighbors k

Output: K-d tree; The set Q of k nearest neighbors of x_{target}(here x_{target} is any point x_i in dataset S)

build the K-d tree;

while *any leaf node has non-unique data* **do**

    for each dimension $h(h = 1, 2, \ldots, K)$ get its variance;

    choose h according to the maximum variance;

    sort the data values in dimension h and get its median m as the segmentation point;

    store the data point of segmentation on the corresponding node;

    other data are divided by m into two subsets (The left sub-node corresponds to the subset of the coordinates $x^{(h)}$ which is equal to or less than the segmentation point. The right sub-node corresponds to the subset of coordinates $x^{(h)}$ greater than the segmentation point);

Search k nearest neighbors based on constructed K-d tree;

Build an empty set Q;

Recursive visit down K-d tree until leaf node is reached. At this point, the leaf node is the current node (x_{current}) and it is marked as visited;

loop

    calculate the distance $d(x_{\text{target}}, x_{\text{current}})$;

    add the x_{current} to Q;

    if *$|Q| < k$, $|Q|$ indicates the number of elements in the set Q* **then**

        if *there is another child node in the current node* **then**

            recursive visit down K-d tree until leaf node is reached and marked it as visited;

            update the leaf node to the current node;

        else

            find a parent node which has not been visited and mark it as visited;

            update the parent node to the current node;

    else

        break **loop**;

end;

if *the current node is leaf node* **then**

    while *the current node has parent node* **do**

        do Block A:;

        find a parent node which has not been visited and mark it as visited;

        update the parent node to the current node;

        calculate the distance $d(x_{\text{target}}, x_{\text{current}})$;

        if *$d(x_{target}, x_{current}) < Q_{max}$, Q_{max} is the farthest distance from x_{target} in Q* **then**

            replace the point farthest from x_{target} in Q with the current node;

        do Block B:;

        if *there is another child node in the current node* **then**

            calculate the distance $d(x_{\text{target}}, l_{\text{cur-tan}})$, $l_{\text{cur-tan}}$ is the tangent line of the current node;

            if *$d(x_{target}, x_{current}) < Q_{max}$* **then**

                recursive visit down K-d tree until leaf node is reached and marked it as visited;

                update the leaf node to the current node;

                calculate the distance $d(x_{\text{target}}, x_{\text{current}})$;

                if *$d(x_{target}, x_{current}) < Q_{max}$* **then**

                    replace the point farthest from x_{target} in Q with the current node;

else

    while *any leaf node has non-unique data* **do**

        do block B;

        do block A;

the process of searching the k nearest neighbors; "[]" denotes the distance that does not need to be calculated in this process (also known as invalid distance). In this matrix, the first row implies that the algorithm can find out the k nearest neighbors of the data point x_1 only by calculating the distance between x_1 and x_2, x_5, x_6, x_{10}, instead of calculating all the distances.

Taking the number k of the nearest neighbors for any data point x_i as a hyper-parameter, the set of the k nearest neighbors for x_i, calculated by K-d tree, is marked by $\text{NN}_k(x_i)$, which could be described as:

$$\text{NN}_k(x_i) = \{x_j \in S | d(x_i, x_j) \leq d(x_i, N_k(x_i)), j \neq i\} \quad (6)$$

where $d(x_i, x_j)$ is the distance between x_i and x_j, which is equivalent to d_{ij}, and $N_k(x_i)$ is the k-th nearest neighbor for point x_i.

Thus, the following formula is proposed to describe the local density ρ_i more reasonably:

$$\rho_i = \frac{1}{\sum_{x_j \in \text{NN}_k(x_i)} d(x_i, x_j)} \quad (7)$$

It is indicated that the local density ρ_i is not required to calculate on the basis of the symmetric full-rank matrix D, which includes all distances between any two data in dataset S. Therefore, by introducing K-d tree method, the calculation of the local density ρ_i is accelerated significantly with the sparse matrix $\widetilde{D}$. This is the first acceleration for SKTDPC algorithm.

C. THE ACCELERATION FOR CALCULATION OF THE RELATIVE-SEPARATION BY SPARSE SEARCH

In Section III.B, ρ_i is obtained based on the symmetric sparse matrix $\widetilde{D}$. According to the definition of relative-separation, Equation (2), δ_i of the non-maximum density point is determined by searching for the nearest data point x_j, which possesses relatively larger local density ρ_j compared with $x_i(\rho_j > \rho_i)$. In other words, the distance information in $\widetilde{D}$ is not enough to support the acquisition of all δ_i, and it is necessary to complete the rest of the distance calculation. At this point, the computational complexity of getting δ_i will reach $O(n^2)$ in the worst case. Thus, it would be insignificant to save a large part of the distance calculation for ρ_i by K-d tree in Section III.B, if all of distance calculations were carried out between any two points in dataset S. For dealing with this problem, a sparse search strategy is proposed to accelerate the calculation of the relative-separation δ_i.

For expressing more clearly, the local density ρ_i for all of data points obtained in Section III.B are sorted in descending order to generate a new sequence $\rho_i^*(i = 1, 2, \ldots, n)$. For any point x_i, set $B(x_i)$ is introduced to represent the set including all points x_j with greater local density ρ_j compared with $x_i(\rho_j > \rho_i)$. That is to say, consists of these data points corresponding to ρ_1^* to ρ_{i-1}^*. Thus, the set $B(x_i)$ is explicit. In addition, for any point x_i, the set of k nearest neighbors, $\text{NN}_k(x_i)$, has also been determined in Section III.B. Then, for x_i with non-maximum density, based on $\text{NN}_k(x_i)$

and $B(x_i)$, if $\exists j \ s.t. \ \rho_j > \rho_i$, the sparse search strategy for δ_i is defined as:

$$\delta_i = \begin{cases} \min\limits_{x_j \in \text{NN}_k(x_i) \cap B(x_i)} d(x_i, x_j), & \text{when } \text{NN}_k(x_i) \cap B(x_i) \neq \phi \\ \min d(x_i, x_j), & \text{otherwise} \end{cases} \quad (8)$$

When x_i is the point with maximum density, $\delta_{\max}$ has the same expression as that defined by the original DPC, Equation (3).

Equation (8) indicates that both the sets $\text{NN}_k(x_i)$ and $B(x_i)$ are important for determining δ_i quickly. One is the set of k nearest neighbors, while the other is the set of these points with greater local density than ρ_i. When the two sets have an intersection, it means that δ_i must be the minimum of distances of x_i from the point x_j in the intersection. All these distance values are known information determined by looking for k nearest neighbors with K-d tree. They are stored in the sparse matrix $\widetilde{D}$. In this case, δ_i can be obtained directly, without any additional calculation for distance. Under the condition that there is no intersection, the unknown distances between x_i and points with local density greater than ρ_i need to be calculated to find the nearest distance. Therefore, whether the intersection is non-empty or not plays an important role in determining the computational complexity.

Normally, most of δ_i could be obtained by the non-empty intersection directly. This can be explained as follows. Here all data points in dataset S are divided into four parts: points with large ρ_i and large δ_i (part 1), points with large ρ_i and small δ_i(part 2), points with small ρ_i and small δ_i (part 3) as well as points with small ρ_i and large δ_i (part 4). Firstly, there are very few data points with large ρ_i and large δ_i, compared with total data. These points are potential cluster centers. Secondly, for points with large ρ_i, including part 1 and part 2, the total amount of elements in the set $B(x_i)$ is relatively very small, since $B(x_i)$ is defined by these points with local density greater than ρ_i and ρ_i itself is also large. In addition, for the small amount of distance calculation, some of the known distances have already been stored in the symmetric sparse matrix $\widetilde{D}$, except for the distances between x_i and the k nearest neighbors. Thus, even though the intersection is empty, the distance calculation, which needs to be supplemented, is not time-consuming at all. Thirdly, the points in part 3 take a vast proportion of total data. They are non-center points with low local density ρ_i. Small ρ_i implies that the averaged distance between x_i and the k nearest neighbors is obviously large, i.e. the k nearest neighbors are not concentrated around x_i, but are decentralized. Further, the k-th nearest neighbor of x_i is relatively far away from x_i. Meanwhile, small δ_i indicates that the shortest distance from x_i is small for the points x_j with larger local density ρ_j. Thus, for any point x_i, there is a high possibility that the point x_j, which determines the value of δ_i, (with larger local density ρ_j and minimum of distance from x_i) is located in the nearest neighbors set $\text{NN}_k(x_i)$. That is to say the intersection of $\text{NN}_k(x_i)$ and $B(x_i)$ is non-empty very

likely. Finally, the points in part 4 are commonly regarded as outliers, which are also very few. Corresponding calculation required for distance are even negligible actually.

In order to show the process of obtaining δ_i more clearly and intuitively by the proposed sparse search strategy, an example is used to confirm the fact that the intersection of $NN_k(x_i)$ and $B(x_i)$ is not empty in most cases, as shown in Fig. 1. There is a dataset with 16 data points that can be seen in Fig. 1. Firstly, the 3 nearest neighbors set $NN_3(x_i)$ of any sample point x_i and the distance $d(x_i, x_j)$, $x_j \in NN_3(x_i)$ are obtained by the sparse calculation method of K-d tree in Section III.A. Then, the ρ_i is obtained by using (7), and the larger density set $B(x_i)$ of the sample point x_i can be obtained. Finally, the δ_i is obtained from the $NN_3(x_i) \cap B(x_i)$ shown in the last column in Fig. 1. From Fig. 1, we can clearly see that the $NN_3(x_i) \cap B(x_i)$ of sample points 0, 1, 2, 4, 5, 6, 8, 10, 11, 12, 13, 14 and 15 is not empty. This means that the δ_i of these data points can be obtained directly from the $NN_3(x_i) \cap B(x_i)$. That is, the nearest distance from x_i in the $NN_3(x_i) \cap B(x_i)$ is δ_i, without any additional calculation. Only the $NN_3(x_i) \cap B(x_i)$ of 3, 7 and 9 three points is empty, so we need to add a little bit of calculation. It is worth noting that although these three points require supplementary calculation, 3 and 9 only needs to be calculated with a very small number of points in the set $\{7\}$, $\{3, 7\}$ respectively. Only point 7 with maximum density needs to calculate the distance from the remaining points. It can be seen that the proposed sparse search strategy effectively avoids the distance calculation of low-density points, which account for most of the data volume. For a very small proportion of high-density points, even if a little additional calculation is needed, the amount of calculation is very small. For the whole algorithm, it is almost negligible.

On the whole, a very small amount of extra distance needs to be calculated to obtain δ_i of all data points. The sparse search strategy captures the crux of these low density points which occupy the main amount of computation, and obtains δ_i through ingenious intersection strategy directly, so that a lot of distance calculation is saved. At this point, we have succeeded in reducing the computational complexity of this step to much less than $O(n(n^{1-1/K} + k))$. The second acceleration of the SKTDPC algorithm has been achieved. Therefore, the problem of high computational complexity is solved fundamentally, for the original DPC algorithm as well as a series of extended DPC algorithms based on k nearest neighbors, by a simpler and more efficient strategy.

D. ADAPTIVE DETERMINATION OF CLUSTER CENTERS

For the determination of cluster centers, it is difficult for the original DPC algorithm to distinguish center points and non-center points on the decision graph manually, if the boundary is not very clear. A reasonable adaptive way can deal with this problem. In this section, a simple and efficient method is proposed.

As an important judgment basis, the variable, called decision value γ, is also introduced, which is defined by the product of ρ_i and δ_i:

$$\gamma_i = \rho_i \times \delta_i \qquad (9)$$

By re-arrangement in descending order of γ value, the newly generated sequence of decision value is marked by γ^*. As analyzed above, the data points with both large ρ_i and large δ_i are potential to become cluster centers. Under this condition, the decision value is also large. Thus, it is crucial to determine adaptively the boundary between the center points and the non-center points in γ^*.

Due to the essential distinction between the center points and the non-center points in the degree of change of decision value, the location of the mutation-point is determined adaptively according to this key feature to lock the cluster centers. For γ^* sequence, there is the relatively large fluctuation, for the center points with relatively large decision value. In contrast, the fluctuation of γ^* value is not obvious for the non-center points with relatively small decision value. The second-order difference for γ^* value is used to describe this fluctuation.

When searching the cluster centers, it is necessary to narrow the search appropriately. That is to say, the points in the front position of γ^* with relatively large γ value that may become the cluster centers needs to be focused on. Because the points with small γ value does not have the characteristics of becoming the cluster centers, either ρ_i or δ_i is small, or both are small. Here, the search range is locked in $[1, \lfloor \sqrt{n} \rfloor]$ preliminarily, where n is the number of data. Previous studies have shown that this search range $[1, \lfloor \sqrt{n} \rfloor]$ is appropriate. In this way, not only the search efficiency can be improved, but also the removal of irrelevant data points will play a positive role in reducing disruption for determining the cluster centers.

Through the analysis of γ^* value of a large number of datasets with different distributions, it is found that there is a general rule that the γ value of the point with the most potential to be the cluster centers, γ_1^*, is much larger than γ_2^* and the value behind it generally. If the γ_1^* value is placed within the search range to determine the location of the mutation-point, the difference value of γ_1^* will be very large, which may lead to the wrong mutation-point being found, thereby affecting the determination of the cluster centers. Thus, the search range $[2, \lfloor \sqrt{n} \rfloor]$ is finally adopted in the present work. This treatment does not affect becoming a cluster center for the data point corresponding γ_1^*. Furthermore, it could avoid possible mistake to distinguish the center points and the non-center points.

Based on the second-order difference of γ^*, the mutation-point is described as:

$$M_p = \max \left\{ l | l = \arg \max_i \left[\left(\frac{i+1}{i} \right)^2 \frac{\xi_i}{\gamma_{\max}^* - \gamma_{\min}^*}, \right. \right.$$
$$\left. \left. i = 2, 3, \ldots, \lfloor \sqrt{n} \rfloor - 2 \right] \right\} \qquad (10)$$

where the function arg max is used to determine the set of variable points i which maximizes $\left(\frac{i+1}{i} \right)^2 \frac{\xi_i}{\gamma_{\max}^* - \gamma_{\min}^*}$, M_p is the

x_i		$NN_3(x_i); d(x_i, x_j), x_j \in NN_3(x_i)$	$\rho_i; B(x_i)$	$NN_3(x_i) \cap B(x_i)$
0		{5,2,3}; (1.38924, 1.33417, 1.11803)	0.260319; {3, 5, 6, 7, 9, 12}	$\neq \emptyset$
1		{6,3,7}; (1.76918, 0.943398, 1.13137)	0.260149; {0, 3, 5, 6, 7, 9, 12}	$\neq \emptyset$
2		{5,3,0}; (2.69258, 2.30868, 1.33417)	0.157843; {0, 1, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13}	$\neq \emptyset$
3		{5,1,0}; (1.21655, 0.943398, 1.11803)	0.305065; {7}	$= \emptyset$
4		{1,6,7}; (2.22036, 1.48661, 1.41421)	0.195267; {0, 1, 3, 5, 6, 7, 8, 9, 11, 12, 13}	$\neq \emptyset$
5		{3,0,8}; (1.38924, 0.894427, 1.21655)	0.285696; {3, 7, 9}	$\neq \emptyset$
6		{1,4,7}; (1.76918, 1.41421, 0.640312)	0.261526; {3, 5, 7, 9, 12}	$\neq \emptyset$
7		{4,1,6}; (1.48661, 1.13137, 0.640312)	0.306909; {}	$= \emptyset$
8		{3,0,5}; (2.00998, 0.894427, 1.7)	0.217183; {0, 1, 3, 5, 6, 7, 9, 11, 12, 13}	$\neq \emptyset$
9		{11,13,12}; (1.38924, 1.33417, 0.583095)	0.302434; {3, 7}	$= \emptyset$
10		{13,12,9}; (2.94279, 1.83848, 1.78885)	0.152204; {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13}	$\neq \emptyset$
11		{13,9,12}; (1.61555, 1.38924, 1.22066)	0.236661; {0, 1, 3, 5, 6, 7, 9, 12}	$\neq \emptyset$
12		{13,9,11}; (1.78885, 0.583095, 1.22066)	0.27835; {3, 5, 7, 9}	$\neq \emptyset$
13		{11,15,9}; (1.61555, 1.56205, 1.33417)	0.221943; {0, 1, 3, 5, 6, 7, 9, 11, 12}	$\neq \emptyset$
14		{12,13,11}; (3.02324, 2.65707, 1.80278)	0.133635; {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15}	$\neq \emptyset$
15		{11,9,13}; (3.13847, 2.74591, 1.56205)	0.134293; {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}	$\neq \emptyset$

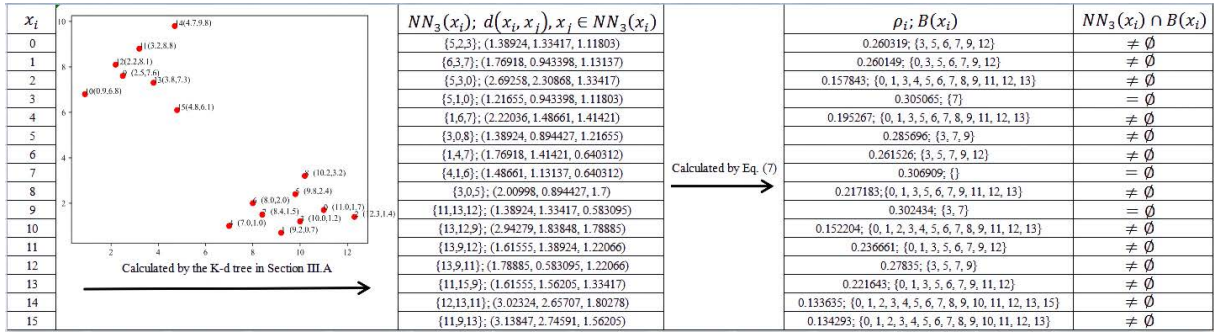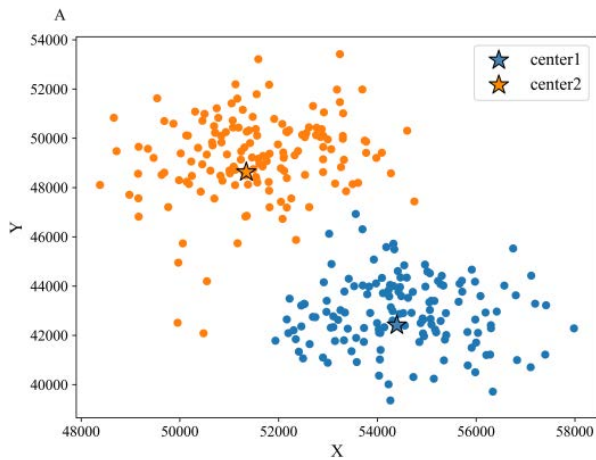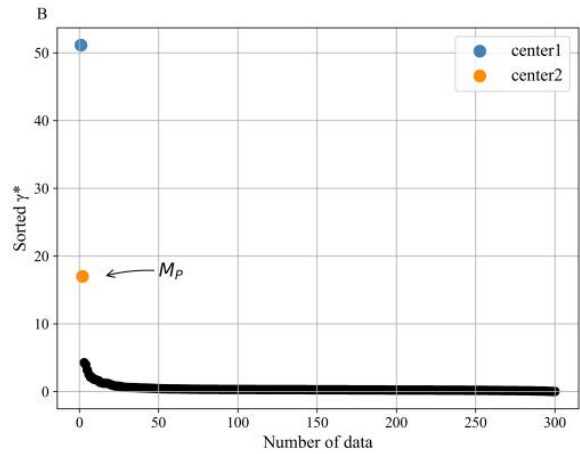Calculated by the K-d tree in Section III.A → Calculated by Eq. (7) →

FIGURE 1. Example demonstration of sparse search strategy for obtaining δ_i.

(a) Distribution of data points in dataset SS2.

(b) The data points are arranged in order γ^* from the largest to the smallest.

FIGURE 2. Simple two-dimensional synthetic dataset as the example.

mutation-point that takes the largest i value when arg max returns multiple i value, ξ_i is the second-order difference for γ_i^*, which is defined by Equation (11), $\gamma_{\min}^*$ and $\gamma_{\max}^*$ represent the minimum and maximum values of γ_i^* in $[2, \lfloor\sqrt{n}\rfloor]$,

$$\xi_i = \mu_i - \mu_{i+1}, \quad i = 2, 3, \ldots, \lfloor\sqrt{n}\rfloor - 2 \quad (11)$$
$$\mu_i = \gamma_i^* - \gamma_{i+1}^*, \quad i = 2, 3, \ldots, \lfloor\sqrt{n}\rfloor - 1 \quad (12)$$

where μ_i is the first-order difference between two adjacent terms γ_i^*. The adjustment coefficient $(\frac{i+1}{i})^2$ is to make $(\frac{i+1}{i})^2 \frac{\xi_i}{\gamma_{\max}^* - \gamma_{\min}^*}$ bigger when γ_i^* is larger, and smaller when γ_i^* is smaller. The distinction between the center points and the non-center points is more prominent through the adjustment coefficient. M_p is the mutation-point with the largest ordering value when the second-order difference of γ^* is mutated. Then all data points with γ_i^* larger than $\gamma_{M_p}^*$, including $\gamma_{M_p}^*$, are the candidate cluster centers of searching, namely the corresponding points from γ_1^* to $\gamma_{M_p}^*$. Finally, the pseudo-center points with large ρ_i small δ_i or small ρ_i large δ_i were removed from these candidate centers as the final cluster centers. Although the γ_i values of these pseudo-center points are large, they do not possess the characteristics of the center points, that is, ρ_i and δ_i are both large. Specifically,

the points $x_i (i \in [1, M_p])$ satisfying both $\rho_i > \frac{\sum_{t=1}^{\lfloor\sqrt{n}\rfloor} \rho_t}{\lfloor\sqrt{n}\rfloor}$ and $\delta_i > \frac{\sum_{t=1}^{\lfloor\sqrt{n}\rfloor} \delta_t}{\lfloor\sqrt{n}\rfloor}$ in the candidate center points are retained, and the remaining pseudo-center points are removed to get the final cluster centers.

Next, a two-dimensional synthetic dataset SS2 is adopted as an example to clearly demonstrate the process of determining M_p. This dataset has a total of 300 data points ($n = 300$). In other words, the subsequent calculation of second-order difference only needs to focus on the 16 largest γ^* in the range of [2, 17], where $\lfloor\sqrt{n}\rfloor = 17$. Based on this information, 15 μ_i from the 16 largest γ^* are firstly calculated according to Equation (12); Then, 14 ξ_i are calculated from 15 μ_i by Equation (11); Finally, the mutation-point M_p is determined by Equation (10), which is $M_p = 2$. Here, the two candidate center points meet the above conditions. Therefore, the final cluster centers are the two data points corresponding to γ_1^* and γ_2^*.

The distribution of data points in dataset SS2 is shown in Fig. 2(a), and the yellow points indicated by arrows in Fig. 2(b) are mutation-point determined by the proposed second-order difference method. In other words, the

mutation-point and the blue point (center 1 and center 2) were identified as the cluster centers of dataset SS2, and the black dots below were the ordinary non-center points.

E. PROCESS OF SKTDPC

The overall process of SKTDPC algorithm includes three main parts: calculation of dual acceleration for ρ_i and δ_i based on K-d tree and sparse search strategy, adaptive determination of cluster centers by second-order difference method for γ, and allocation of non-cluster center points. The implementation of the first part has been described in Sections III.B and III.C. The second part has been described in Section III.D thoroughly. The third part for allocating non-center points follows the original DPC depicted in Section II.A. Therefore, the specific process of SKTDPC algorithm is summarized as following Algorithm 2.

Algorithm 2: SKTDPC Algorithm

Input: The dataset $S = \{x_1, x_2, \ldots, x_n\} \in R^{n \times K}$; The number of the nearest neighbors k

Output: The clustering result Y

1: Construct a K-d tree for dataset S, by Algorithm 1.
2: Search the k nearest neighbors of the point x_i based on the constructed K-d tree, by Algorithm 1. Meanwhile, the symmetric sparse distance matrix $\widetilde{D}$ is obtained.
3: According to Equation (7) by the k nearest neighbors, the first acceleration calculation is carried out to obtain ρ_i, and the descending order processing is made for ρ_i to generate $\rho_i^* (i = 1, 2, \ldots, n)$.
4: According to Equations.(3) and (8) by the sparse search method, the second acceleration calculation is carried out to obtain δ_i.
5: According to Equation (9), calculate γ_i and further obtain γ_i^* by descending order processing.
6: According to Equation (10), calculate the mutation point M_p to determine the candidate cluster centers as the data points with decision value $\gamma_i^* (i = 1, 2, \ldots, M_p)$. The candidate centers $x_i (i \in [1, M_p])$, which satisfies both $\rho_i > \frac{\sum_{t=1}^{\lfloor \sqrt{n} \rfloor} \rho_t}{\lfloor \sqrt{n} \rfloor}$ and $\delta_i > \frac{\sum_{t=1}^{\lfloor \sqrt{n} \rfloor} \delta_t}{\lfloor \sqrt{n} \rfloor}$ conditions, is the final cluster centers.
7: Assign the non-center points, according to the allocation principle, that each non-center point and its nearest point with a higher local density have the same cluster.
8: Return the clustering result Y.

Algorithm 2 summarizes the entire process of SKTDPC. On the one hand, SKTDPC replaces the calculation of distance for symmetric full-rank distance matrix D with the symmetric sparse distance matrix $\widetilde{D}$. The K-d tree method and the sparse search strategy by the intersection between sets $NN_k(x_i)$ and $B(x_i)$ are developed to accelerate the calculation of both ρ_i and δ_i. On the other hand, the second-order difference method is used to find the boundary between the

center points and the non-center points. Therefore, adaptively determining the cluster centers is achieved quickly and successfully. Finally, the present algorithm can effectively reduce the computational complexity while maintaining or even improving the clustering accuracy.

F. ANALYSIS OF COMPLEXITY

It should be stressed that the analysis of computational complexity of DPC in some literature only focus on the variable, data number n, then the complexity is $O(n^2)$ [18]–[30]. Different from these literature, the present work considers the influence of the two variables, n and K, simultaneously. Thus, the computational complexity of the original DPC can be regarded as $O(n^2 K)$.

The computational complexity of SKTDPC algorithm is mainly determined by the four parts: (1) Calculation process of local density ρ_i based on K-d tree. The computational complexity of this part mainly depends on the process of searching k nearest neighbors of data points. The process includes construction of the tree and the search of k nearest neighbors. The complexity of these two parts is $O(nK \log n)$ and $O(n(n^{1-1/K} + k))$ respectively. Thus, the complexity of the calculation process of local density ρ_i based on K-d tree is $O(nK \log n) + O(n(n^{1-1/K} + k))$. (2) Acquisition process of relative-separation δ_i based on the sparse search strategy with intersection between $NN_k(x_i)$ and $B(x_i)$. The complexity of this part is far less than $O(n(n^{1-1/K} + k))$, as analyzed in Section III.C in detail. (3) Adaptive determination of cluster centers. The complexity of this part is determined by descending order, second-order difference, and determining the center points that satisfy the mean value condition. The complexity of the three steps is $O(n \log n)$, $O(\lfloor \sqrt{n} \rfloor - 2)$ and $O(M_p)$, respectively. Thus, the computational complexity of the whole part is $O(n \log n) + O(\lfloor \sqrt{n} \rfloor - 2) + O(M_p)$. (4) Allocation of non-center points. The computational complexity of this part is $O(n)$. The total computational complexity of SKTDPC algorithm is $O(nK \log n) + O(n(n^{1-1/K} + k)) + O(n \log n) + O(\lfloor \sqrt{n} \rfloor - 2) + O(M_p) + O(n)$. Normally, the dimension K is much smaller than the data number n. Therefore, the overall computational complexity of the SKTDPC algorithm is the largest one among the four parts. That is, the complexity is $O(n(n^{1-1/K} + k))$. In order to make it easier to analyze and compare with the complexity $O(n^2 K)$ of DPC, $O(n(n^{1-1/K} + k))$ could be transformed into $O(n^2(1/\sqrt[K]{n} + k/n))$, where $1/\sqrt[K]{n}$, k/n take values in range $(0, 1)$. In other words, the computational complexity of SKTDPC algorithm is much lower than the complexity $O(n^2 K)$ of DPC.

IV. EXPERIMENTS AND RESULTS

In this section, the present SKTDPC algorithm is compared with the six state-of-the-art and typical clustering algorithms, including FSDPC [26], the original DPC [14], DGDPC [38], DPC-KNN [18], DBSCAN [16] and the K-means algorithm [15]. For the seven algorithms, the first five algorithms belong to the DPC series. FSDPC, the original DPC and DGDPC are the algorithms taking cut-off distance d_c as a

TABLE 1. Characteristics of synthetic datasets.

Name	Data	Clusters	Dimension
Flame	240	2	2
Spiral	312	3	2
Aggregation	788	7	2
R15	600	15	2
S1	5000	15	2
S3	5000	15	2
A1	3000	20	2
A3	7500	50	2

TABLE 2. Characteristics of UCI real datasets.

Name	Data	Clusters	Dimension
Seeds	210	3	7
Iris	150	3	4
Banknote authentication	1372	2	4
Wine	178	3	13
Ecoli	336	8	7
Parking Birmingham	35501	3	5
Pendigits	10992	10	16

hyper-parameter, and DGDPC has an extra merging threshold l hyper-parameter. However, SKTDPC and DPC-KNN adopt the hyper-parameter k, the number of nearest neighbors. DBSCAN is another kind of density-based clustering algorithm, for which two hyper-parameters, are used. In contrast, the K-means algorithm is a fast clustering algorithm, in which the allocation of non-centers points is determined by the nearest distance from cluster centers.

A. INTRODUCTION TO DATASETS AND METRICS

To verify the clustering effect and efficiency of the SKT-DPC algorithm, 15 commonly used clustering datasets [38]–[45] are adopted in the present work. These include the eight synthetic datasets (http://cs.joensuu.fi/sipu/data-sets/) shown in Table 1 and the seven UCI real datasets [46] (http://archive.ics.uci.edu/ml) which are shown in Table 2. In the two tables, the basic information such as number of data, number of class clusters, and data dimension is listed. Here, in order to visualize the clustering effect clearly and intuitively, in Table 1 all the eight synthetic datasets are two-dimensional. In contrast, the UCI real datasets are high-dimensional. They are discussed in Sections IV.B, IV.C and IV.D, respectively.

The algorithmic clustering performance could be effectively evaluated by using running time and five well-known indexes, including accuracy (Acc) [47], adjusted mutual information (AMI) [48], adjusted rand index (ARI) [48], normalized mutual information (NMI) [49] and Fowlers-Mallows index (FMI) [50]. The five indexes are powerful tools to measure the clustering results, and the maximum values are 1 for all the indexes. The closer to 1 they are, the better the clustering effect is. In addition, running time is an important performance indicator used to measure the clustering efficiency. The smaller the value is, the higher the clustering efficiency is, vice versa.

Among the other six clustering algorithms compared with SKTDPC, the code for original DPC and DPC-KNN are obtained by retranslating the MATLAB source code, provided by the authors of original DPC and DPC-KNN, into Python language form. The FSDPC and DGDPC codes are written in accordance with the original reference. The algorithms for DBSCAN and K-means are programmed by the sklearn.cluster library in Python. In order to make a fair comparison, for each algorithm experiments are conducted with its optimal hyper-parameters. At the same time, the values of all the indicators below are average by the 20 times independently repeated experiments for each algorithm, to avoid the occurrence of contingency. In addition, all algorithms are implemented by Python 3.8.0. The experiments are carried out in a computer environment with a core i7 2.3 GHz processor, Windows 10 operating system and 16GB RAM. The relevant code for this article is published in https://github.com/Nutshe/code.

B. EXPERIMENTS ON SYNTHETIC DATASETS

Among the eight synthetic datasets with different distributions, the four classic datasets Flame, Spiral, Aggregation and S3 were used in the reference on original DPC [14]. The 15 class clusters of dataset R15 are distributed in the ring and have similar Gaussian distribution. The datasets S1, A1 and A3 are the three commonly used clustering datasets with different characteristics of overlapping, complexity and number of class clusters. Applying to the eight two-dimensional synthetic datasets, the clustering results for the seven algorithms are shown in Figs. 3-10, visually. Meanwhile, the five clustering evaluation indexes are shown in Tables 3-5, in which DPC refers to the original DPC [14]. In addition, the values of optimal hyper-parameter, abbreviated by Par, suitable for each algorithm are also given in Table 3.

For the Acc, AMI, ARI, NMI and FMI index values of the seven algorithms shown in Tables 3-5 and the clustering effect shown in Figs. 3-10, we can see that the SKTDPC algorithm shows the best performance on all of the eight synthetic datasets. This is because SKTDPC introduces the idea of k nearest neighbors to define the local density and obtains the correct cluster centers adaptively by second-order difference method. Through these two methods, SKTDPC can capture more comprehensive information for data distribution and avoid the wrong selection of cluster centers, which also plays an important role in improving the cluster effect. By further comparing SKTDPC with other algorithms, it is demonstrated that there are relative weaknesses with different degree for these six algorithms. Firstly, FSDPC, the original DPC and DGDPC are slightly inferior to SKTDPC, even though they perform well on most datasets. This is because these two algorithms only consider the global structure of data, resulting in partial information loss. By contrast, the local density of SKTDPC is calculated based on the distances from k nearest neighbors, which can deal with local data information well. In addition, FSDPC and the original DPC need to determine the clustering center manually. Thus

TABLE 3. ACC of algorithms on synthetic datasets.

Data	SKTDPC		FSDPC		DPC		DGDPC		DPC-KNN		DBSCAN		K-means	
	Acc	Par	Acc	Par	Acc	Par	Acc	Par	Acc	Par	Acc	Par	Acc	Par
Flame	**1**	3	**1**	6	**1**	6	**1**	0.5/5	**1**	5	0.975	0.09/8	0.838	2
Spiral	**1**	4	**1**	2	**1**	2	**1**	0/5	**1**	7	**1**	0.04/2	0.343	3
Aggregation	**0.997**	6	0.992	0.19	0.992	0.19	0.996	0.5/5	0.996	5	0.953	0.05/8	0.860	7
R15	**0.997**	5	0.996	1	**0.997**	1	**0.997**	0.5/5	0.996	5	0.873	0.02/6	0.996	15
S1	**0.997**	7	0.992	1	0.992	1	0.992	1/1	0.992	6	0.975	0.03/13	0.992	15
S3	**0.901**	3	0.858	2	0.857	2	0.868	1/1	0.522	6	0.700	0.04/73	0.873	15
A1	**0.998**	6	0.971	2	0.973	2	0.972	0.5/1	0.663	5	0.825	0.03/25	0.948	20
A3	**0.996**	7	0.990	0.2	0.988	0.2	0.991	1/1	0.625	5	0.921	0.03/62	0.989	50

TABLE 4. AMI, ARI of algorithms on synthetic datasets.

Data	SKTDPC		FSDPC		DPC		DGDPC		DPC-KNN		DBSCAN		K-means	
	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI
Flame	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.839	0.923	0.386	0.453
Spiral	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0
Aggregation	**0.992**	**0.996**	0.979	0.990	0.978	0.991	**0.992**	**0.996**	0.989	0.993	0.949	0.911	0.834	0.730
R15	**0.994**	**0.993**	0.993	**0.993**	0.993	**0.993**	0.993	**0.993**	0.993	**0.993**	0.845	0.717	**0.994**	**0.993**
S1	**0.994**	**0.994**	0.991	0.991	0.991	0.991	0.991	0.991	0.991	0.991	0.961	0.962	0.991	0.992
S3	**0.858**	**0.803**	0.796	0.729	0.796	0.728	0.814	0.757	0.576	0.566	0.667	0.392	0.833	0.775
A1	**0.997**	**0.996**	0.958	0.941	0.958	0.942	0.958	0.941	0.782	0.562	0.871	0.733	0.959	0.944
A3	**0.996**	**0.992**	0.991	0.978	0.990	0.978	0.991	0.979	0.763	0.538	0.932	0.858	0.987	0.980

TABLE 5. NMI, FMI of algorithms on synthetic datasets.

Data	SKTDPC		FSDPC		DPC		DGDPC		DPC-KNN		DBSCAN		K-means	
	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI
Flame	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.841	0.964	0.399	0.736
Spiral	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0.328
Aggregation	**0.992**	**0.997**	0.982	0.987	0.984	0.986	**0.992**	0.996	0.987	0.990	0.950	0.933	0.837	0.788
R15	**0.994**	**0.993**	**0.994**	**0.993**	**0.994**	**0.993**	**0.994**	**0.993**	**0.994**	**0.993**	0.856	0.736	**0.994**	**0.993**
S1	**0.994**	**0.994**	0.991	0.992	0.991	0.992	0.991	0.992	0.991	0.992	0.962	0.964	0.992	0.992
S3	**0.859**	**0.817**	0.780	0.745	0.799	0.746	0.803	0.766	0.575	0.572	0.670	0.449	0.835	0.790
A1	**0.997**	**0.997**	0.959	0.947	0.959	0.945	0.959	0.947	0.781	0.560	0.873	0.752	0.960	0.947
A3	**0.996**	**0.992**	0.989	0.979	0.990	0.979	0.987	0.978	0.762	0.538	0.935	0.861	0.988	0.980

TABLE 6. Clustering efficiency of algorithms on synthetic datasets.

Data	Running time (s)						
	SKTDPC	FSDPC	DPC	DGDPC	DPC-KNN	DBSCAN	K-means
Flame	0.1	0.24	0.31	0.34	0.28	0.17	0.15
Spiral	0.14	0.29	0.33	0.39	0.37	0.14	0.15
Aggregation	0.53	0.73	0.82	0.88	0.74	0.19	0.32
R15	0.35	0.54	0.61	0.86	0.79	0.70	0.27
S1	8.16	14.38	18.70	22.13	20.29	1.10	0.46
S3	10.58	16.78	19.61	23.27	21.07	8.63	0.42
A1	4.04	6.91	7.81	8.02	7.75	1.41	0.43
A3	12.86	30.41	40.06	49.63	45.93	2.72	1.62

TABLE 7. ACC of algorithms on different UCI datasets.

Data	SKTDPC		FSDPC		DPC		DGDPC		DPC-KNN		DBSCAN		K-means	
	Acc	Par	Acc	Par	Acc	Par	Acc	Par	Acc	Par	Acc	Par	Acc	Par
Seeds	**0.914**	3	0.895	1	0.904	2	0.909	1/1	0.895	6	0.348	0.09/3	0.890	3
Iris	**0.960**	2	0.957	4	**0.960**	4	0.892	1/3	**0.960**	7	0.740	0.12/5	0.887	3
Banknote authentication	0.683	2	0.680	2	0.687	2	0.685	1/2	0.517	5	**0.742**	0.09/2	0.576	2
Wine	0.893	6	0.853	2	0.854	2	0.762	1/1	0.859	6	0.815	0.5/21	**0.949**	3
Ecoli	**0.807**	2	0.542	2	0.542	2	0.586	1/2	0.631	3	0.622	0.2/10	0.595	8
Parking Birmingham	**0.575**	4	0.541	1	0.543	1	0.557	1/1	0.556	4	0.432	0.08/67	0.336	3

the possible wrong selection of cluster centers would also lead to poor clustering effect. Secondly, for DPC-KNN and DBSCAN, both of them show obviously poor clustering effect on S3, A1 and A3 datasets, as shown in Figs. 8-10.
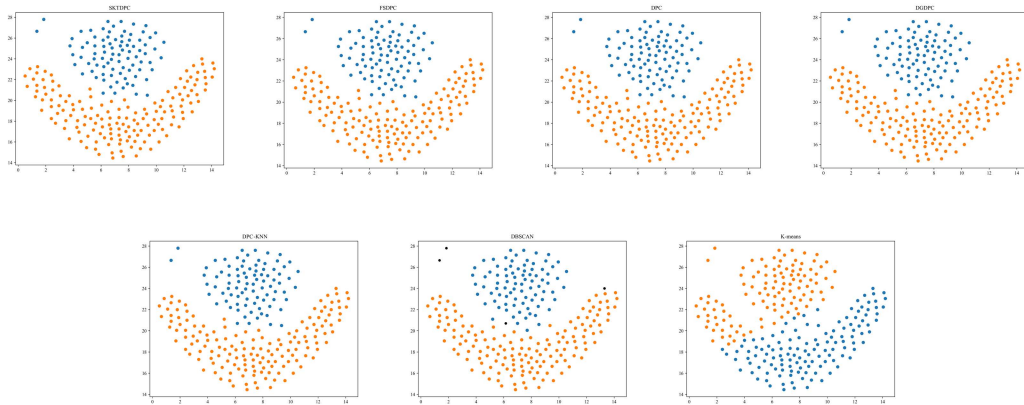
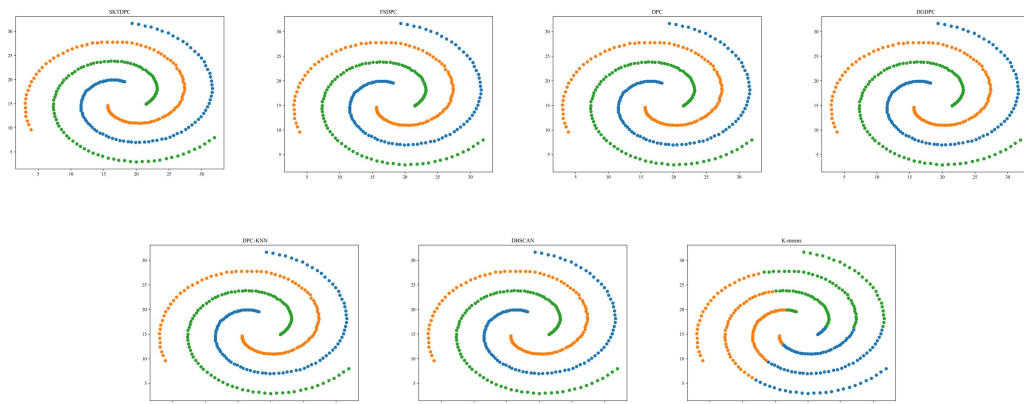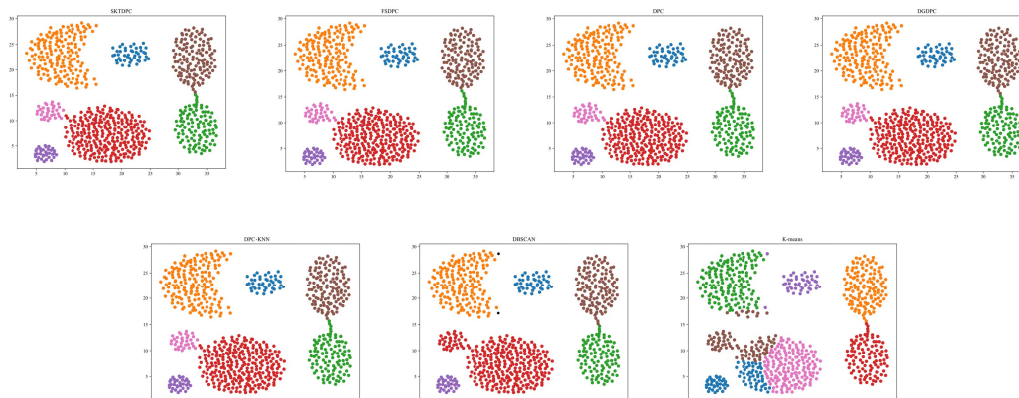**FIGURE 3.** Clustering results of the seven algorithms on Flame dataset.

FIGURE 4. Clustering results of the seven algorithms on Spiral dataset.

TABLE 8. AMI, ARI of algorithms on different UCI datasets.

Data	SKTDPC		FSDPC		DPC		DGDPC		DPC-KNN		DBSCAN		K-means	
	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI
Seeds	**0.734**	**0.844**	0.675	0.785	0.704	0.826	0.728	0.831	0.675	0.785	0.090	0.542	0.674	0.803
Iris	**0.862**	**0.923**	0.858	0.911	**0.862**	**0.923**	0.789	0.839	0.861	**0.923**	0.640	0.729	0.742	0.811
Banknote authentication	0.222	0.673	0.225	0.655	0.230	0.678	0.228	0.676	0.108	0.589	**0.537**	**0.770**	0.017	0.514
Wine	0.726	0.801	0.680	0.745	0.682	0.746	0.619	0.683	0.685	0.749	0.590	0.712	**0.842**	**0.902**
Ecoli	**0.675**	**0.815**	0.574	0.559	0.573	0.559	0.601	0.605	0.620	0.617	0.496	0.676	0.618	0.559
Parking Birmingham	**0.078**	**0.548**	0.052	0.443	0.052	0.443	0.063	0.526	0.061	0.477	0.017	0.382	0.010	0.376

TABLE 9. NMI, FMI of algorithms on different UCI datasets.

Data	SKTDPC		FSDPC		DPC		DGDPC		DPC-KNN		DBSCAN		K-means	
	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI	NMI	FMI
Seeds	**0.734**	**0.844**	0.675	0.785	0.704	0.826	0.728	0.831	0.675	0.785	0.090	0.542	0.674	0.803
Iris	**0.862**	**0.923**	0.858	0.911	**0.862**	**0.923**	0.789	0.839	0.861	**0.923**	0.640	0.729	0.742	0.811
Banknote authentication	0.222	0.673	0.225	0.655	0.230	0.678	0.228	0.676	0.108	0.589	**0.537**	**0.770**	0.017	0.514
Wine	0.726	0.801	0.680	0.745	0.682	0.746	0.619	0.683	0.685	0.749	0.590	0.712	**0.842**	**0.902**
Ecoli	**0.675**	**0.815**	0.574	0.559	0.573	0.559	0.601	0.605	0.620	0.617	0.496	0.676	0.618	0.559
Parking Birmingham	**0.078**	**0.548**	0.052	0.443	0.052	0.443	0.063	0.526	0.061	0.477	0.017	0.382	0.010	0.376

This implies that the processing ability of the two algorithms is obviously weak for datasets with a high degree of overlap relatively. Moreover, the accuracy of DBSCAN is slightly lower than that of other algorithms on Flame, Aggregation, R15 and S1 datasets. This is caused by its recognition of noise points, as shown in related figures. Finally, the clustering
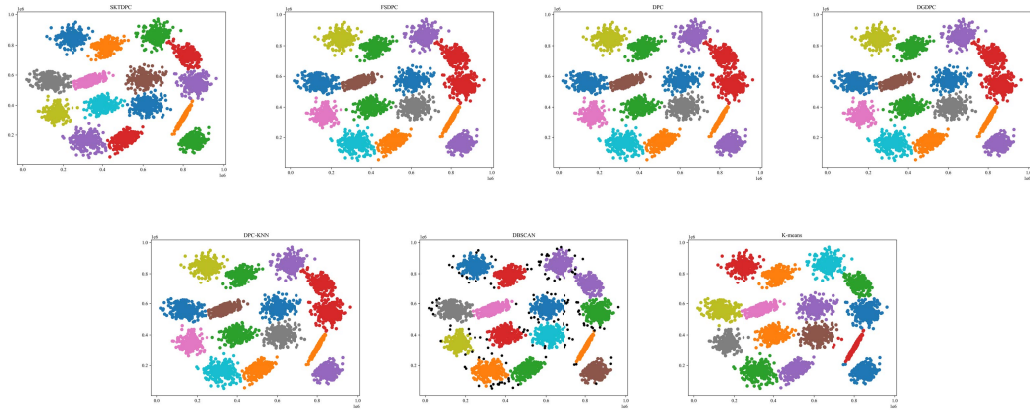
FIGURE 5. Clustering results of the seven algorithms on Aggregation dataset.
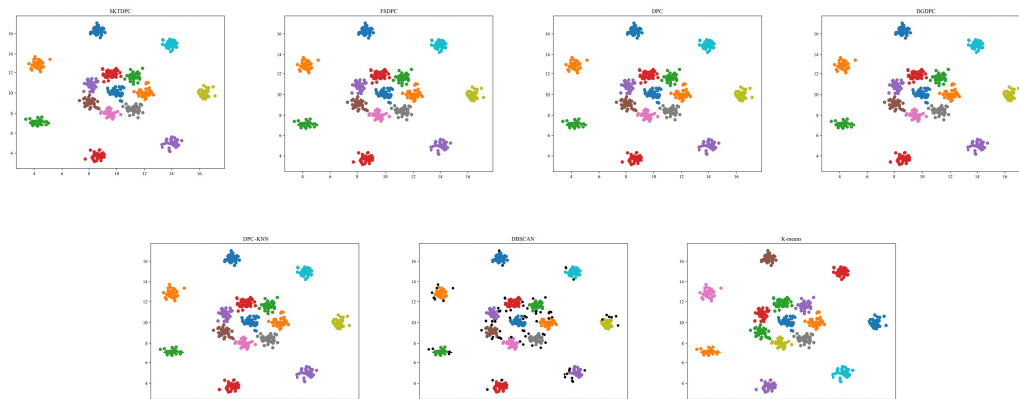
FIGURE 6. Clustering results of the seven algorithms on R15 dataset.

effect of K-means algorithm is very poor on Flame, Spiral and Aggregation datasets, as shown in Figs. 3-5. This is attributed to the defects of K-means itself, which cannot capture the structural characteristics of non-convex dataset. In contrast, the other six density-based algorithms show good performance on these datasets.

The running time of SKTDPC algorithm and the other six algorithms is shown in Table 6 to evaluate the algorithmic clustering efficiency. It can be found that among the five algorithms belonging to the DPC series, including SKTDPC, FSDPC, the original DPC, DGDPC and DPC-KNN, the clustering efficiency of SKTDPC algorithm newly proposed is much higher than that for the other four algorithms. Especially when the dataset size is large enough, the acceleration effect is relatively more obvious. This is precisely because SKTDPC accelerates the calculation of local density and relative-separation by K-d tree and the sparse search strategy, and further requires less distance calculation and storage space compared with the other four algorithms. It can also be found that the running speed of K-means and DBSCAN is relatively faster compared with the DPC series. As discussed above, however, K-means is only applicable

to globular clusters. For non-convex clusters, the clustering effect is very poor, even though the efficiency is very high. As for DBSCAN, it is sensitive to the setting of two hyper-parameters and is difficult to use when the data density is not evenly distributed. In addition, as shown above, the processing ability of this algorithm is weak obviously for datasets with a high degree of overlap, such as S3, A1 and A3.

C. EXPERIMENTS ON REAL-WORLD DATASETS

In order to further verify the clustering effect and efficiency of SKTDPC algorithm, a comparative analysis is made on six UCI real datasets (Seeds, Iris, Banknote, Wine, Ecoli, Parking Birmingham) with different dataset size, number of class clusters and data dimension. In contrast with the eight synthetic datasets, which are two-dimensional, the UCI real datasets are relatively high-dimensional and complex. The performance of SKTDPC and other six algorithms is compared. The index values and parameter values of all algorithms are shown in Tables 7-10, in which "-" indicates that the value for running time at this position is very large or even cannot be obtained.
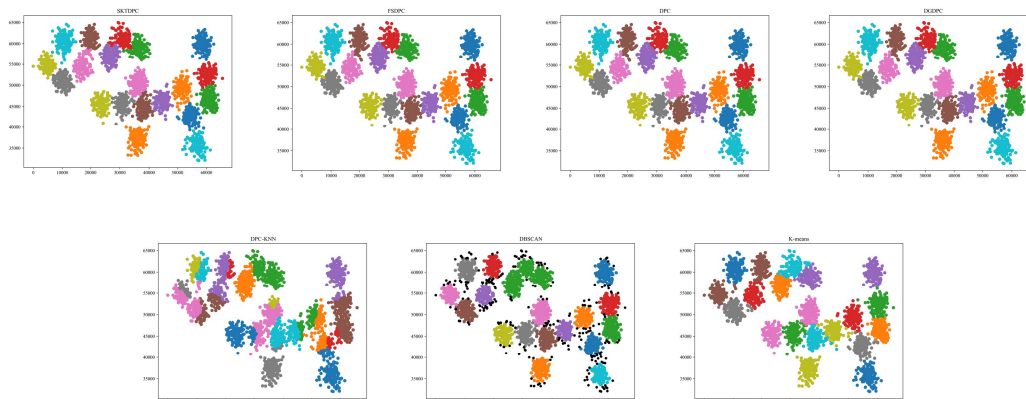
FIGURE 7. Clustering results of the seven algorithms on S1 dataset.
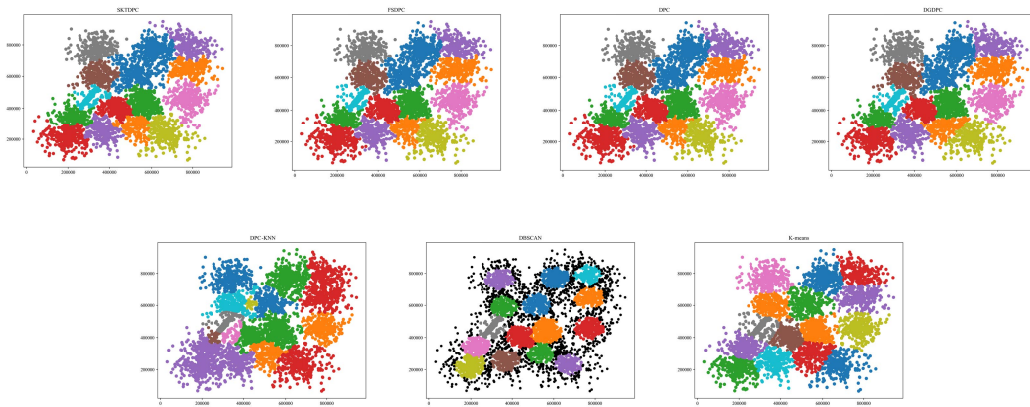
FIGURE 8. Clustering results of the seven algorithms on S3 dataset.

TABLE 10. Clustering efficiency of algorithms on different UCI datasets.

Data	Running time (s)						
	SKTDPC	FSDPC	DPC	DGDPC	DPC-KNN	DBSCAN	K-means
Seeds	0.16	0.17	0.19	0.27	0.39	0.26	0.11
Iris	0.09	0.12	0.15	0.20	0.27	0.09	0.08
Banknote authentication	1.63	2.97	3.85	6.12	11.58	0.10	0.19
Wine	0.17	0.19	0.22	0.31	0.40	0.19	0.08
Ecoli	0.31	0.43	0.48	0.61	0.76	0.14	0.14
Parking Birmingham	685.91	1316.23	–	–	–	9.22	0.43

From the Acc, AMI, ARI, NMI and FMI values shown in Tables 7-9 it can be found that: (1) SKTDPC algorithm achieves the best clustering results compared with the DPC series and K-means algorithm in all real datasets, and the clustering accuracy is improved to a certain degree. For DBSCAN algorithm, it gives the best clustering accuracy on Banknote authentication, better than SKTDPC slightly. However, DBSCAN shows much lower accuracy than that given by SKTDPC on other types of datasets. (2) The overall clustering effect of DPC series algorithm is better than K-means and DBSCAN algorithm, indicating that the general

applicability of DPC series algorithm is stronger relatively. (3) FSDPC and DGDPC algorithms can maintain similar clustering accuracy as the original DPC, but there is still distinction in some extent for these two algorithms compared with SKTDPC algorithm. This is because FSDPC, DPC and DGDPC ignore local data information. In addition, SKTDPC can obtain the location of mutation-point adaptively in the clustering process to determine the cluster centers without interrupting the algorithm. Furthermore, due to the complexity and sparsity of the Banknote and Parking datasets, the clustering results of the seven algorithms is not satisfactory.

FIGURE 9. Clustering results of the seven algorithms on A1 dataset.
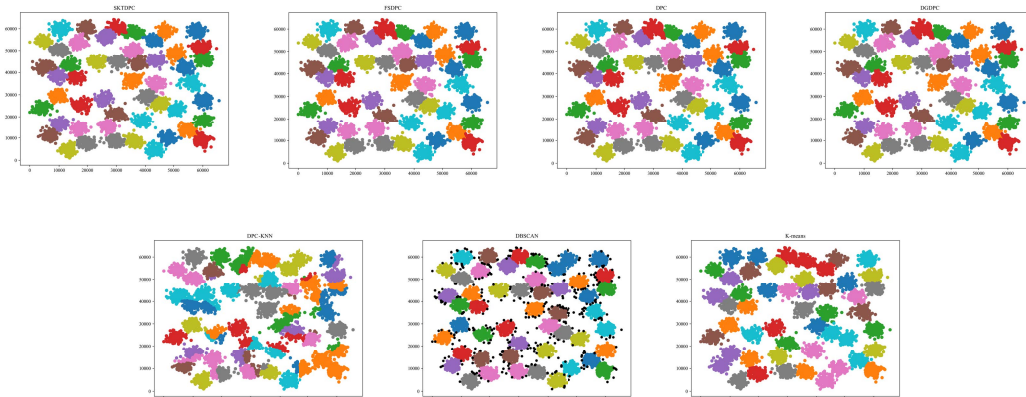
FIGURE 10. Clustering results of the seven algorithms on A3 dataset.

For this kind of dataset, it is necessary to further study its inherent data structure characteristics, to find a more effective clustering method.

By analyzing Table 10, it can be found that the proposed SKTDPC algorithm obviously improves the clustering efficiency. In small datasets, the clustering efficiencies of the seven algorithms are roughly the same, but the distinction becomes more obvious with the increase of data volume. For the dataset Parking Birmingham with the largest data volume, the clustering efficiency of SKTDPC is 1.92 times and more times higher than that of FSDPC, DPC, DGDPC and DPC-KNN, respectively. At this time, DPC, DGDPC and DPC-KNN are on the verge of collapse and almost lost their execution ability because they require huge amount of computation and storage space. In addition, the running time of these two algorithms is seriously time-consuming, which is already meaningless. The high efficiency for SKTDPC is mainly attributed to the dual acceleration strategies to deal with large datasets, as analyzed in Section III.F. One is the acceleration for calculation of the local density by K-d tree. Thus a sparse distance matrix $\widetilde{D}$ is calculated instead of a full-rank matrix D to find the k

nearest neighbors. Another is the acceleration for calculation of the relative-separation by the sparse search strategy with intersection between $NN_k(x_i)$ and $B(x_i)$. Therefore, the dual accelerations strategies play an important role in reducing the algorithmic complexity, which makes SKTDPC algorithm show more prominent on large datasets. It should also be noted that although the running speed of DBSCAN and K-means algorithms is fast, due to their own limitations, the universality of the algorithms are too weak to produce an ideal clustering results.

D. ROBUSTNESS TO NUMBER OF NEAREST NEIGHBORS K
In this section, this paper will analyze and discuss the influence of the proposed algorithm SKTDPC on the Acc under different parameter k settings. The number of nearest neighbors k is the only hyper-parameter of SKTDPC, so it is crucial to analyze the change of clustering effect under different settings. The adjustment of parameters is to accommodate datasets with different distribution characteristics. At the same time, within the scope of the regulation of as small as possible to avoid algorithm into local

(a) Acc of SKTDPC on synthetic datasets.

(b) Acc of SKTDPC on UCI datasets.
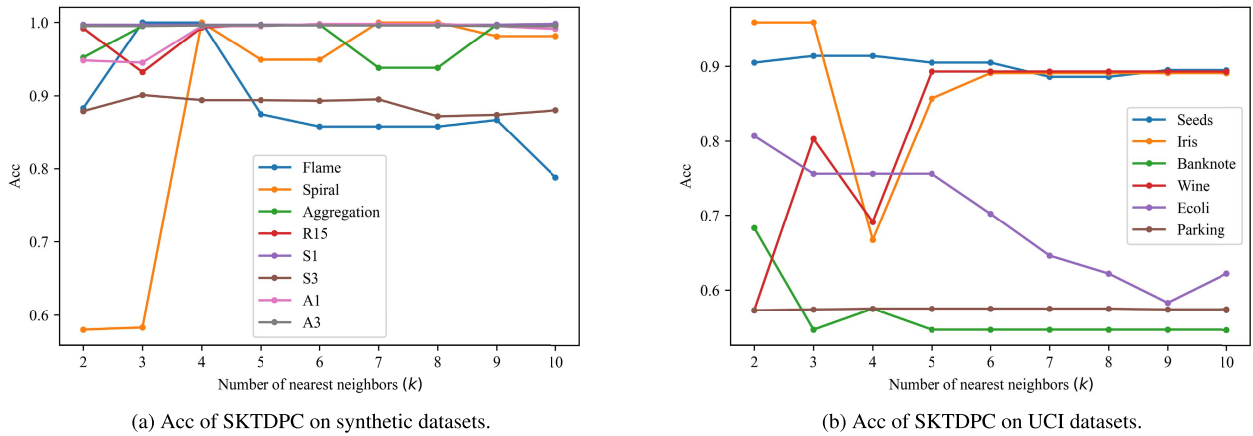
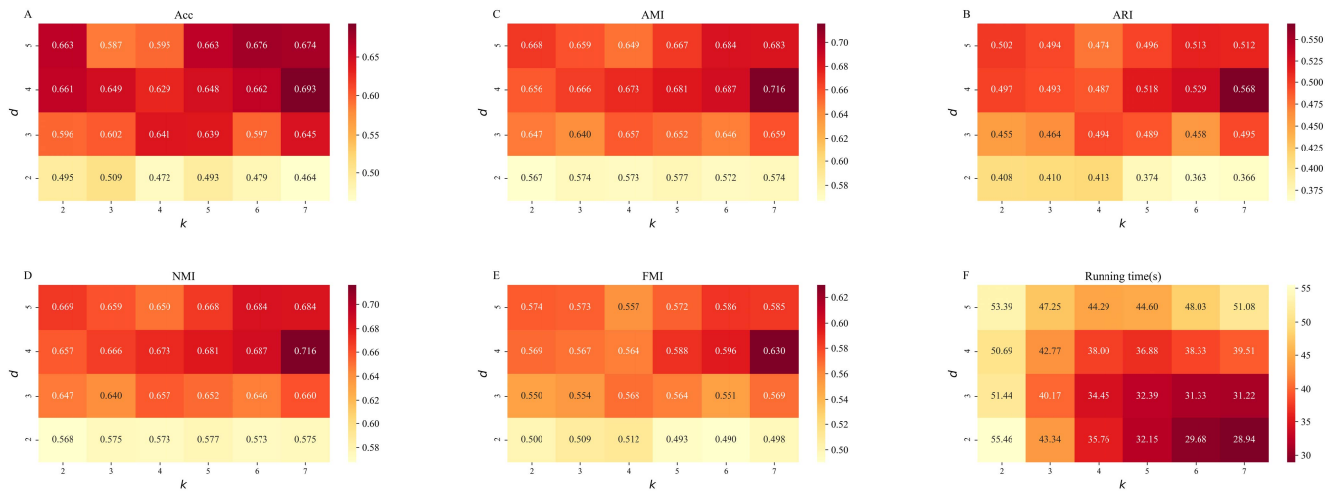**FIGURE 11.** **Acc of SKTDPC under different parameters *k*.**

FIGURE 12. **Results of six indexes obtained by SKTDPC algorithm for Pendigits dataset under different parameter combinations.**

TABLE 11. **Experimental results of PCA+SKTDPC and SKTDPC under optimal parameters.**

DATA (Pendigits)	Acc	AMI	ARI	NMI	FMI	Running time (s)	Par
PCA+SKTDPC	0.693	0.716	0.568	0.716	0.630	39.51	7/4
SKTDPC	0.707	0.744	0.610	0.744	0.661	182.4	7

optimum and can achieve ideal clustering result is what we are after. If the parameters of the algorithm need to be optimized in a wide range, it will not only affect the search efficiency of the optimal parameters, but also make the robustness and stability of the algorithm worse and it is difficult to achieve the ideal clustering effect. Therefore, we focus on analyzing the change of Acc within a reasonable range of $k = [2, 10]$ for the SKTDPC algorithm, as shown in Fig. 11.

We can clearly see that most of the datasets of SKT-DPC show stable and ideal clustering effect relatively in

a small range of parameters, which shows the excellent robustness of the algorithm. On Spiral, Iris, Wine and Ecoli datasets, although Acc shows relatively large fluctuations, the algorithm can also quickly achieve ideal clustering effect in a small range of parameter adjustment, which is not easy. According to our analysis, the reason for this large fluctuation may be that the algorithm automatically identifies incorrect cluster centers under the parameter k with low Acc value. Therefore, the proposed SKTDPC algorithm can achieve the ideal clustering effect and robustness suitable for most datasets within a small range of parameter adjustment.

E. ADDITIONAL DISCUSSION

The focus of the present work is on dealing with datasets with arbitrary shape and large size. As for higher dimensional datasets, the improvement for clustering efficiency of SKTDPC is not obvious. However, it is not a vexing

problem. In this section, the applicability of SKTDPC on higher dimensional dataset is briefly discussed and some reasonable suggestions are given. Many effective dimension reduction methods can be used to handle this case, such as principal component analysis (PCA), locally linear embedding (LLE), laplacian eigenmaps (LE), etc.

The following is a simple example of processing 16-dimensional dataset Pendigits using PCA as a dimension reduction method. Fig. 12 shows the heat map of Acc, AMI, ARI, NMI, FMI and running time obtained by PCA+SKTDPC algorithm by different combinations of k and d, where d is the dimension after dimension reduction. The darker the color of the rectangular block is, the better the indicator value is. We can see that Acc, AMI, ARI, NMI and FMI are all optimal with the set of parameters $k = 7$ and $d = 4$. However, the corresponding running time of 39.51s is not the minimum in the range considered. Meanwhile, there are non-unique parameter combinations with values of running time less than 39.51s. In this case, for determining parameter combination the principle should be satisfied, that is the highest efficiency is achieved on the premise of guaranteeing the ideal clustering effect, instead of pursuing speed blindly. In accord with this principle, the final parameter combination, $k = 7$ and $d = 4$, is adopted. Under this parameters, the running time results of PCA+SKTDPC and SKTDPC are shown in Table 11.

V. CONCLUSION

An extended DPC algorithm, called SKTDPC, is successfully proposed by K-d tree, sparse search and second-order difference methods. Applying to eight synthetic datasets with two dimensions and six real datasets, comparisons have been carried out between SKTDPC and the six typical clustering algorithms, including FSDPC, the original DPC, DGDPC, DPC-KNN, DBSCAN and K-means algorithms. The main conclusions can be summarized as follows.

Firstly, the algorithmic complexity is obviously reduced by dual accelerations. One is the acceleration for calculation of the local density with a sparse distance matrix, which is attributed to fast search of k nearest neighbors by K-d tree. Another is the acceleration for calculation of the relative-separation by a sparse search strategy with the intersection between the set of k nearest neighbors and the set consisting of the data points with larger local density for any data point. Experimental validation demonstrates that compared with the DPC series algorithms, SKTDPC algorithm can achieve higher clustering efficiency on all datasets. The larger the dataset, the greater the advantage of SKTDPC.

Secondly, experiments indicate that SKTDPC algorithm can realize the best clustering effect in general, compared with the other algorithms. Furthermore, it is indicated that compared with K-means and DBSCAN algorithm, SKTDPC algorithm has a relatively stronger general applicability for datasets with arbitrary distribution characteristics,

even though they have better clustering efficiency in some cases.

Finally, the second-order difference method for decision values is adopted to determine the location of the mutation-point adaptively, which avoids the trouble of selecting the cluster centers manually. It is also verified by experiments that the present method can produce the correct number of cluster centers automatically.

For future work, datasets with insufficient target data, high complexity or high sparsity will be further explored and studied to enhance the application ability of clustering algorithm.

REFERENCES

[1] Q. Zhang, C. Zhu, L. T. Yang, Z. Chen, L. Zhao, and P. Li, "An incremental CFS algorithm for clustering large data in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1193–1201, Jun. 2017, doi: 10.1109/TII.2017.2684807.

[2] A. Fahad, N. Alshatri, Z. Tari, and A. Alamri, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 267–279, Sep. 2014, doi: 10.1109/TETC.2014.2330519.

[3] B. J. Jain, "Consistency of mean partitions in consensus clustering," *Pattern Recognit.*, vol. 71, pp. 26–35, Nov. 2017, doi: 10.1016/j.patcog.2017.04.021.

[4] M. G. Omran, A. P. Engelbrecht, and A. Salman, "An overview of clustering methods," *Intell. Data Anal.*, vol. 11, no. 6, pp. 583–605, 2007, doi: 10.1016/j.patcog.2017.04.021.

[5] J. Liu and C. Zhao, "Density gain-rate peaks for spectral clustering," *IEEE Access*, vol. 9, pp. 46000–46010, 2021, doi: 10.1109/ACCESS.2021.3066498.

[6] Y.-W. Chen, D.-H. Lai, H. Qi, J.-L. Wang, and J.-X. Du, "A new method to estimate ages of facial image for large database," *Multimed. Tools Appl.*, vol. 75, no. 5, pp. 2877–2895, Feb. 2016, doi: 10.1007/s11042-015-2485-9.

[7] Y. Wang, Z. Wei, and J. Yang, "Feature trend extraction and adaptive density peaks search for intelligent fault diagnosis of machines," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 105–115, Jan. 2019, doi: 10.1109/TII.2018.2810226.

[8] M. Xu, Y. Li, R. Li, F. Zou, and X. Gu, "EADP: An extended adaptive density peaks clustering for overlapping community detection in social networks," *Neurocomputing*, vol. 337, pp. 287–302, Apr. 2019, doi: 10.1016/j.neucom.2019.01.074.

[9] J. Chen and P. S. Yu, "A domain adaptive density clustering algorithm for data with varying density distribution," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2310–2321, Jun. 2021, doi: 10.1109/TKDE.2019.2954133.

[10] H. Yu, L. Chen, and J. Yao, "A three-way density peak clustering method based on evidence theory," *Knowl.-Based Syst.*, vol. 211, Jan. 2021, Art. no. 106532, doi: 10.1016/j.knosys.2020.106532.

[11] X. Yang, Z. Cai, R. Li, and W. Zhu, "GDPC: Generalized density peaks clustering algorithm based on order similarity," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 3, pp. 719–731, Mar. 2021, doi: 10.1007/s13042-020-01198-0.

[12] X. Li and K.-C. Wong, "Evolutionary multiobjective clustering and its applications to patient stratification," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1680–1693, May 2019, doi: 10.1109/TCYB.2018.2817480.

[13] M. d'Errico, E. Facco, A. Laio, and A. Rodriguez, "Automatic topography of high-dimensional data sets by non-parametric density peak clustering," *Inf. Sci.*, vol. 560, pp. 476–492, Jun. 2021, doi: 10.1016/j.ins.2021.01.010.

[14] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014, doi: 10.1126/science.1242072.

[15] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-means clustering algorithm," *J. Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979, doi: 10.2307/2346830.

[16] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.

[17] J. Hou and H. Cui, "Experimental evaluation of a density kernel in clustering," in *Proc. 7th Int. Conf. Intell. Control Inf. Process. (ICICIP)*, Siem Reap, Cambodia, Dec. 2016, pp. 55–59.

[18] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on K-nearest neighbors and principal component analysis," *Knowl.-Based Syst.*, vol. 99, pp. 135–145, May 2016, doi: 10.1016/j.knosys.2016.02.001.

[19] J. Xie, H. Gao, W. Xie, X. Liu, and P. W. Grant, "Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors," *Inf. Sci.*, vol. 354, pp. 19–40, Aug. 2016, doi: 10.1016/j.ins.2016.03.011.

[20] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Inf. Sci.*, vol. 450, pp. 200–226, Jun. 2018, doi: 10.1016/j.ins.2018.03.031.

[21] Y. Liu, D. Liu, F. Yu, and Z. Ma, "A double-density clustering method based on 'nearest to first in' strategy," *Symmetry*, vol. 12, no. 5, p. 747, May 2020, doi: 10.3390/sym12050747.

[22] S. Gong and Y. Zhang, "EDDPC: An efficient distributed density peaks clustering algorithm," *J. Comput. Res. Dev.*, vol. 53, no. 6, pp. 1400–1409, 2016, doi: 10.7544/issn1000-1239.2016.20150616.

[23] L. Bai, X. Cheng, J. Liang, H. Shen, and Y. Guo, "Fast density clustering strategies based on the K-means algorithm," *Pattern Recognit.*, vol. 71, pp. 375–386, Nov. 2017, doi: 10.1016/j.patcog.2017.06.023.

[24] X. Xu, S. Ding, M. Du, and Y. Xue, "DPCG: An efficient density peaks clustering algorithm based on grid," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 5, pp. 743–754, 2018, doi: 10.1007/s13042-016-0603-2.

[25] X. Xu, S. Ding, and Z. Shi, "An improved density peaks clustering algorithm with fast finding cluster centers," *Knowl.-Based Syst.*, vol. 158, pp. 65–74, Oct. 2018, doi: 10.1016/j.knosys.2018.05.034.

[26] X. Xu, S. Ding, Y. Wang, L. Wang, and W. Jia, "A fast density peaks clustering algorithm with sparse search," *Inf. Sci.*, vol. 554, pp. 61–83, Apr. 2021, doi: 10.1016/j.ins.2020.11.050.

[27] W. Tong, S. Liu, and X.-Z. Gao, "A density-peak-based clustering algorithm of automatically determining the number of clusters," *Neurocomputing*, vol. 458, pp. 655–666, Oct. 2021, doi: 10.1016/j.neucom.2020.03.125.

[28] Y. Lv, M. Liu, and Y. Xiang, "Fast searching density peak clustering algorithm based on shared nearest neighbor and adaptive clustering center," *Symmetry*, vol. 12, no. 12, p. 2014, Dec. 2020, doi: 10.3390/sym12122014.

[29] K. G. Flores and S. E. Garza, "Density peaks clustering with gap-based automatic center detection," *Knowl.-Based Syst.*, vol. 206, Oct. 2020, Art. no. 106350, doi: 10.1016/j.knosys.2020.106350.

[30] J.-L. Lin, J.-C. Kuo, and H.-W. Chuang, "Improving density peak clustering by automatic peak selection and single linkage clustering," *Symmetry*, vol. 12, no. 7, p. 1168, Jul. 2020, doi: 10.3390/sym12071168.

[31] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975, doi: 10.1145/361002.361007.

[32] J.-Y. Chen and H.-H. He, "A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data," *Inf. Sci.*, vol. 345, pp. 271–293, Jun. 2016, doi: 10.1016/j.ins.2016.01.071.

[33] M. Wang, F. Min, Y.-X. Wu, and Z.-H. Zhang, "Active learning through density clustering," *Expert Syst. Appl.*, vol. 85, pp. 305–317, Nov. 2017, doi: 10.1016/j.eswa.2017.05.046.

[34] J. Xu, G. Wang, and W. Deng, "DenPEHC: Density peak based efficient hierarchical clustering," *Inf. Sci.*, vol. 373, pp. 200–218, Dec. 2016, doi: 10.1016/j.ins.2016.08.086.

[35] S. Ding, X. Xu, and Y. Wang, "Optimized density peaks clustering algorithm based on dissimilarity measure," *J. Softw.*, vol. 31, no. 12, pp. 1–13, Nov. 2020, doi: 10.13328/j.cnki.jos.005813.

[36] J. Xu, G. Wang, T. Li, W. Deng, and G. Gou, "Fat node leading tree for data stream clustering with density peaks," *Knowl.-Based Syst.*, vol. 120, pp. 99–117, Mar. 2017, doi: 10.1016/j.knosys.2016.12.025.

[37] X. Xu, Y. Ju, Y. Liang, and P. He, "Manifold density peaks clustering algorithm," in *Proc. 3rd Int. Conf. Adv. Cloud Big Data*, Yangzhou, China, Oct. 2015, pp. 311–318.

[38] Z. Zhang, Q. Zhu, F. Zhu, J. Li, D. Cheng, Y. Liu, and J. Luo, "Density decay graph-based density peak clustering," *Knowl.-Based Syst.*, vol. 224, Jul. 2021, Art. no. 107075, doi: 10.1016/j.knosys.2021.107075.

[39] C. Ren, L. Sun, Y. Yu, and Q. Wu, "Effective density peaks clustering algorithm based on the layered K-nearest neighbors and sub-cluster merging," *IEEE Access*, vol. 8, pp. 123449–123468, 2020, doi: 10.1109/ACCESS.2020.3006069.

[40] Z. Bian, F.-L. Chung, and S. Wang, "Fuzzy density peaks clustering," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 7, pp. 1725–1738, Jul. 2021, doi: 10.1109/TFUZZ.2020.2985004.

[41] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, p. 4, Mar. 2007, doi: 10.1145/1217299.1217303.

[42] L. Fu, J. Qu, and H. Chen, "Mechanical drilling of printed circuit boards: The state-of-the-art," *Circuit World*, vol. 33, no. 4, pp. 3–8, Nov. 2007, doi: 10.1108/03056120710836882.

[43] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognit.*, vol. 41, no. 1, pp. 191–203, Jan. 2008, doi: 10.1016/j.patcog.2007.04.010.

[44] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 9, pp. 1273–1280, Sep. 2002, doi: 10.1109/TPAMI.2002.1033218.

[45] Y. Xia, G. Wang, and S. Gao, "An efficient clustering algorithm for 2D multi-density dataset in large database," in *Proc. Int. Conf. Multimedia Ubiquitous Eng. (MUE)*, Seoul, South Korea, 2007, pp. 78–82.

[46] A. Asuncion and D. Newman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 2007.

[47] S. Ding, H. Jia, and Z. Shi, "Spectral clustering algorithm based on adaptive Nystrom sampling for big data analysis," *J. Softw.*, vol. 25, no. 9, pp. 2037–2049, 2014, doi: 10.13328/j.cnki.jos.004643.

[48] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Jan. 2010.

[49] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New J. Phys.*, vol. 11, no. 3, 2009, Art. no. 033015, doi: 10.1088/1367-2630/11/3/033015.

[50] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *J. Amer. Stat. Assoc.*, vol. 78, no. 383, pp. 553–569, Sep. 1983, doi: 10.1080/01621459.1983.10478008.

YUNXIAO SHAN was born in 1995. She is currently pursuing the Ph.D. degree with the School of Science, Harbin University of Science and Technology, Harbin, China. Her current research interests include machine learning, data mining, and fuzzy logic.

SHU LI was born in 1980. He received the Ph.D. degree from Tianjin University, Tianjin, China. He is currently a Professor with the School of Electrical and Electronic Engineering, Harbin University of Science and Technology, China. As the first author or corresponding author, he has published over 20 SCI indexed articles in the internationally renowned journals. He also has obtained six software copyrights. His research interests include machine learning, data mining, and modeling and calculation for phase transition theory.

FUXIANG LI was born in 1972. He received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China. He is currently a Professor with the School of Science, Harbin University of Science and Technology, Harbin. He has published more than 20 papers. His research interests include nonlinear numerical analysis, computational mathematics, and machine learning.

MING ZHOU was born in 1997. She is currently pursuing the M.S. degree with the School of Science, Harbin University of Science and Technology, Harbin, China. Her current research interests include machine learning, mathematics of computation, and nonlinear numerical analysis.

YUXIN CUI was born in 1995. She is currently pursuing the Ph.D. degree with the School of Science, Harbin University of Science and Technology, Harbin, China. Her current research interests include machine learning and stochastic logic system analysis.

SHUAI LI was born in 1998. He is currently pursuing the Ph.D. degree with the School of Materials Science and Chemical Engineering, Harbin University of Science and Technology, Harbin, China. His current research interests include machine learning, material research, and high entropy alloy.

XIANG LI was born in 1997. He is currently pursuing the M.S. degree with the School of Science, Harbin University of Science and Technology, Harbin, China. His current research interests include machine learning and data mining.

● ● ●