Information Sciences 450 (2018) 200–226

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Shared-nearest-neighbor-based clustering by fast search and find of density peaks☆

Rui Liu[a], Hong Wang[a,b,c,*], Xiaomei Yu[a,b,c]

[a] School of Information Science and Engineering, Shandong Normal University, Jinan, Shandong 250358, China
[b] Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan, Shandong 250014, China
[c] Institute of Life Science, Shandong Normal University, Jinan, Shandong 250014, China

ARTICLE INFO

Article history:
Received 10 October 2017
Revised 31 January 2018
Accepted 11 March 2018
Available online 20 March 2018

Keywords:
Clustering
Shared nearest neighbor
Density peaks
Fast search clustering
Local density

ABSTRACT

Clustering by fast search and find of density peaks (DPC) is a new clustering method that was reported in Science in June 2014. This clustering algorithm is based on the assumption that cluster centers have high local densities and are generally far from each other. With a decision graph, cluster centers can be easily located. However, this approach suffers from certain disadvantages. First, the definition of the local density and distance measurement is too simple; therefore, the DPC algorithm might perform poorly on complex datasets that are of multiple scales, cross-winding, of various densities, or of high dimensionality. Second, the one-step allocation strategy is not robust and has poor fault tolerance. Thus, if a point is assigned incorrectly, then the subsequent allocation will further amplify the error, resulting in more errors, which will have a severe negative impact on the clustering results. Third, the cutoff distance d_c is generally difficult to determine since the range of each attribute is unknown in most cases. Even when being normalized or using the relative percentage method, a small change in d_c will still cause a conspicuous fluctuation in the result, and this is especially true for real-world datasets. Considering these drawbacks, we propose a shared-nearest-neighbor-based clustering by fast search and find of density peaks (SNN-DPC) algorithm. We present three new definitions: SNN similarity, local density ρ and distance from the nearest larger density point δ. These definitions take the information of the nearest neighbors and the shared neighbors into account, and they can self-adapt to the local surroundings. Then, we introduce our two-step allocation method: inevitably subordinate and possibly subordinate. The former quickly and accurately recognizes and allocates the points that certainly belong to one cluster by counting the number of shared neighbors between two points. The latter assigns the remaining points by finding the clusters to which more neighbors belong. The algorithm is benchmarked on publicly available synthetic datasets, UCI real-world datasets and the Olivetti Faces dataset, which are often used to test the performance of clustering algorithms. We compared the results with those of DPC, fuzzy weighted K-nearest neighbors density peak clustering (FKNN-DPC), affinity propagation (AP), ordering points to identify the clustering structure (OP-

☆ The source code of this paper is available at https://github.com/liurui39660/SnnDpc

* Corresponding author at: School of Information Science and Engineering, Shandong Normal University, Jinan, Shandong 250358, China.
E-mail addresses: xxliuruiabc@163.com (R. Liu), wanghong106@163.com (H. Wang).

https://doi.org/10.1016/j.ins.2018.03.031

TICS), density-based spatial clustering of applications with noise (DBSCAN), and K-means. The metrics used are adjusted mutual information (AMI), adjusted Rand index (ARI), and Fowlkes–Mallows index (FMI). The experimental results prove that our method can recognize clusters regardless of their size, shape, and dimensions; is robust to noise; and is remarkably superior to DPC, FKNN-DPC, AP, OPTICS, DBSCAN, and K-means.

1. Introduction

Clustering, also known as unsupervised classification, divides objects into subsets or clusters according to the similarity measure of the object (physical or abstract) such that the objects within the cluster have a high degree of similarity and that the objects belonging to different clusters have a high degree of dissimilarity [36]. Cluster analysis plays an important role in the fields of social sciences, psychology, biology, statistics, pattern recognition and information retrieval as an important basis for other problems.

Cluster analysis is a challenging problem in data mining and machine learning. Over the past few decades, a number of clustering algorithms have been developed for different types of applications. Typical algorithms include K-means [25] and K-medoids [23] based on partitioning, CURE [18] and BIRCH [45] based on hierarchy, DBSCAN [11] and OPTICS [2] based on density, WaveCluster [30] and STING [41] based on grids, statistical clustering [8] based on models, and spectral clustering [39] based on graph theory. In recent years, with the advancements in cluster analysis, some new clustering methods, such as subspace clustering [1], ensemble clustering [35], and deep embedded clustering [43], have been proposed. The performances of these algorithms are different. The classical K-means clustering algorithm achieves good clustering results on datasets with convex spherical structures. Although DBSCAN provides good clustering results on irregular clusters and coiled clusters and has a strong anti-noise capability, for variable-density clusters and high-dimensional data, the clustering result is poor [36,42]. Moreover, selecting the radius and threshold also represents a difficult problem for DBSCAN.

In June 2014, Rodriguez et al. reported the DPC algorithm (clustering by fast search and find of density peaks) [28] in the well-known scientific journal Science. DPC is a new clustering algorithm based on density and distance. Compared with traditional clustering algorithms, the DPC algorithm has many advantages, including the following:

1. The algorithm is simple and efficient, and it can quickly find the high density peak point (cluster center) without iteratively calculating the objective function.
2. The DPC algorithm is suitable for cluster analysis on large-scale data.

Because of the above advantages, in a short period of three years, the DPC algorithm has become widely used in computer vision [32], image recognition [7], text mining [46] and other fields.

Although the DPC algorithm has obvious advantages over other clustering algorithms, it has the following shortcomings:

1. The definition of the local density and distance measurement is too simple; therefore, the clustering result of the DPC algorithm might be poor when working with complex datasets that are of multiple scales, cross-winding, of various densities, or of high dimensions.
2. The allocation strategy is sensitive and has poor fault tolerance. Thus, if a point is assigned incorrectly, then the subsequent allocation will further amplify the error, resulting in more errors that will have a serious negative impact on the clustering results.
3. The cutoff distance, d_c, is generally difficult to determine since the range of each attribute is unknown in most cases. Moreover, even if being normalized or using the relative percentage method, a small change in d_c will still cause a conspicuous fluctuation in results.

To solve the above problems, this paper proposes the shared-nearest-neighbor-based clustering by fast search and find of density peaks (SNN-DPC) algorithm. The main innovations of the SNN-DPC algorithm include the following:

1. A similarity measurement based on shared neighbors is proposed. This criterion can be used to calculate the similarity between points according to the shared neighbor information.
2. A local density metric of points based on shared neighbors is proposed. This criterion can be applied not only to simple datasets but also to complex datasets that are of multiple scales, cross-winding, of various densities, or of high dimensions.
3. An adaptive metric of distance from the nearest larger density point is proposed. This metric can be adjusted according to the local density information to ensure that the correct points are more easily chosen as the centroid.
4. A novel and fast density peak clustering algorithm based on the new density and similarity measure is proposed. This algorithm can quickly and accurately find the density peak (center) of each cluster.
5. A two-step point allocation algorithm based on shared neighbors is proposed to improve the probability that the non-central points are correctly allocated and to avoid further errors when a point is incorrectly assigned.

The remainder of this paper is organized as follows. In Section 2, we introduce the research progress related to DPC algorithm. In Section 3, we introduce the basic definitions and processes of the traditional DPC algorithm and reveal some problems within them. In Section 4.1, we propose an effective similarity metric based on shared neighbors to find the hidden structural information and improve the distance measurement between points. In Section 4.2, we propose the SNN-DPC clustering algorithm and point allocation algorithm based on shared neighbors. In Section 5, we compare the SNN-DPC algorithm with other classical clustering algorithms. The results demonstrate that the SNN-DPC algorithm can highlight the feature of the cluster centers and correctly allocate the non-central points. In Section 6, we will discuss the argument of the SNN-DPC algorithm, the sensitivity of the case order, and the time consumption. In Section 7, we summarize the advantages and disadvantages of the SNN-DPC algorithm and point out the direction of our future research.

2. Related works

Clustering analysis is an active field in data mining research, and scholars have proposed many clustering algorithms. These clustering algorithms can be divided into partitioning methods, density-based methods, graph-based methods, hierarchical methods, grid-based methods, model-based methods, propagation-based methods and deep learning methods.

Proposed in 2014, the DPC algorithm is a new clustering method based on density and distance [28].

The research on the DPC clustering algorithm over the past three years primarily concerns the following aspects:

The first aspect is improving the density measure of the DPC algorithm.

Xie proposed a density peak searching and point assigning algorithm based on the fuzzy weighted K-nearest neighbor (FKNN-DPC) [42] technique to solve the problem of the non-uniformity of point density measurements in the DPC algorithm. This method uses K-nearest neighbor information to define the local density of points and to search and discover cluster centers.

Mehmood presented a non-parametric method for DPC via heat diffusion [26] for estimating the probability distribution of a given dataset. Based on heat diffusion in an infinite domain, this method accounts for both the selection of the cutoff distance and the boundary correction of the kernel density estimation.

Du proposed density peak clustering based on K-nearest neighbors (DPC-KNN) [10], which introduces the concept of K-nearest neighbors (KNN) to DPC and provides another option for computing the local density. Note that although the names DPC-KNN and FKNN-DPC are similar, there is no direct relation between these methods.

The second aspect is to automatically determine the numbers of clusters and cluster centers.

In [22], Ju considered that the results obtained through the normalized local density and distance from the nearest larger density point reflect the true characteristics of the data.

Li proposed an automatic clustering algorithm for determining the density of clustering centers [24]. In this algorithm, it is considered that if the shortest distance between a potential cluster center and a known cluster center is less than the cutoff distance d_c, then the potential cluster center is a redundant center. Otherwise, it is regarded as the actual center of another cluster.

Xu proposed a density-peak-based hierarchical clustering method (DenPEHC) [44]. This method generates clusters directly on each possible clustering layer; it also introduces a grid granulation framework to help DenPEHC work with large-scale and high-dimensional datasets.

The third aspect is the application of the DPC algorithm.

Zhong proposed an improved density and distance-based clustering approach [47] to evaluate the performance of EAM. This approach simplifies the current evaluation method such that the commitment in terms of resources for manual data analysis and performance ranking can be significantly reduced.

In [32], Shi et al. applied the DPC algorithm to scene image clustering. In [7], Chen et al. used the DPC algorithm to obtain a possible age estimate based on a face image. In [46], Zhang et al. utilized the density peak clustering algorithm to extract multi-document abstracts. In [40], Wang et al. applied the DPC algorithm and information entropy to detect and remove the noise data field from datasets.

Huang et al. proposed an application of the density peak clustering method [19] to explore the community structure in the network, which detects community centers and performs expansion corresponding to different communities.

Shi et al. introduced clustering methods based on fast search and peak density discovery [31] into overlapping community partitioning problems. By defining a new distance matrix algorithm to overcome the defects of the integer adjacency matrix and describe the possibility of each point belonging to different clusters, the overlapping community was achieved.

3. DPC algorithm and analysis

3.1. Notations

Table 1 presents the major symbols and notations used in the following parts.

Table 1
Notations in traditional DPC and SNN-DPC.

Symbol	Meaning
n	The number of records in the dataset X
m	The actual number of clusters in the dataset X
k	The argument of how many neighbors will be considered
$\rho = (\rho_1, \ldots, \rho_n)$	The local density
$\delta = (\delta_1, \ldots, \delta_n)$	The distance from larger density point
$\gamma = (\gamma_1, \ldots, \gamma_n)$	The decision value, the element-wise product of ρ and δ
$X = (x_1, \ldots, x_n)$	The dataset with x_i as its i-th data point
$\Gamma(i) = (x_1, \ldots, x_k)$	The set of K-nearest neighbors of point i
$L(i) = (x_1, \ldots, x_k)$	The set of k points with the highest similarity to point i
$SNN(i) = (x_1, \ldots)$	The set of shared nearest neighbors of point i; may be an empty set
$D = \{d_{ij}\}$	The distances of the pairs of data points in X
$Sim(i, j)$	The SNN similarity between points i and j

3.2. DPC algorithm

Rodriguez et al. presented their DPC algorithm in the journal Science in 2014. DPC is a new clustering algorithm based on density and distance. This algorithm has its basis on the assumptions that cluster centers are surrounded by neighbors with lower local density and that they are at a relatively larger distance from any points with a higher local density. There are two important metrics to describe each data point i: its local density ρ_i and its distance from the nearest larger density point δ_i.

The DPC algorithm provides two methods for calculating the local density for a data point: the cutoff distance method and the kernel distance method. For a data point i, its local density ρ_i is expressed in Eq. 1 with the cutoff distance method and in Eq. 2 with the kernel distance method.

$$\rho_i = \sum_{i \neq j} \chi(d_{ij} - d_c), \quad \chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{1}$$

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

where d_{ij} is the Euclidean distance between data points i and j. $d_c > 0$, the cutoff distance, is the neighborhood radius of a point, which is set by the user. Thus, the local density ρ_i is positively correlated to the number of points with a distance from i of less than d_c. The most obvious difference between the two methods is that for Eq. 1, ρ_i is a discrete value, whereas for Eq. 2, it is a continuous value.

According to Eqs. 1 and2, ρ_i is sensitive to d_c, but [28] indicates that for large datasets, the influence of d_c is relatively small and vice versa.

Subsequently, DPC defines δ_i as in Eq. 3.

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{3}$$

As shown in 3, δ_i is the minimum distance between point i and another point j whose ρ_j is higher than ρ_i. Moreover, for point i with the highest ρ_i, its δ_i is conventionally defined as Eq. 4.

$$\delta_i = \max_{i \neq j} (\delta_j) \tag{4}$$

As shown in Eqs. 3 and4, the points that are local or global maxima with respect to ρ_i have the maximum δ_i.

Meanwhile, to simplify the selection of the cluster centers, DPC computes the decision value γ_i for each data point i. Eq. 5 presents the definition of γ_i.

$$\gamma_i = \rho_i \times \delta_i \tag{5}$$

According to [28], the DPC clustering process is divided into two steps: finding the density peaks, i.e., the cluster centers, and assigning the remaining points to their corresponding clusters.

In step one, DPC first computes the tuple (ρ_i, δ_i) for each point i. Then, these tuples are used to obtain the decision graph, where the X axis is ρ_i and the Y axis is δ_i. Next, the points with relatively high ρ_i and δ_i values are chosen as centers.

In step two, after the cluster centers are chosen, each remaining point will be assigned to the cluster to which its nearest neighbor of higher density belongs. This information is obtained during the calculation of δ_i.

In addition, similar to other density-based clustering methods, DPC also divides points into three types: boundary points, core points and noise points. The border region for each cluster is the set of points that belong to this cluster, but their

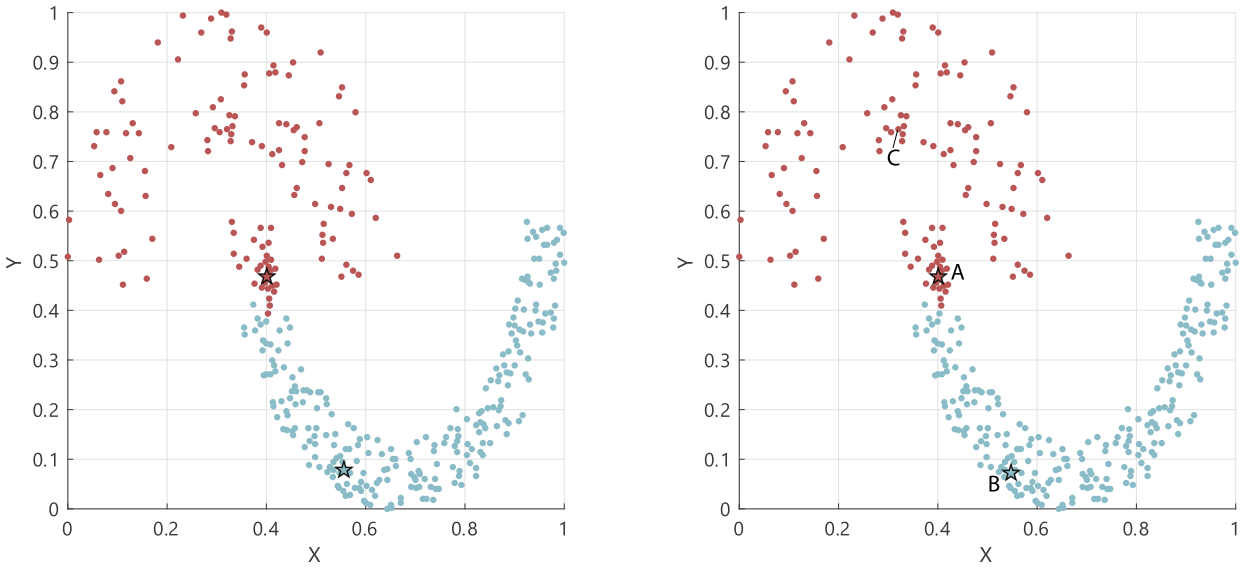


Fig. 1. Results of the traditional DPC algorithm on the Jain dataset.

distances from points in other clusters are smaller than the cutoff distance d_c. The highest ρ_i within the border region of a cluster is denoted as ρ_b. Any point i of the cluster is considered to be the core point if $\rho_i \geq \rho_b$, and the other points are the cluster halo, which can be regarded as noise, including the points in the border region.

3.3. Analysis

Although the experimental results obtained with DPC have shown that it can perform well in many instances, the following drawbacks are obvious.

3.3.1. Various densities

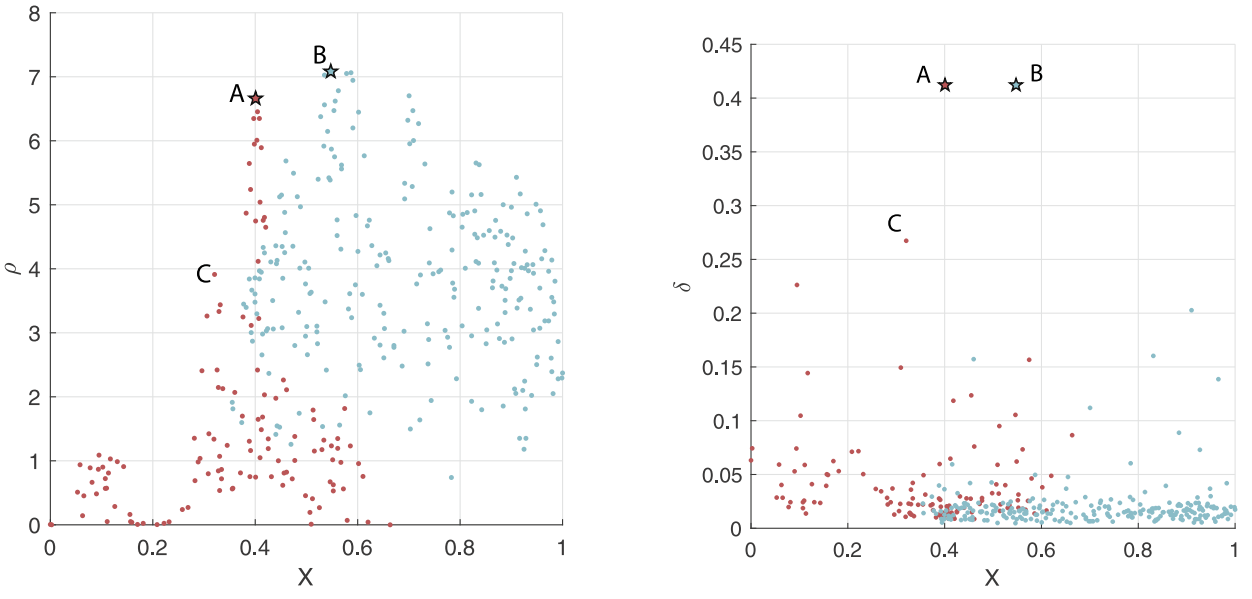
Fig. 1 shows the best clustering results of the DPC algorithm with two types of distance measurements on the classic Jain dataset. As shown in this figure, there are clearly two clusters with different densities in the Jain dataset: the left upper branch with low density and the lower right branch with high density. However, as shown, regardless of the cutoff distance or the kernel distance, the cluster centers are chosen in only one cluster, thus resulting in misclassification.

We can perform a further analysis from Fig. 2, which shows the ρ value and the δ value of each data point using the kernel distance. As shown, the ρ values of half of the points in the high-density cluster are greater than those of the points in the low-density cluster, including its true cluster center, denoted as C. Meanwhile, it can also be observed that the value of δ_A is considerably larger than that of δ_C. The value of δ_A is the distance from point A to point B, which are the top-2 highest densities in the lower right branch, and the value of δ_C is the distance from point C to its nearest point with a larger density. Thus, all the points in the upper branch are at a disadvantage in terms of ρ and δ simultaneously, thereby making it impossible to find an appropriate cluster center in the upper branch, resulting in two cluster centers being selected in the high-density branch, that is, the lower right branch.

From the above example, it is easy to find that, irrespective of which distance measurement is used, the DPC algorithm may not be able to choose the correct cluster center for datasets with variable densities. This is because the measurements of the distance and density are not reasonable without considering the influence of the neighborhoods of the points. On the one hand, the points in the clusters with low density often have very small ρ values. Even large δ values cannot effectively improve their status in the decision graph. On the other hand, the points in high-density branches tend to have higher ρ values, and their δ values are also usually high since branches are generally far from each other. Therefore, these points are easier to be chosen as cluster centers than those in low-density clusters. Consequently, DPC can easily choose an incorrect point as the cluster center, which leads to incorrect clustering results.

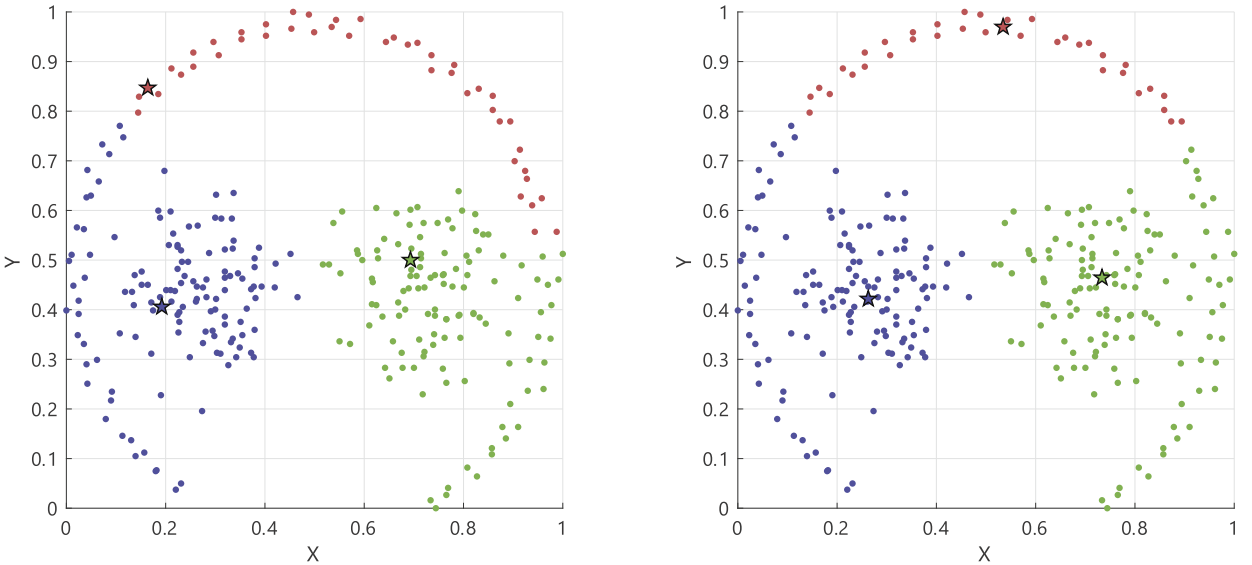
3.3.2. Allocation strategy

Fig. 3 presents the result on the classic Pathbased dataset, where two clusters are surrounded by a ring-shaped cluster. This figure clearly shows that the DPC algorithm can correctly find three cluster centers with both distance measurements: the cutoff distance and the kernel distance. However, there are some differences between these distance measurements. We take the result of the kernel distance as an example. The points are assigned to the correct cluster in the initial allocation



(a) ρ value (b) δ value

Fig. 2. ρ and δ values of the result of the traditional DPC algorithm on the Jain dataset.



(a) Cutoff distance (b) Gaussian kernel distance

Fig. 3. Results of the traditional DPC algorithm on the Pathbased dataset.

process. However, the points on both sides of the ring are assigned to incorrect clusters because they are closer to the centers of the other two clusters. Additionally, due to DPC's one-step allocation strategy, if one of these points is assigned to an incorrect cluster, then the following points with lower ρ values may also be assigned to incorrect clusters, thereby producing catastrophic clustering results. This fact is also the same under the cutoff distance measurement.

Thus, irrespective of which distance measurement is used, for non-convex datasets, the DPC algorithm may not assign the non-center points to the correct cluster. This is because the DPC algorithm assigns every remaining point to the same cluster as its nearest neighbor with higher density in one step. This one-step allocation strategy does not take the neighborhood information into account. When a point is assigned to an incorrect cluster, the subsequent allocation of the points will be misled, resulting in a large allocation error.

To solve the above problems of the DPC algorithm, a density peak clustering algorithm based on shared nearest neighbors (SNN-DPC) is proposed. The SNN-DPC algorithm introduces the concept of shared nearest neighbors (SNN)[21] to improve the definitions of distance, local density ρ and distance from the nearest larger density point δ to handle the problems that arise from not considering the influence of the neighborhoods. Because the ρ and δ defined by the SNN-DPC algorithm reflect the local features of the datasets, they can reflect the natural structure of the data. Therefore, the SNN-DPC algorithm can improve the clustering results on complex datasets such as multi-scale, cross-winding, variable-density and high-dimensional datasets. Simultaneously, the SNN-DPC algorithm retains most of the advantages of the DPC algorithm.

4. SNN-DPC Algorithm

In this section, we present a detailed description of the SNN-DPC algorithm. First, we define the similarity and the density based on shared neighbors. Then, we propose the improved SNN-DPC algorithm, which includes the density peak discovery algorithm and the point allocation algorithm. Finally, we present the analysis of the time and space complexities of the SNN-DPC algorithm.

4.1. Definitions

The DPC algorithm may not produce satisfactory results on some complex datasets because the DPC algorithm directly calculates the distance and density between points but does not focus on the environment in which the points are located. However, the majority of neighbors of a point typically still belong to the same cluster, and this fact can be used to define a more appropriate proximity measurement. Thus, we introduce an indirect distance and density measurement method that considers the effects of points' neighbors. Our approach draws on the concept of shared neighbors to characterize the local density of points and the distance between them.

The basic idea of SNN is that two points are considered more similar if they have more common neighbors. This can be represented by Eq. 6.

Definition 1 (Shared Nearest Neighbors)**.** For any points i and j in dataset X, the set of K-nearest neighbors of point i is $\Gamma(i)$, and the set for j is $\Gamma(j)$; the shared neighbors of point i and point j are their common neighbor sets, expressed as

$$SNN(i, j) = \Gamma(i) \cap \Gamma(j) \tag{6}$$

Definition 2 (SNN Similarity)**.** According to the above basic idea, we first present the formula of the SNN similarity.

For any points i and j in dataset X, their SNN similarity is defined as

$$Sim(i, j) = \begin{cases} \dfrac{|SNN(i, j)|^2}{\sum\limits_{p \in SNN(i,j)} (d_{ip} + d_{jp})}, & \text{if } i, j \in SNN(i, j) \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where d is the distance between points i and j. The SNN similarity is calculated only when point i and point j appear in each other's K-neighbor sets. Otherwise, the SNN similarity between point i and point j is 0.

The non-zero part of Eq. 7 can be expressed in the form of Eq. 8, from which the meaning of SNN similarity can be clearly observed.

$$Sim(i, j) = |SNN(i, j)| \times \frac{1}{\frac{1}{|SNN(i, j)|} \sum\limits_{p \in SNN(i,j)} (d_{ip} + d_{jp})} \tag{8}$$

The left part of the equation represents the number of shared neighbors of points i and j. The right part is the reciprocal of the mean value of the distance from points i and j to all their shared neighbors, which represents the density around the two points to a certain extent. By examining the shared neighbors and the density of the two points simultaneously, SNN similarity can better adapt to a variety of environments.

After defining the SNN similarity of any two points, we use this similarity to calculate the local density ρ_i of point i.

Definition 3 (SNN Local Density)**.** Let point i be any point in dataset X and $L(i) = \{x_1, x_2, \ldots, x_k\}$ be the set of k points with the highest similarity to point i. Then, the local density of point i is defined as the sum of the similarity of k points with the highest similarity to point i. The equation for the local density is as follows:

$$\rho_i = \sum_{j \in L(i)} Sim(i, j) \tag{9}$$

The local density ρ_i of point i has the following properties:

1. When k is small, the number of shared neighbors between point i and one of its neighbors j is smaller, and the distance from point j to point i is closer; thus, k reflects the neighborhood and local density within a smaller neighborhood of i. In contrast, when k is large, it reflects a larger neighborhood of i. Since the distance between points in a low-density cluster is large, changes in k will have a greater impact on low-density clusters.

2. When $|SNN(i, j)|$ is constant, if the distance from i and j to each of their shared neighbors is small, that is, $\sum_{p\in SNN(i,j)}(d_{ip}+d_{jp})$ is small, then ρ_i is larger. In other words, if the distance between i and j is small and the distance from each shared neighbor point to i and j is also small, then the density of point i is large. It can be observed that closer points in space contribute more to ρ_i.
3. When $\sum_{p\in SNN(i,j)}(d_{ip}+d_{jp})$ is constant, if the number of shared neighbors is large between i and j, that is, $|SNN(i, j)|$ is large, then ρ_i is larger. In other words, if most points around i are of the same cluster, then the density of i is larger. It is clear that points belonging to the same cluster contribute more to ρ_i.

In general, the local density ρ not only utilizes the distance information but also obtains information about the cluster structure through the number of SNNs. It fully uncovers the intrinsic relationship between points.

For the distance from the nearest larger density point δ, we add a compensation mechanism based on proximity distance; thus, it may also be high for δ values of points in low-density clusters.

Definition 4 (Distance from Nearest Larger Density Point)**.** Let point i be any point in dataset X; find a point j whose local density is larger than i. Then, minimize the distance between i and j multiplied by the distance from i and j to their K-nearest neighbors; take this as the δ value of point i. The equation for δ is as follows:

$$\delta_i = \min_{j:\rho_j>\rho_i}\left[d_{ij}\left(\sum_{p\in\Gamma(i)} d_{ip}+\sum_{q\in\Gamma(j)} d_{jq}\right)\right] \tag{10}$$

The δ value of the point with the highest local density is the largest δ value among the other points, as shown in Eq. 11.

$$\delta_i = \max_{j\in(X-i)}(\delta_j) \tag{11}$$

The δ value has the following properties:

1. When k increases, the number of nearest neighbors between i and j will increase, and the distances from i and j to their neighbors will also increase, ultimately causing an increase in $d_{ij}(\sum_{p\in\Gamma(i)} d_{ip}+\sum_{q\in\Gamma(j)} d_{jq})$. However, the effect of the increase varies; for clusters of low density, it is notable, whereas for clusters of high density, the impact is much more slight.
2. When d_{ij} is constant, if the distance from i and j to their respective k neighbors is large, then the value of $d_{ij}(\sum_{p\in\Gamma(i)} d_{ip}+\sum_{q\in\Gamma(j)} d_{jq})$ is larger; then, the δ_i value of the candidate is larger. In other words, if the neighbors of i and j are far from them, there will be a higher compensation for the low density and vice versa. In this way, the center of the low-density cluster is more easily found from the decision graph.
3. When $\sum_{p\in\Gamma(i)} d_{ip}+\sum_{q\in\Gamma(j)} d_{jq}$ is constant, if the distance between the points i and j is large, then $d_{ij}(\sum_{p\in\Gamma(i)} d_{ip}+\sum_{q\in\Gamma(j)} d_{jq})$ is larger; thus, the candidate δ_i value is larger. This is based on the assumption that the "distance between cluster centers is relatively large" in traditional DPC. In other words, if a point is close to the nearest larger density point, then the point has a low probability of being a cluster center.

In summary, the distance from a larger density point δ not only determines the distance factor but also considers the neighbor information of each point, which compensates for the point in the low-density cluster and enhances the fairness of the δ value.

For traditional DPC, although [28] has been the subject of many experiments to demonstrate its performance, there are still shortcomings. The traditional DPC algorithm only relies on the distance between points (cutoff distance or Gaussian kernel function distance) to calculate the local density; neither the nearest neighborhood information nor shared neighborhood information is considered. Although this density measurement method allows the DPC algorithm to achieve good performance on some simple datasets, it cannot adequately reflect the complex relationship between points. Thus, it cannot find the real cluster structure implicit in the data or provide satisfactory results when addressing large differences in cluster density, which makes the traditional DPC algorithm suffer from poor performance on datasets that are of multiple scales, cross-winding, high dimensionality, significantly different in terms of cluster density, or of other complicated features.

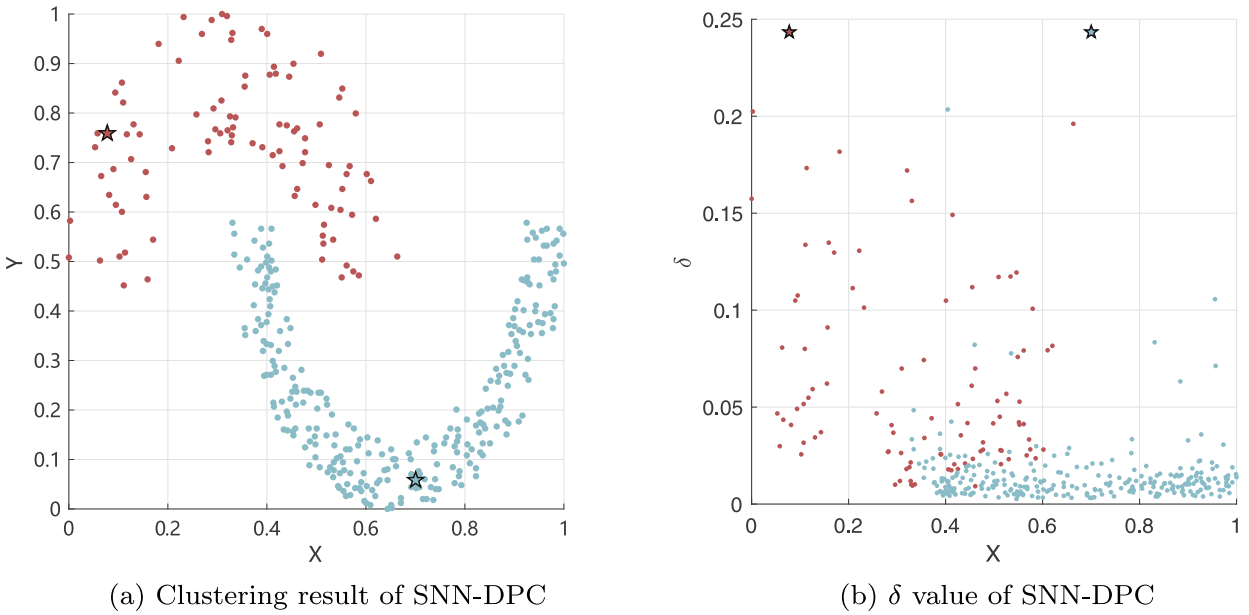
For SNN-DPC, the sum of the distances from points of a low-density cluster to their K-neighbors may be large; thus, they receive greater compensation for their δ value. Fig. 4a and4b show the results of SNN-DPC on the Jain dataset. Compared to Fig. 2b, the δ values of points in the upper branch are generally larger than those of the lower branch. This is because the density of the upper branch is significantly smaller and because the distances from the points to their respective k neighbors are larger; thus, they receive greater compensation. Even if the density is at a disadvantage, the higher δ value still makes the center of the upper branch distinguished in the decision graph.

Next, we present the definition of the decision value γ, which remains unchanged from the traditional DPC algorithm.

Definition 5 (Decision Value)**.** Let point i be any point in point set X; then, the decision value γ_i is the product of the local density ρ_i and the distance from the nearest larger density point δ_i. The formula is shown in Eq. 12.

$$\gamma_i = \rho_i \times \delta_i \tag{12}$$

The role of the γ value is to assist in manually selecting the cluster center point; a larger γ value means larger ρ and δ values, which suggest that the point is more likely to be a cluster center.



(a) Clustering result of SNN-DPC (b) δ value of SNN-DPC

Fig. 4. Result and δ value of SNN-DPC algorithm on the Jain dataset.

In the next section, we will introduce the main process of the SNN-DPC algorithm. The SNN-DPC algorithm refers to the concepts of "inevitable subordinate point" and "possible subordinate point", and for convenience, we will present the definitions here.

Definition 6 (Inevitable Subordinate Point)**.** Assume that point A has been assigned to a cluster and that point B has not yet been assigned; then, B is an inevitable subordinate point of cluster of A if and only if it satisfies

$$|\{p|p \in \Gamma(A) \cap p \in \Gamma(B)\}| \geq k/2 \tag{13}$$

Specifically,

$$|SNN(A, B)| \geq k/2 \tag{14}$$

In other words, if points A and B are considered to belong to the same cluster, at least half of their respective k neighborhoods are shared with both.

Fig. 5 intuitively explains the emergence of this equation, where A is assigned and B is not, and l_A and l_B are two lines vertical to segment AB. We assume that $k = 10$ and that all points are distributed uniformly. In Fig. 5a, both A and B have 10 points as neighbors, among which 5 are shared by both. The intersection of the neighborhoods of A and B is not an empty set, indicating that A and B are close enough to be considered within the same cluster. For Fig. 5b, the situation is quite the opposite. There are many points intermediate between A and B, but the intersection of the neighborhoods of A and B is an empty set, which indicates that A and B are far from each other and may not be considered within the same cluster.

If an unallocated point does not satisfy the condition of inevitable subordination, we call it a possible subordinate point, and we provide the following definition:

Definition 7 (Possible Subordinate Point)**.** Assume that point A has been assigned to a cluster and that point B has not yet been assigned; then, B is a possible subordinate point of cluster of A if and only if it satisfies

$$0 < |\{p|p \in \Gamma(A) \cap p \in \Gamma(B)\}| < k/2 \tag{15}$$

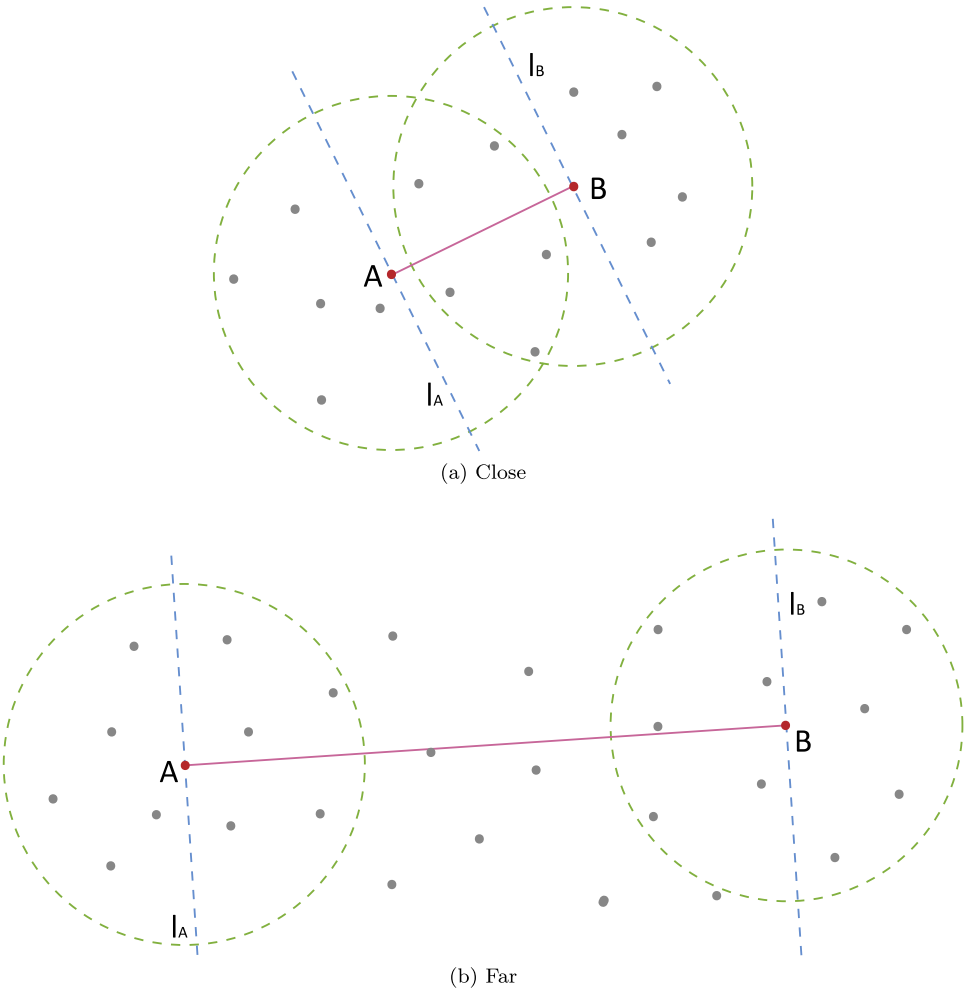
Specifically,

$$0 < |SNN(A, B)| < k/2 \tag{16}$$

In addition to the differences in definition, another difference between inevitable and possible subordinate points is that, under non-extreme cases, each point is able to possibly be subordinate to multiple clusters at the same time, whereas they can be inevitably subordinate to only one cluster.

However, in an extreme case, a point may be inevitably subordinate to two clusters simultaneously. This case will occur only when all the following conditions are met:

1. The argument k is an even number
2. Two points A_1 and A_2 are assigned to different clusters



(a) Close

(b) Far

Fig. 5. Explanation of inevitable subordinate points.

3. One point B remains to be assigned
4. Satisfying Eq. 17

$$\begin{aligned} |\{p|p \in \Gamma(A_1) \cap p \in \Gamma(B)\}| = k/2 \\ |\{p|p \in \Gamma(A_2) \cap p \in \Gamma(B)\}| = k/2 \end{aligned} \tag{17}$$

This circumstance may occasionally partially affect the clustering result. To avoid it, we present some simple but useful methods here:

1. Using odd k argument only
2. When it occurs, temporarily add 1 to k, making it an odd number
3. Assigning B to one of them according to the distance from them
4. Assigning B to one of them according to the distance from their centers.

Apart from them, other complicating methods, such as introducing the concept of fuzzy clustering, also contribute to this situation, and we may consider them in our future works.

Note that for convenience, if the point is not inevitably subordinate to any cluster, we refer to it as a possible subordinate point without specifying the cluster.

4.2. Processes

In the process of the algorithm, SNN-DPC follows the basic idea of the traditional DPC algorithm, but the key steps are improved. The overall process is still divided into three steps: calculation of ρ and δ, selection of cluster centers and allocation of non-central points. The detailed algorithm process is shown below.

Algorithm 1 SNN-DPC.

Require: dataset $X = \{x_1, x_2, \ldots, x_n\}$ (n is the number of entries), number of neighbors k
Ensure: result of clustering $\Psi = \{C_1, C_2, \ldots, C_m\}$ (m is the number of clusters)
1: Initialize dataset X (normalization, etc.)
2: Calculate distance matrix $D^{n\times n} = \{d_{ij}\}^{n\times n}$
3: Calculate similarity matrix according to Eq. 7
4: Calculate local density ρ according to Eq. 9
5: Calculate distance from nearest larger density point δ according to Eqs. 10 and 11
6: Calculate decision value γ according to Eq. 12, sort it in ascending order and record the new order of all elements
7: Construct a $\rho - \delta$ decision graph or γ graph; points in the former are represented as (ρ_i, δ_i), and points in the latter are represented as (i, γ_i), where γ_i is the sorted array obtained from the previous step
8: Select point(s) of large ρ and δ values in the $\rho - \delta$ decision graph or select point(s) of large γ value in the γ graph as cluster centers
9: Apply Algorithm 2 to allocate inevitable subordinate points
10: Apply Algorithm 3 to allocate possible subordinate points

Algorithm 2 Allocation of Inevitable Subordinate Points.

Require: set of centers $Center = \{c_1, c_2, \ldots, c_m\}$, number of neighbors k, distance matrix $D^{n\times n} = \{d_{ij}\}^{n\times n}$
Ensure: preliminary results $\Psi = \{C_1, C_2, \ldots, C_m\}$
1: Initialize queue Q, push all points of $Center$ into Q
2: **while** Q is not empty **do**
3: Pop a point *this* at Q's head
4: **for all** *this*'s unallocated neighbor point *next* **do**
5: **if** *next* does not belong to any cluster and satisfies $|SNN(this, next)| \geq k/2$ **then**
6: Allocate *next* to the cluster of *this*
7: Push *next* to the tail of Q
8: **end if**
9: **end for**
10: **end while**

Line 9 of the SNN-DPC algorithm refers to the allocation algorithm of inevitable subordinate points, and Eqs. 13 and 14 define the criteria of inevitable subordinate points. To allocate inevitable subordinate points, the algorithm uses a breadth-first search, starting from one of the cluster centers and then observing its K-nearest neighbor points in turn. If the number of SNNs is larger than half of k, then the observed point is considered subordinate to the cluster of the current point, and it is pushed into the queue so that we can observe its neighbors in the next few turns. Algorithm 2 describes the entire process.

Line 10 of the SNN-DPC algorithm refers to the allocation algorithm of possible subordinate points, and Eqs. 15 and 16 define the criteria of possible subordinate points. To allocate possible subordinate points, the algorithm traverses all unallocated points and counts the number of neighbors belonging to each cluster to form an allocation matrix. After this, the algorithm looks for the maximum in the matrix and traverses the matrix again, assigning the points represented by the row where the maximum is to the cluster represented by the column. When the traversal is completed, the recognition matrix will be updated and the process will repeat again until the remaining points are allocated. Algorithm 3 describes the entire process.

Using the two-step allocation strategy of "inevitable subordinate points" and "possible subordinate points", as many points as possible are allocated to the correct clusters.

If two points are inevitably subordinate to the same cluster, then they should be close in space. However, if we only take distance as the criterion, it will be susceptible to the randomness of the point distribution. Fig. 6 shows an example to illustrate the advantage of our allocation strategy, where the width and height of each grid are 1, and we assume that the distance between A and B is slightly less than 1. When allocating B, since the nearest point to B is A, according to the traditional DPC's one-step allocation strategy, B will be incorrectly assigned to the red cluster. Moreover, if B_1, B_2, etc. are not yet allocated, then in the subsequent allocation, they will all be assigned to the red cluster because of the misclassification of B. However, for Algorithms 2 and3, we assume that $k = 4$; then, there are $\Gamma(A) = \{A, A_1, A_2, A_3\}$, $\Gamma(B) = \{B, A, B_1, B_2\}$ and $SNN(A, B) = \{A\}$. Since $|SNN(A, B)| < k/2$, B will not be considered inevitably subordinate to the red cluster. During the allocation of possible subordinate points, in the recognition matrix *recog*, since $\Gamma(B) = \{B, A, B_1, B_2\}$, the value of the column corresponding to the red cluster is 1, but the value of the blue cluster is 2; therefore, B will be classified into the blue cluster, which is the correct result. Compared with the traditional DPC algorithm, this method can address clusters of different shapes more flexibly.

Algorithm 3 Allocation of Possible Subordinate Points.

Require: number of neighbors k, preliminary clustering results $\Psi = \{C_1, C_2, \ldots, C_m\}$
Ensure: final clustering results $\Psi = \{C_1, C_2, \ldots, C_m\}$

```
while not all points are allocated do
    Find all unallocated points and re-number them
    Form an allocation matrix M whose rows correspond to unallocated points and columns correspond to clusters
    for all unallocated point p do
        for all neighbor point q of point p do
            +1 to the column corresponding to the cluster that q belongs in the p-th row
        end for
    end for
    Find the max value most in M
    if most > 0 then
        Record the rows and columns that most appears as Row = {r_1, r_2, ..., r_i} and Col = {c_1, c_2, ..., c_i}
        Allocate the r_i-th point to the c_i-th cluster
    else
        k = k + 1
    end if
end while
```

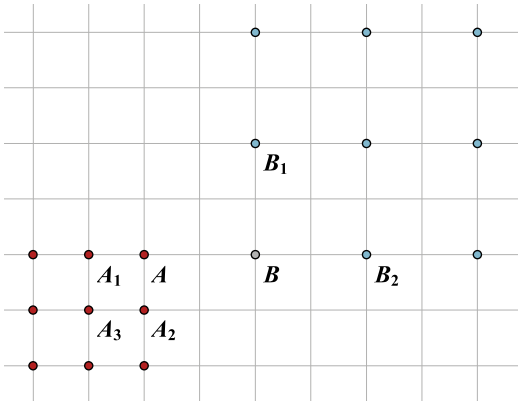
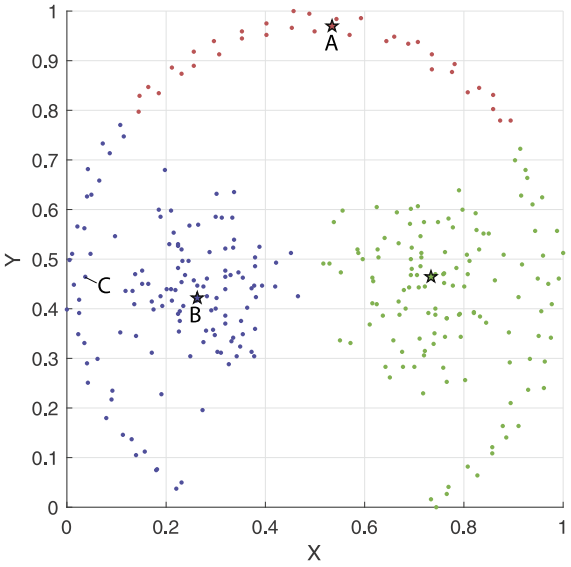


Fig. 6. Example of the advantage of allocating inevitable and possible subordinate points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

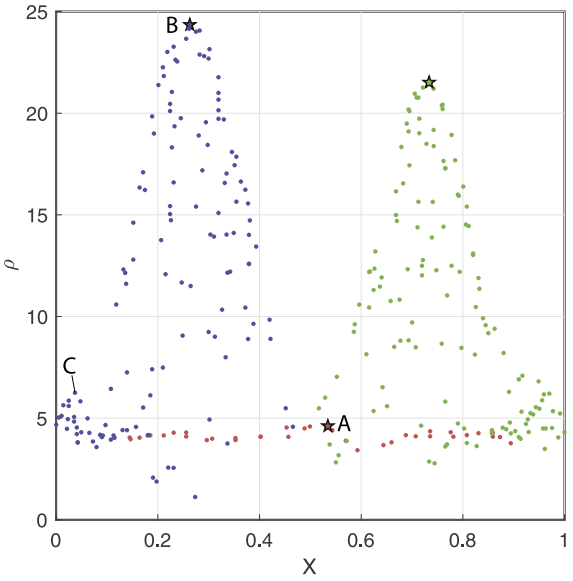
Traditional DPC algorithm uses the method of allocating the points to the cluster of the nearest larger density point without taking the local information or the neighbor information into account. Moreover, since the allocation is performed in descending order of density, it is easy to misclassify the points whose local densities are even higher than some centers. In addition, since each assignment must refer to the previously assigned points, when a point is misclassified, the subsequent points will amplify the error to form a chain error.

Fig. 7a is the classic Pathbased dataset, where two clusters are surrounded by a ring-shaped cluster. Fig. 7b clearly shows that although the traditional DPC algorithm can find a cluster center on each of the three clusters, there is a serious deviation in the allocation of non-center points. At first, the allocation is correct. However, since points are allocated in descending order of their local density, as shown in Fig. 7b, points near the cluster B are of higher densities, thus they are assigned to the cluster B in an early stage. When point C is to be assigned, the less dense cluster A has not extended to the vicinity of C, which causes C to be incorrectly assigned to cluster B. This affects the subsequent allocation, all points that directly or indirectly refer to C are allocated to the cluster of B, leading to the results shown in Fig. 7a.

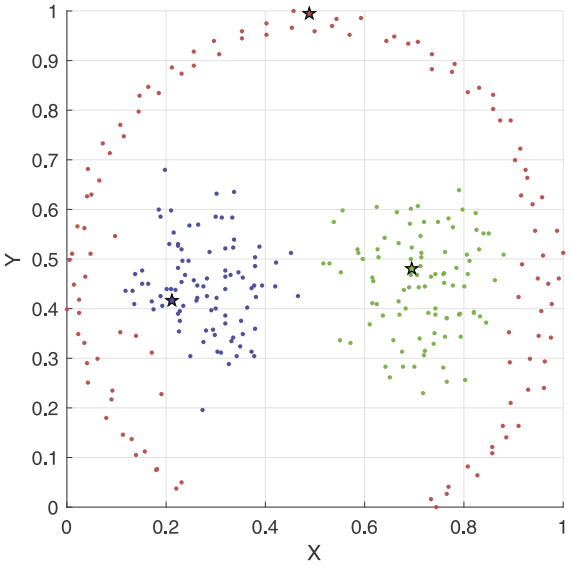
Using the allocation strategy shown in Algorithms 2 and 3, the clustering results obtained from the same dataset as in Fig. 7a are shown in Fig. 7c. Algorithm 2 applies a breadth-first search to ensure that the allocation is gradually extended from the center of each cluster to the surroundings, avoiding using points of other clusters as references. Moreover, Algorithm 2 fully considers the neighbor information, and it allocates points only when the number of SNNs is greater than $k/2$. Algorithm 3 also allocates points that are most likely to be correctly allocated and updates information while allocating points. The allocation of each point in the entire process is cautious; more information is obtained to improve the probability of being correctly assigned, thereby achieving satisfactory results.



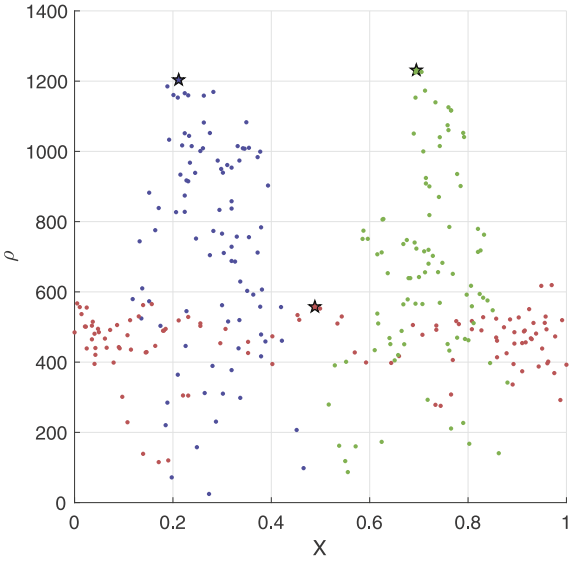
(a) Clustering result of traditional DPC



(b) ρ value of traditional DPC



(c) Clustering result of SNN-DPC



(d) ρ value of SNN-DPC

Fig. 7. Comparison between traditional DPC and SNN-DPC on Pathbased.

4.3. Analysis of complexity

4.3.1. Time complexity

In this part, we set the total number of points to n, the number of cluster centers to m, and the number of neighbors to k.

According to the previous section, the detailed analysis of Algorithm 1 is as follows:

Line 1: the normalization needs to handle every point, that is, approximately $O(n)$.

Line 2: calculating the distance matrix needs $O(n^2)$.

Line 3: calculating the SNN similarity according to Eq. 7. Since the algorithm needs to handle every pair of points, the basic complexity is $O(n^2)$. Calculating the intersection can be performed within $O(k)$ using a hash table. Therefore, the general complexity of this line is $O(kn^2)$.

Line 4 and line 5 calculate ρ and δ according to Eqs. 9 and 10, respectively. The cost of δ is the same as in traditional DPC: $O(n^2)$. For ρ, the algorithm only needs to query the KNN information of each point, that is, $O(kn)$. Thus, the complexity of these two lines is $O(kn^2)$.

Line 6: because of the sort, the complexity is $O(n\log n)$.

Line 7 and line 8 are plotting parts and are omitted.

Line 9: referring to the analysis of Algorithm 2 below, the complexity is $O(mn^2)$.

Line 10: referring to the analysis of Algorithm 3 below, the complexity is $O((k+m)n^2)$.

The overall time complexity of the SNN-DPC algorithm is $O((k+m)n^2)$.

According to the previous section, the detailed analysis of Algorithm 2 is as follows:

Line 1: there are m centers; thus, the cost is $O(m)$.

Line 2: the "while" loop provides $O(n)$ in the worst case.

Line 4: the "for" loop provides $O(n)$ in the worst case.

Line 5: the value of $|SNN(this, next)|$ is obtained from Algorithm 1; thus, every time, the time cost is only $O(1)$ to query.

Line 6 and line 7 are both $O(1)$.

Overall, the time complexity of the "allocation of inevitable subordinate points" is $O(mn^2)$.

According to the previous section, the detailed analysis of Algorithm 3 is as follows:

Line 1: in the worst case, all points are allocated in this algorithm; thus, it provides a multiplier $O(n)$.

Line 2: it costs $O(n)$ to scan all points.

Lines 4, 5 and 6: there are at most n unallocated points, and each of them has k neighbors; therefore, the general complexity is $O(kn)$.

Line 9: to find the maxima, the algorithm needs $O(mn)$.

Lines 11 and 12: the algorithm needs to scan matrix M; therefore, the time complexity is $O(n)$.

The overall time complexity of Algorithm 3 is the basic loop $O(n)$ multiplied by the highest complexity in the loop, that is, $O(kn)$ or $O(mn)$; thus, the overall time complexity is $O(kn^2)$ or $O(mn^2)$, which can be merged as $O((k+m)n^2)$.

Overall, we can conclude that the time complexity of the entire SNN-DPC algorithm is the larger of $O(mn^2)$ and $O(kn^2)$, that is, $O((k+m)n^2)$.

For the time complexity of the traditional DPC algorithm, the time for obtaining the distance between every pair of points is $O(n^2)$. The time to calculate the local density ρ_i is $O(n^2)$ since, for each point i, all other points need to be observed. The time used to compute the distance δ_i is $O(n^2)$ since it also requires evaluating all other points. Finally, the time complexity of assigning non-center points is $O(n)$ since this process is predefined when calculating δ_i. Thus, the time complexity of traditional DPC is $O(n^2)$.

However, both k and m are relatively small numbers compared with n. Therefore, they will not negatively affect the running time to a great extent. We demonstrate that the actual running time of the SNN-DPC is no more than k times the traditional DPC's running time in Section 6.3.

4.3.2. Space complexity

In this part, we set the total number of points to n, the number of cluster centers to m, and the number of neighbors to k.

Apart from the original dataset, SNN-DPC needs to calculate a distance matrix and a similarity matrix, which have complexities of $O(n^2)$. Other data structures, such as the ρ and δ array, are all $O(n)$.

For Algorithm 2, the only data structure needed is a queue, which is $O(n)$. Algorithm 3 requires a matrix, and in the worst case, its complexity is $O(mn)$.

The overall space complexity of SNN-DPC proposed in this article is the same as that of traditional DPC, which is $O(n^2)$.

5. Experiment

To demonstrate the effectiveness of the SNN-DPC algorithm, we use classical synthetic datasets and real-world datasets to test its performance. Moreover, we take FKNN-DPC [42], traditional DPC [28], DBSCAN [11], OPTICS [2], AP [14] and K-means [25] as the control group, where the AP, DBSCAN and K-means algorithms are implemented in the sklearn library [27] of Python and OPTICS is implemented in the pyclustering library. The DPC algorithm is based on the source code provided by the author, but since our datasets do not contain noise, we remove the "Halo" part. For the FKNN-DPC algorithm, since we cannot obtain the source code from the original author, we implement the process referring to [42]. All the results shown are the optimal results after argument tuning.

The synthetic datasets and real-world datasets used in the experiments are presented in Table 2 and Table 3, respectively. Note that the Olivetti Faces dataset is an image dataset; thus, every pixel of the image is regarded as a single attribute.

5.1. Metrics, preprocessing and argument selection

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular, any evaluation metric should not take the absolute values of the cluster labels into account. Rather, it should consider whether this clustering defines separations of the data similar to

Table 2
Synthetic datasets.

Dataset	Source	No. of records	No. of attributes	No. of clusters
Aggregation	[17]	788	2	7
Flame	[16]	240	2	2
Jain	[20]	373	2	2
Pathbased	[5]	300	2	3
R15	[37]	600	2	15
Spiral	[5]	312	2	3
D31	[37]	3100	2	31
DIM512	[13]	1024	512	16
S2	[15]	5000	2	15

Table 3
Real-world datasets.

Dataset	Source	No. of records	No. of attributes	No. of clusters
Iris	[3]	150	4	3
Wine	[3]	178	13	3
WDBC	[34]	569	30	2
Seeds	[6]	210	7	3
Segmentation	[3]	2310	19	7
Libras movement	[9]	360	90	15
Ionosphere	[33]	351	34	2
Waveform	[4]	5000	21	3
Waveform (noise)	[4]	5000	40	3
Ecoli	[3]	336	8	8
Parkinsons	[3]	197	23	2
Dermatology	[3]	366	33	6
Balance Scale	[3]	625	4	3
Spectrometer	[3]	531	102	48
Olivetti faces	[29]	400	92×112	40

some ground-truth set of classes or satisfies some assumption such that members that belong to the same class are more similar than members of different classes according to some similarity metric.

Therefore, in this section, we will evaluate the clustering results using three evaluation indices that are independent of the absolute values of labels: adjusted mutual information (AMI)[38], adjusted Rand index (ARI)[38] and Fowlkes-Mallows index (FMI)[12]. The upper bound of the three indicators is 1, where larger values indicate better clustering results

Before proceeding to the experiments, we need to preprocess the datasets to eliminate the influence of missing values and differences in the ranges of different dimensions. In the subsequent algorithmic process, we will replace the missing values with the mean of all valid values of the same dimension. For the difference in the range of values, we use the "min-max normalization method" shown in Eq. 18 to process the data of each dimension of the data, mapping all the data linearly to the range [0,1], therein eliminating the difference in dimensions and increasing the efficiency of the calculation.

$$x'_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{18}$$

where x'_{ij} is the re-scaled data in the i-th row and j-th column, x_{ij} is the original data in the i-th row and j-th column, and x_j is the original data in the entire j-th column.

To more objectively reflect the actual results of various algorithms, we perform argument tuning on each of the algorithms, thereby ensuring that their best performances are compared.

In detail, for the SNN-DPC and FKNN-DPC algorithms, since they have only one integer argument k, we select the argument continuously from 4 to 50. The lower bound is 4 because, for some datasets, a small k may lead the algorithm to become endless and cause an error. For the upper bound, a large k fails to significantly affect the results of the algorithms, making it less meaningful to be further tested.

The author of the traditional DPC algorithm provides a rule of thumb that one can choose d_c to make the number of neighbors be between 1 and 2% of the total number of points. We will modify this percentage to obtain its best results.

For the DBSCAN and OPTICS algorithms, they both have two arguments, ε and *minpts*. The former is a floating point number, whereas the latter is an integer. Since all the datasets are normalized, we loop the ε from 0.01 to 1, with a step size of 0.01; we loop *minpts* from 1 to 50. The reason for its upper bound is the same as SNN-DPC and FKNN-DPC. Additionally, the OPTICS algorithm implemented in the pyclustering library allows users to specify the number of clusters; thus, we always assign the correct cluster number to it.

The AP algorithm provides only one argument, which in the sklearn library is named "preference". A large preference allows the AP algorithm to choose more points as centers and vice versa. For the AP algorithm, no universal rule can be

Table 4
Performances of different clustering algorithms on different synthetic datasets.

Algorithm	AMI	ARI	FMI	Arg-	AMI	ARI	FMI	Arg-
	Aggregation				Spiral			
SNN-DPC	0.9500	0.9594	0.9681	15	**1.0000**	**1.0000**	**1.0000**	5
FKNN-DPC	0.9775	0.9855	0.9886	20	**1.0000**	**1.0000**	**1.0000**	5
DPC	**1.0000**	**1.0000**	**1.0000**	3.4	**1.0000**	**1.0000**	**1.0000**	1.8
DBSCAN	0.9529	0.9779	0.9827	0.04/6	**1.0000**	**1.0000**	**1.0000**	0.04/2
OPTICS	0.9221	0.9753	0.9807	0.06/10	**1.0000**	**1.0000**	**1.0000**	0.04/1
AP	0.7873	0.7658	0.8150	−1.21	0.2932	0.1569	0.3409	−0.19
K-Means	0.7935	0.7300	0.7884	7	−0.0055	−0.0060	0.3274	3
	Flame				D31			
SNN-DPC	0.8975	0.9502	0.9768	5	**0.9642**	**0.9509**	**0.9525**	41
FKNN-DPC	**1.0000**	**1.0000**	**1.0000**	6	0.9522	0.9275	0.9298	28
DPC	**1.0000**	**1.0000**	**1.0000**	2.8	0.9554	0.9365	0.9385	0.6
DBSCAN	0.8234	0.9388	0.9712	0.09/8	0.8895	0.8078	0.8186	0.04/38
OPTICS	0.6898	0.8968	0.9508	0.10/8	0.8211	0.8673	0.8763	0.03/23
AP	0.4987	0.5403	0.7498	−6.36	0.8367	0.7425	0.7665	0.23
K-Means	0.3863	0.4534	0.7364	2	0.9593	0.9453	0.9470	31
	Jain				DIM512			
SNN-DPC	**1.0000**	**1.0000**	**1.0000**	12	**1.0000**	**1.0000**	**1.0000**	5
FKNN-DPC	0.0562	0.1318	0.6430	10	**1.0000**	**1.0000**	**1.0000**	20
DPC	0.6183	0.7146	0.8819	0.9	**1.0000**	**1.0000**	**1.0000**	0.6
DBSCAN	0.8650	0.9758	0.9906	0.08/2	**1.0000**	**1.0000**	**1.0000**	0.36/2
OPTICS	0.8542	0.9756	0.9905	0.08/1	0.9029	0.9432	0.9478	0.19/1
AP	0.6582	0.7952	0.9212	−1.77	1.0000	1.0000	1.0000	−1.00
K-Means	0.4916	0.5767	0.8200	2	**1.0000**	**1.0000**	**1.0000**	16
	Pathbased				S2			
SNN-DPC	**0.9001**	**0.9294**	**0.9529**	9	0.9386	0.9264	0.9313	35
FKNN-DPC	0.8344	0.8744	0.9165	9	0.9180	0.8889	0.8963	22
DPC	0.5212	0.4717	0.6664	3.8	0.9437	0.9352	0.9395	1.5
DBSCAN	0.8710	0.9011	0.9340	0.08/10	0.8511	0.7485	0.7744	0.04/30
OPTICS	0.4364	0.6364	0.7517	0.06/4	0.6723	0.7713	0.7891	0.03/27
AP	0.5199	0.4775	0.6577	−4.10	0.4616	0.5704	0.6080	−3.06
K-Means	0.5098	0.4613	0.6617	3	**0.9461**	**0.9379**	**0.9420**	15
	R15							
SNN-DPC	**0.9938**	**0.9928**	**0.9933**	10				
FKNN-DPC	0.9907	0.9892	0.9899	27				
DPC	**0.9938**	**0.9928**	**0.9933**	0.6				
DBSCAN	0.9825	0.9819	0.9831	0.04/12				
OPTICS	0.9734	0.9785	0.9799	0.04/11				
AP	0.9907	0.9891	0.9898	−0.17				
K-Means	**0.9938**	**0.9928**	**0.9932**	15				

used to select the parameters; thus, we set the upper limit of the argument search to several times the maximum similarity and gradually narrow the search range.

The only argument of the K-means algorithm is the correct number of clusters; thus, we simply follow this instruction.

5.2. Synthetic datasets

In this part, we select a number of synthetic datasets that are widely used to test a variety of clustering algorithms. These datasets are different in terms of the overall distribution and numbers of points and clusters. They can simulate different situations to compare the performance of various clustering algorithms in different scenarios.

Table 4 shows the clustering results in terms of the AMI, ARI and FMI scores on all synthetic datasets listed in Table 2. For the K-means algorithm, the initial centers are chosen by the K-means++ method rather than a random process. Thus, the results are the same for every run, and there is no need to provide the statistical results. The SNN-DPC, FKNN-DPC and traditional DPC algorithms can visually find the centers of clusters through a decision graph or γ graph. However, in many application scenarios, even if the user knows the exact number of clusters, the clustering algorithm still cannot obtain the desired classification results. These three algorithms have their own methods for calculating the local density and the nearest large density distance; thus, we directly specify the correct number of clusters. Then, we use the decision function γ to automatically determine the clusters and test their performance in the absence of interaction. However, since there is a large density difference in the Jain dataset, we manually choose centers from the decision graph rather than automatically select them based on the γ values.

Next, we will present the clustering results of some datasets in the experiment. The points with different colors in the figures are assigned to different clusters. Except for the DBSCAN and OPTICS algorithms, the cluster centers obtained from other algorithms are represented by stars, while crosses indicate noise determined by the DBSCAN and OPTICS algorithms.

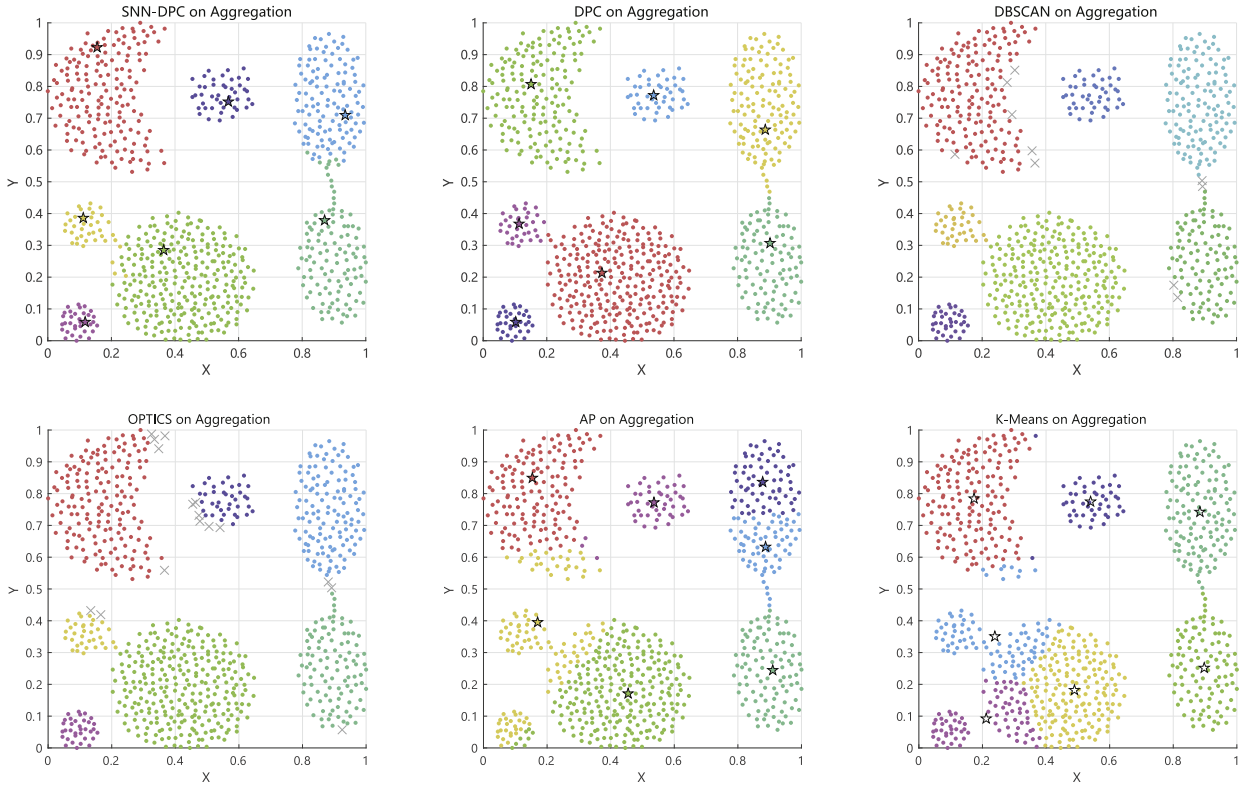


Fig. 8. The clustering results on aggregation by 6 algorithms.

The clustering results in Fig. 8 show that the SNN-DPC, DPC, DBSCAN and OPTICS algorithms can detect clusters in the Aggregation dataset, but the AP and K-means algorithms fail to do so. The SNN-DPC and DPC algorithms recognize both the clusters and the centers. For the DBSCAN and OPTICS algorithms, although some points are marked as noise, the general shapes of each cluster are correct. The AP algorithm successfully finds the correct number of clusters, but it chooses two centers from one cluster, which divides a single cluster into two clusters. The K-means algorithm also encounters a similar problem: it divides a cluster into three clusters, and it chooses the cluster center between two clusters.

Fig. 9 shows the results of each algorithm on the Flame dataset. As shown, the SNN-DPC, traditional DPC, DBSCAN and OPTICS algorithms can correctly identify clusters. Furthermore, SNN-DPC and traditional DPC can also find the cluster centers. For the AP algorithm, although it correctly identifies the upper cluster and chooses the appropriate cluster center, it divides the lower cluster into two clusters, therein choosing a cluster center in each half, which leads to an unsatisfactory result. The K-means algorithm assigns the left end of the lower cluster to the upper cluster such that the entire dataset resembles being slashed from the diagonal, which also leads to serious errors.

The Jain dataset shown in Fig. 10 is a dataset in which two crescent-shaped clusters of different densities are intertwined with each other. As shown, the SNN-DPC algorithm can flexibly address datasets of this type; thus, the two clusters are perfectly distinguished. For the other algorithms, the results can be roughly divided into two types. The first type consists of the DBSCAN and OPTICS algorithms, which can accurately identify the lower clusters; however, both algorithms mistakenly divide the left end of the upper clusters into a new cluster. This type of error is closely related to the inability of traditional density-based clustering algorithms to handle variable-density clusters. Although OPTICS has been optimized for such problems, they cannot be completely avoided. The second type of result is that obtained by the traditional DPC, AP, and K-means algorithms, which all incorrectly assign the left end of the lower cluster to the upper cluster. If we inspect the results carefully, we can observe that the traditional DPC algorithm selects two cluster centers in the lower cluster, leaving the upper cluster with no center, and the AP algorithm incorrectly allocates some points of the upper cluster to the lower cluster, which is also beyond our understanding.

Fig. 11 shows the results of the Pathbased dataset. As shown, the AP algorithm cannot accurately detect cluster centers, considering the lower right end of the half-ring cluster as a new cluster. For the DPC and K-means algorithms, although they can correctly select the cluster centers, during the process of point allocation, the left and right sides of the half-ring clusters are incorrectly assigned to the other two clusters, leaving the half-ring cluster only a small part on the top. The AP algorithm also embodies this problem. For the DBSCAN algorithm, although the other two clusters are correctly assigned, the entire half-circle cluster is considered as noise since its density is far less than the *Minpts*. The OPTICS algorithm does

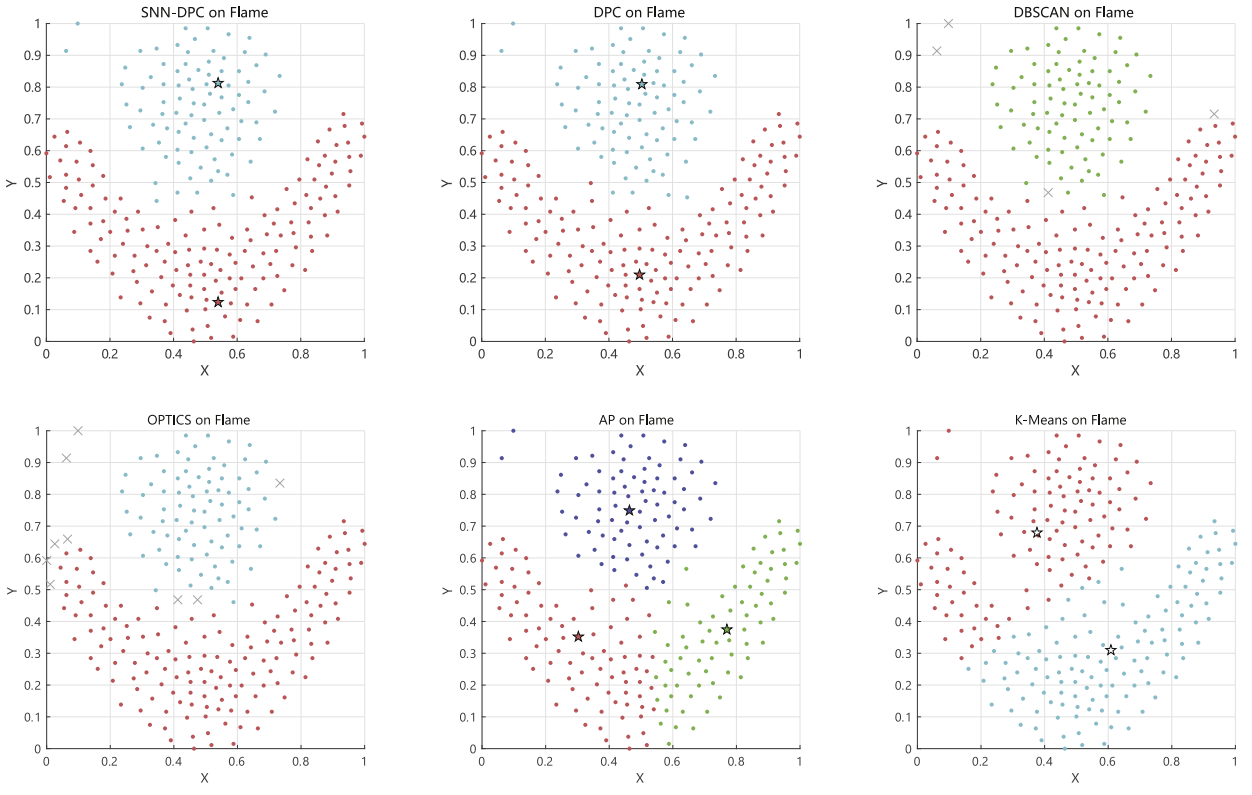


Fig. 9. The clustering results on flame by 6 algorithms.

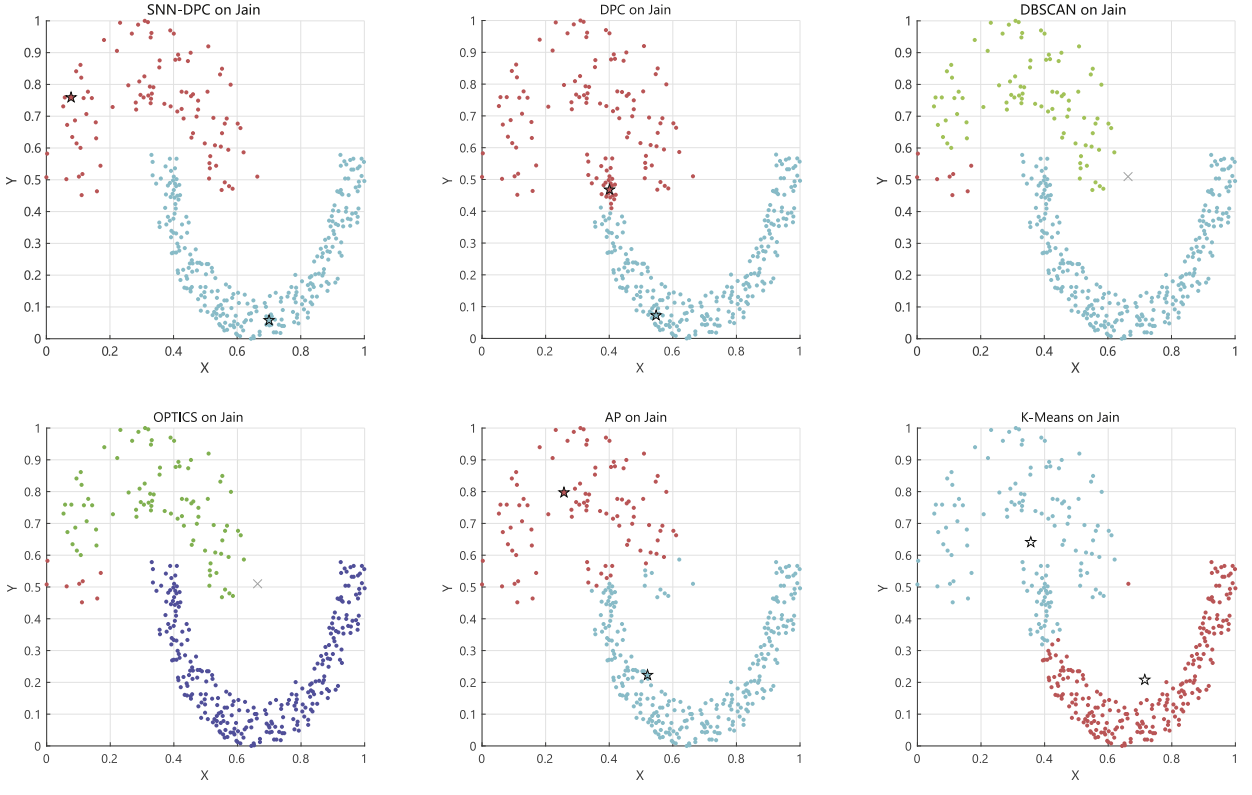


Fig. 10. The clustering results on Jain by 6 algorithms.

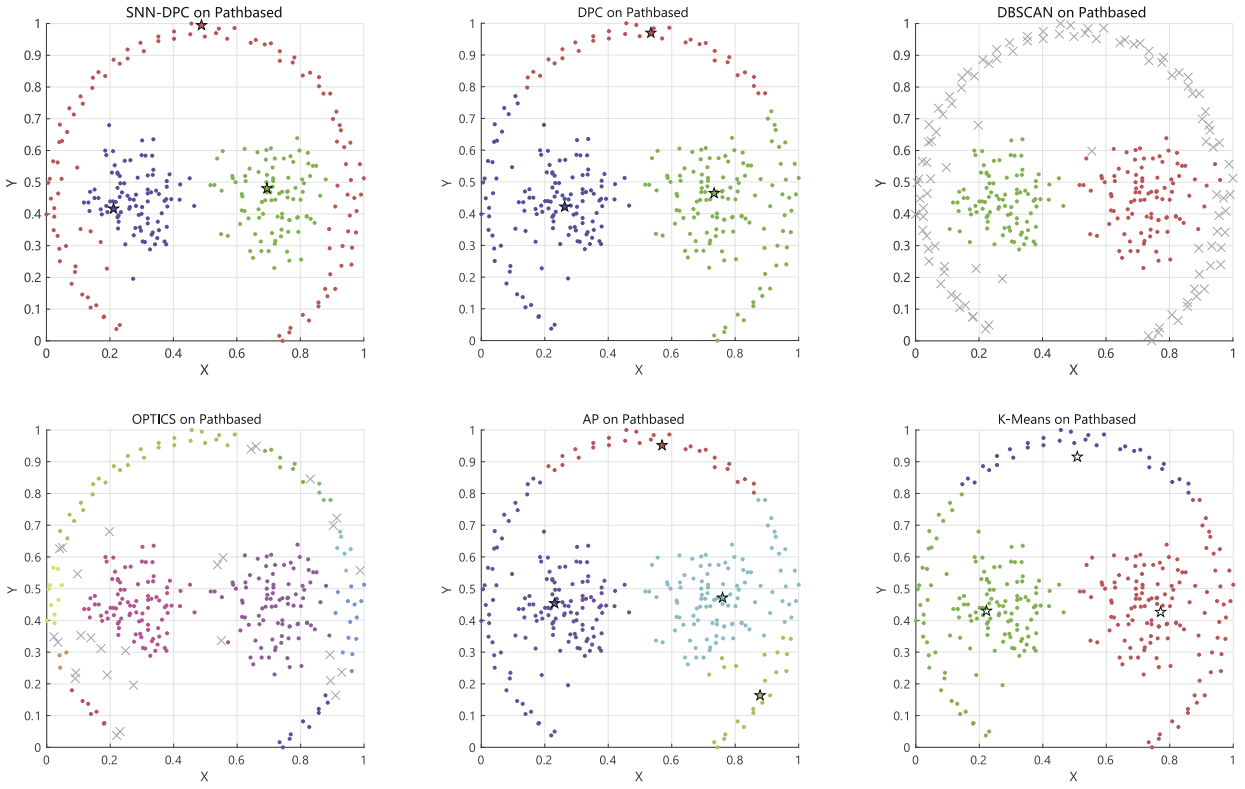


Fig. 11. The clustering results on pathbased by 6 algorithms.

not ignore the half-ring cluster, but it is split into several small clusters. Only the SNN-DPC algorithm can ensure that all three clusters are correctly distinguished.

Fig. 12 displays the results on the R15 dataset. The distribution of points makes it the most straightforward dataset for all the algorithms. Although there are some small defects among them, all the algorithms can recognize both the clusters and centers.

The clustering shown in Fig. 13 demonstrates the ability to address cross-winding datasets. The results of SNN-DPC, traditional DPC, DBSCAN and OPTICS are all perfect. Additionally, notice that the centers of SNN-DPC and traditional DPC are not the same. When using the improved ρ and δ, the centers of SNN-DPC are closer to the end of each cluster, leaving it with fewer chances to specify an incorrect cluster center. The effect is particularly conspicuous when the dataset becomes more complex and has a higher density. In terms of the AP algorithm, since the number of clusters is formed during the process rather than assigned by the user, it marks numerous centers on each branch. This also demonstrates a shortcoming of the AP algorithm. For K-means, it is evident that the dataset is evenly divided into three parts, and the cluster center is chosen from the center of each part. Apparently, even if we pass the correct number of clusters to K-means, it cannot efficiently address the cross-winding clusters.

Fig. 14 shows the results on the D31 dataset. In general, all the algorithms behave similarly on the D31 dataset. It appears that each algorithm can correctly find clusters and reasonably allocate points, but a closer examination reveals that, to different extents, OPTICS and DBSCAN mark some points as noise, whereas the AP algorithm mistakenly divides some clusters into two clusters. However, for the K-means algorithm, the result is almost as good as the SNN-DPC algorithm and the traditional DPC algorithm.

For the S2 dataset shown in Fig. 15, only the SNN-DPC, traditional DPC, and K-means algorithms correctly identify all clusters. Neither the DBCAN algorithm nor the OPTICS algorithm can correctly distinguish the three clusters in the lower right corner, and they all mark many boundary points as noise. For the AP algorithm, an anomaly occurred. Regardless of how we adjust the value of the “preference” argument, the AP algorithm resulted in far more than 15 cluster centers.

In conclusion, it can be observed that SNN-DPC performs better than the other algorithms on several test cases; only a few errors occur in the boundary point assignment on the Aggregation, Flame and S2 datasets, making the results slightly poorer than those of other algorithms.

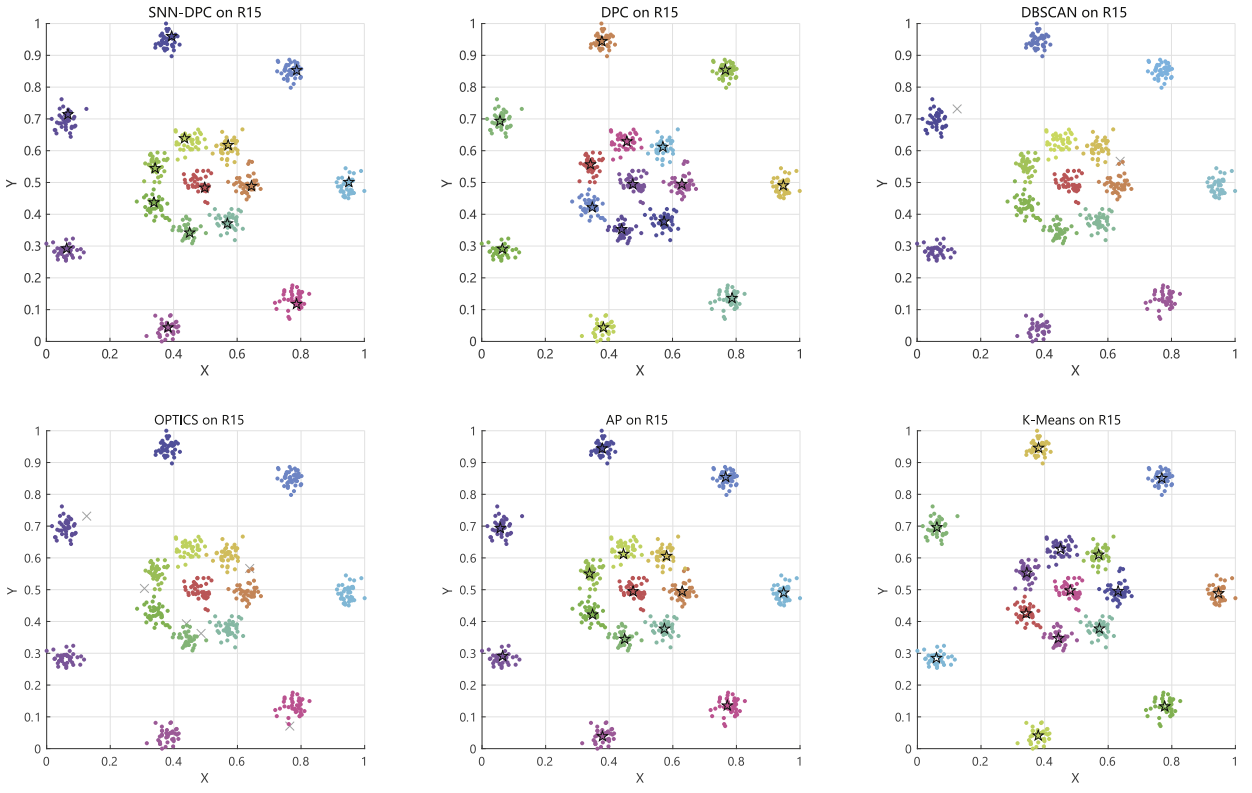


Fig. 12. The clustering results on R15 by 6 algorithms.

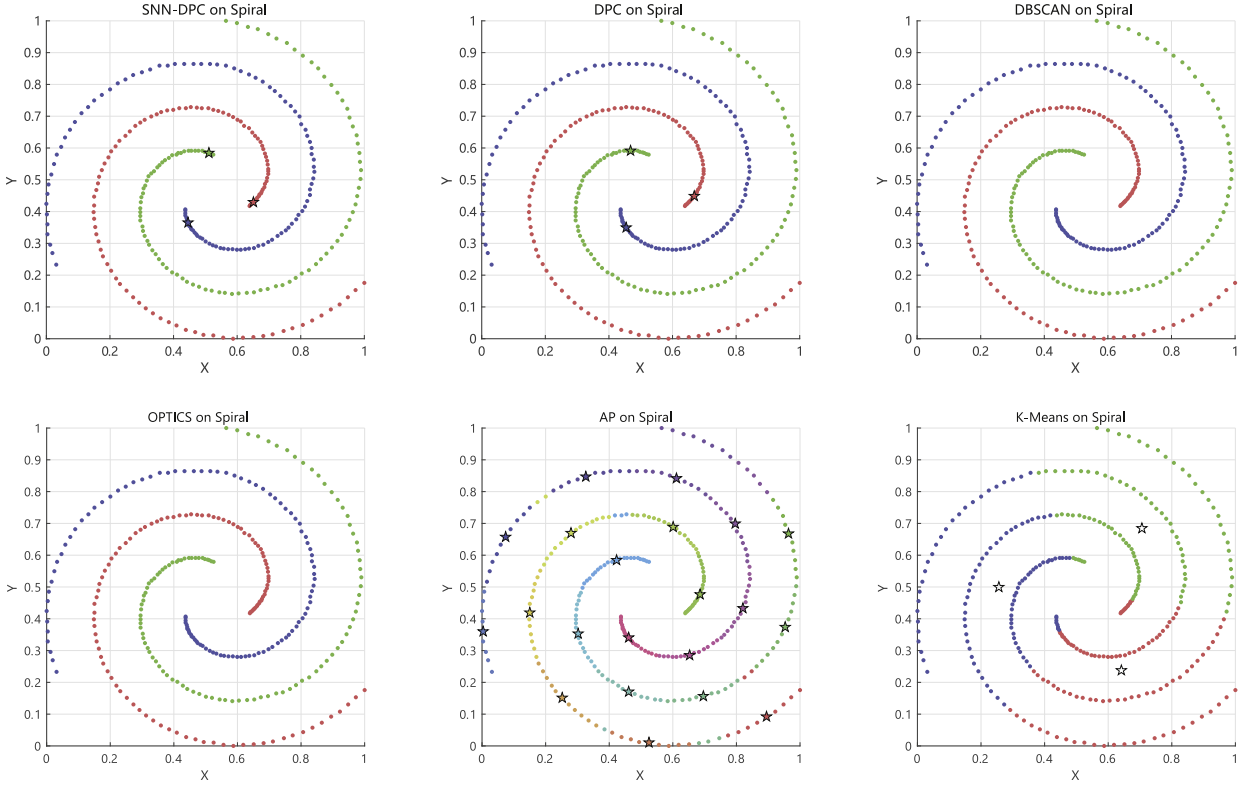


Fig. 13. The clustering results on spiral by 6 algorithms.

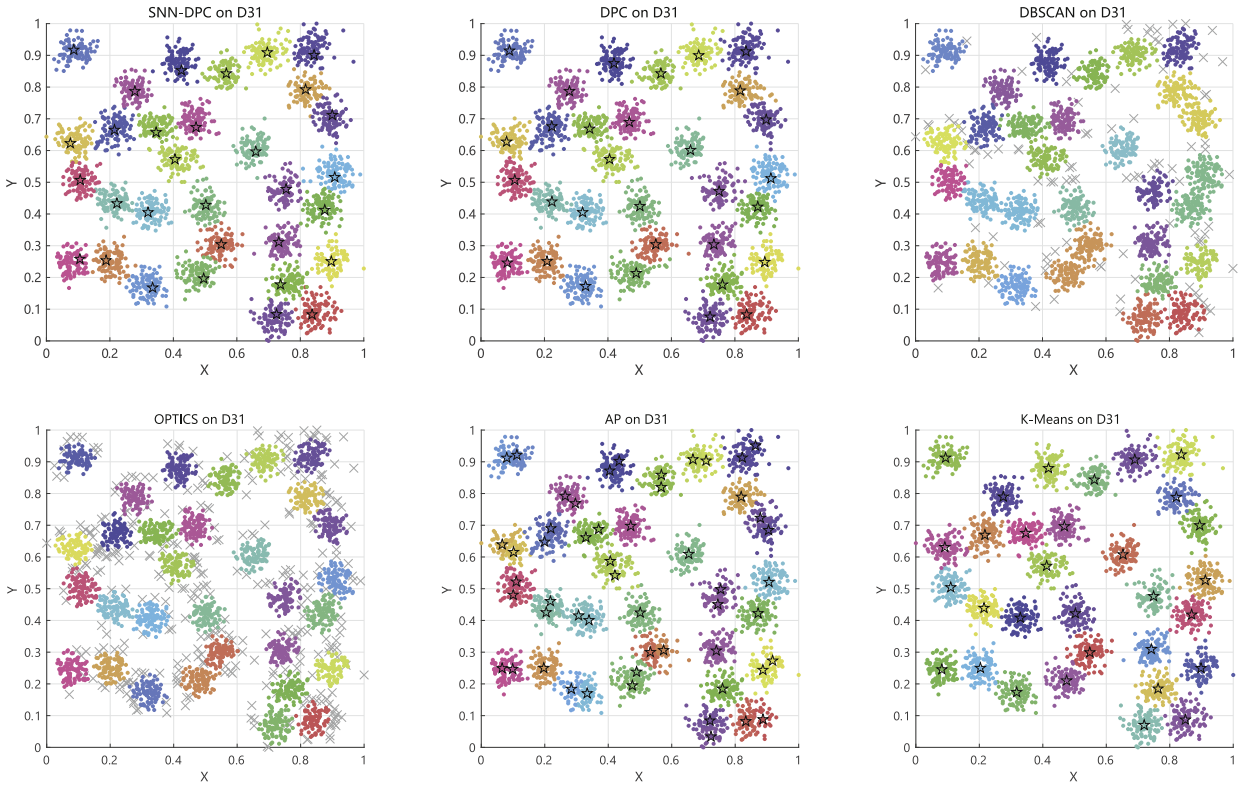


Fig. 14. The clustering results on D31 by 6 algorithms.

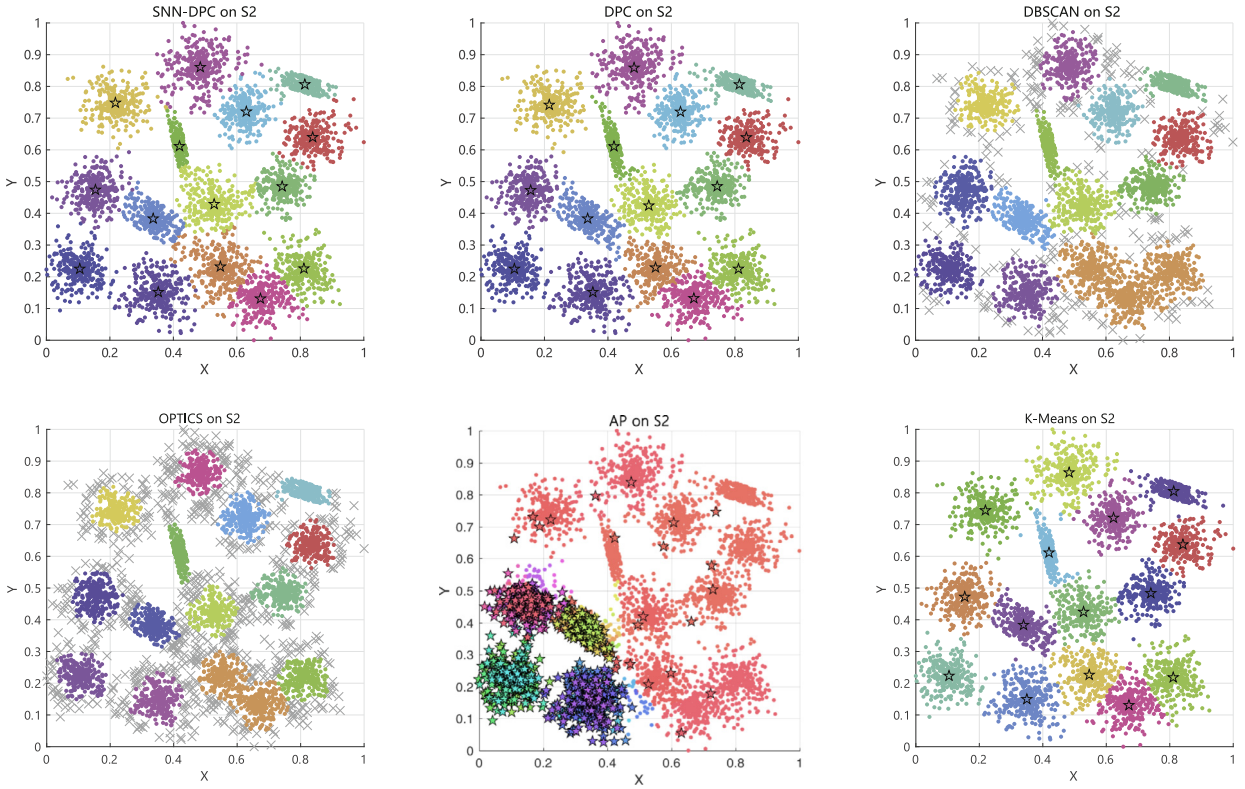


Fig. 15. The clustering results on S2 by 6 algorithms.

Table 5
Performances of different clustering algorithms on different real-world datasets.

Algorithm	AMI	ARI	FMI	Arg-	AMI	ARI	FMI	Arg-
	Iris				Waveform			
SNN-DPC	**0.9124**	**0.9222**	**0.9479**	15	**0.3984**	**0.4176**	**0.6164**	7
FKNN-DPC	0.8831	0.9038	0.9355	22	0.0774	0.0086	0.5050	6
DPC	0.8606	0.8857	0.9233	0.2	0.3261	0.2698	0.5292	0.1
DBSCAN	0.5692	0.6120	0.7291	0.12/5	0.0856	0.0097	0.4813	0.38/5
OPTICS	0.4513	0.6886	0.7868	0.15/5	0.0286	0.0918	0.2661	0.47/48
AP	0.5479	0.5701	0.7099	−0.57	0.2891	0.3014	0.5178	−2.20
K-Means	0.7331	0.7163	0.8112	3	0.3630	0.2536	0.5037	3
	Wine				Waveform(noise)			
SNN-DPC	**0.8735**	**0.8992**	**0.9330**	18	0.3259	**0.3108**	**0.6049**	10
FKNN-DPC	0.8038	0.7990	0.8667	9	0.0711	0.0122	0.5025	6
DPC	0.7065	0.6724	0.7835	2.0	0.0896	0.0695	0.4580	2.1
DBSCAN	0.5484	0.5292	0.7121	0.50/21	0.0000	0.0000	0.5773	0.02/2
OPTICS	0.3698	0.4119	0.6296	0.59/7	–	–	–	–
AP	0.3330	0.3170	0.6126	-2.02	0.0796	0.1336	0.2467	-2.43
K-Means	0.8473	0.8685	0.9126	3	**0.3645**	0.2519	0.5023	3
	WDBC				Ecoli			
SNN-DPC	**0.7520**	**0.8503**	**0.9305**	12	**0.6711**	**0.7547**	**0.8243**	6
FKNN-DPC	0.3560	0.4009	0.7658	9	0.4755	0.5535	0.6919	9
DPC	0.0007	−0.0028	0.7257	1.3	0.4978	0.4465	0.5775	0.4
DBSCAN	0.3581	0.4786	0.7570	0.46/38	0.4516	0.5255	0.6623	0.20/22
OPTICS	0.0856	0.4305	0.6767	0.51/65	0.4260	0.6642	0.7515	0.23/29
AP	0.5936	0.1322	0.7879	2.62	0.5339	0.4907	0.6134	−0.86
K-Means	0.6110	0.7302	0.8770	2	0.5051	0.4190	0.5542	8
	Seeds				Dermatology			
SNN-DPC	**0.7509**	**0.7890**	**0.8589**	6	**0.8749**	**0.8689**	**0.9021**	19
FKNN-DPC	0.6971	0.7422	0.8276	9	0.8355	0.8127	0.8504	35
DPC	0.7299	0.7670	0.8444	0.7	0.7840	0.7760	0.8221	1.6
DBSCAN	0.5302	0.5291	0.6711	0.24/16	0.5721	0.4165	0.5395	0.99/3
OPTICS	0.3802	0.4190	0.6350	0.81/5	0.2934	0.3430	0.4563	0.99/1
AP	0.4465	0.3936	0.6933	−2.07	0.6898	0.5935	0.6766	−0.84
K-Means	0.6705	0.7049	0.8026	3	0.8748	0.7426	0.7947	6
	Segmentation				Parkinsons			
SNN-DPC	0.6725	0.5770	0.6457	7	0.1529	**0.2916**	**0.8032**	5
FKNN-DPC	0.5830	0.4367	0.5581	27	0.0728	0.1601	0.6582	7
DPC	**0.6927**	**0.6004**	**0.6730**	1.5	**0.2478**	0.1256	0.6187	1.2
DBSCAN	0.4965	0.4543	0.5277	0.15/2	0.0071	0.0252	0.5775	0.50/17
OPTICS	0.4312	0.4600	0.5361	0.15/1	0.0368	0.0986	0.5049	0.45/9
AP	0.2089	0.3445	0.3409	1.80	0.1098	0.0343	0.2246	0.23
K-Means	0.6102	0.5049	0.5758	7	0.2129	0.0520	0.5957	2
	Libras Movement				Balance Scale			
SNN-DPC	**0.5834**	**0.3927**	**0.4507**	11	**0.4082**	**0.5716**	**0.7731**	20
FKNN-DPC	0.4754	0.3184	0.3976	11	0.0351	0.0236	0.5548	9
DPC	0.5358	0.3193	0.3717	0.3	0.1154	0.1394	0.5024	1.1
DBSCAN	0.4183	0.1965	0.2570	0.90/2	0.0902	0.1394	0.1510	0.03/1
OPTICS	0.1377	0.0828	0.2126	0.59/1	0.0633	0.1062	0.1165	0.03/2
AP	0.1487	0.2056	0.1971	4.31	0.0902	0.1420	0.1553	0.97
K-Means	0.5232	0.3094	0.3612	15	0.0132	0.0015	0.0444	3
	Ionosphere				Spectrometer			
SNN-DPC	0.3644	0.4949	0.7798	5	**0.4033**	**0.2377**	**0.2879**	12
KNN-DPC	0.1314	0.1321	0.5841	26	0.3052	0.1589	0.2110	23
DPC	0.1504	0.2357	0.6491	0.5	0.3226	0.2097	0.2601	1.1
DBSCAN	**0.5947**	**0.7226**	**0.8740**	0.78/9	0.0902	0.1394	0.1510	0.03/1
OPTICS	0.0970	0.3383	0.6085	0.58/1	0.0902	0.1370	0.1469	0.03/1
AP	0.1367	0.0773	0.5137	1.92	0.0902	0.1420	0.1553	0.97
K-Means	0.1294	0.1776	0.6053	2	0.0388	0.0236	0.0708	48

5.3. Read-world datasets

In this section, 14 UCI datasets are selected to demonstrate the performance of the SNN-DPC clustering algorithm. These datasets are different in terms of scale, feature number and cluster number. Table 3 provides a summary of each UCI dataset. We fail to obtain the results with the OPTICS algorithm on the dataset Waveform(noise), possibly due to its large scale; thus, there are only four '-'.

As shown in Table 5, SNN-DPC outperforms the other algorithms in most test cases.

Table 6
Performances of clustering algorithms on the olivetti faces dataset.

Algorithm	AMI	ARI	FMI	No clusters	Arg-
SNN-DPC	0.82440	**0.72012**	**0.72844**	40	6
	0.82005	0.69194	0.69976	52	6
FKNN-DPC	0.64892	0.48317	0.50671	40	5
	0.65540	0.47573	0.50475	45	5
	0.57461	0.39485	0.43487	33	5
DPC	**0.82592**	0.68628	0.69926	40	0.4
	0.76826	0.61803	0.64378	34	0.4
DBSCAN	0.81168	0.64552	0.66	54	2.00/2
OPTICS	0.42857	0.50361	0.58452	40	0.59/2
AP	0.72969	0.62603	0.64	54	−0.347
K-Means	0.79436	0.65730	0.67	40	40

5.4. Olivetti faces dataset

The Olivetti Faces dataset [29], which includes face images of 40 people, each with 10 different angles of images, is a widely used database in machine learning fields. Since the number of people (number of clusters) is greater than the number of faces per person (number of points within clusters), the density estimation for the DPC algorithm faces certain difficulties. However, SNN-DPC uses an improved density based on shared neighbors; thus, the impact is less obvious. To reduce the storage and computational complexities, we first scale each image (originally 92×112) to a smaller size of 15×15; then, we perform principal component analysis (PCA) to filter out attributes of cumulative contribution rates greater than 90%. Finally, we apply clustering algorithms on the data.

In this experiment, for the three density peak clustering algorithms, we test not only the performance of automatic decision making according to the known numbers of clusters but also the situations of manually making decisions according to the decision graph or the γ map. For the other three traditional algorithms, we only iteratively change their arguments and show the best result.

The results of all tested algorithms are shown in Table 6. As shown, the three metrics of the SNN-DPC algorithm are remarkably higher than those of the other algorithms, even when selecting centers manually.

Fig. 16a and16b show the results of the traditional DPC and SNN-DPC algorithms under the condition of choosing 40 centers automatically. In the figures, faces of the same color are divided into the same cluster. The white dots on the upper right corner of photos indicate the cluster center. Gray photos mean that the number of photos of this person in this cluster is less than 4. Such photos will not be marked as cluster centers, and thus, there may be some clusters without a center.

Fig. 16b shows the results for traditional DPC. It can be observed that at least 3 gray photos appear on the 1st, 5th, 17th, 18th, 38th, 39th, and 40th individuals, and the 15th, 28th, 29th, and 31st individuals are divided into two clusters. This phenomenon shows that even if the correct number of cluster centers is specified, the traditional DPC algorithm still has difficultly in locating the clusters. In addition, in the case of individuals who are only divided into one cluster, the centers of the 4th, 6th, 8th, 10th, 11th, 18th, and 35th individuals are not on any of their photos. This result suggests that the traditional DPC algorithm may also incorrectly merge multiple clusters into one cluster.

Fig. 16a shows the results for SNN-DPC. As shown, the 17th, 20th and 27th individuals have at least 3 gray photos. The 1th, 15th, 28th, 31st, 32th, 35th, 37th, and 40th individuals are divided into two clusters. This result indicates that compared to the traditional DPC algorithm, SNN-DPC has more chances to detect a cluster. Moreover, only the centers of the 33th, 30th, and 36th individuals are not on any of their photos, which is reduced by 4 when compared with traditional DPC, also demonstrating that SNN-DPC is less likely to incorrectly merge clusters.

6. Discussion

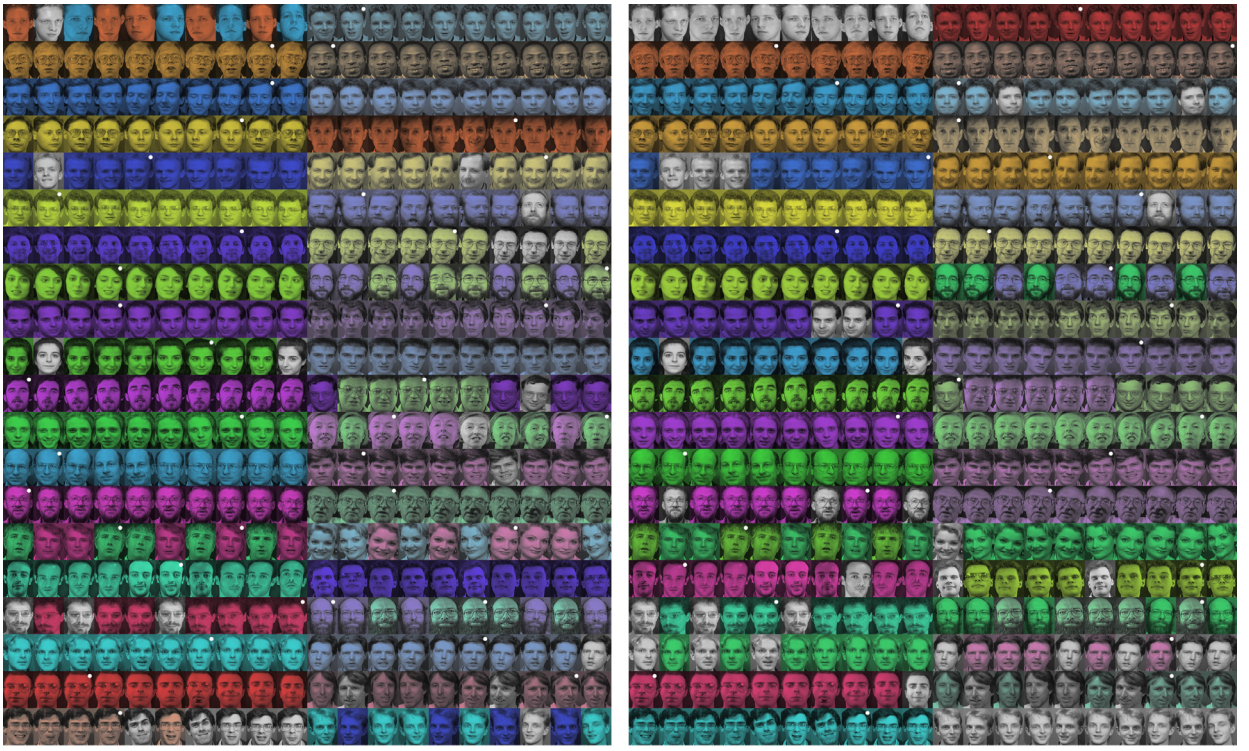
In this section, we will discuss the performance of SNN-DPC from several aspects. In detail, we will focus on the influences of the argument and the order of cases as well as the running time.

6.1. Argument

In this part, we will discuss the influence of the argument in the SNN-DPC algorithm.

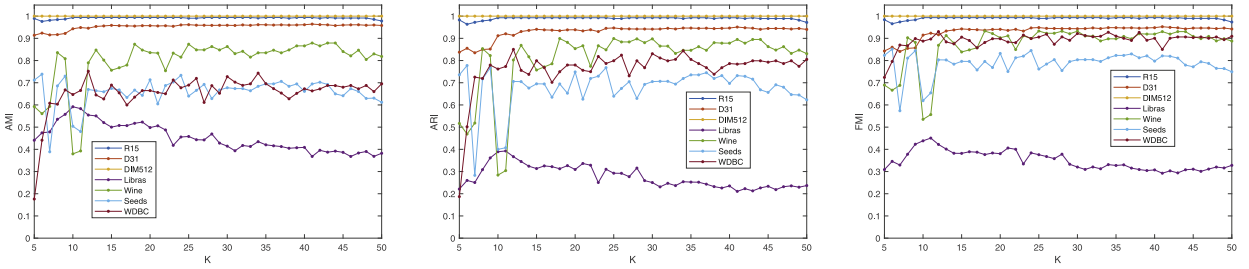
In the SNN-DPC algorithm, k is a significant argument; we use k to determine how many neighbors of a point will be considered during the process, and it affects many crucial steps in the algorithm procedure. In other words, the value of k directly determines the performance of the SNN-DPC algorithm. Thus, it deserves a discussion.

Fig. 17 shows the AMI, ARI and FMI metric values of some representative datasets with different k argument values. The k value ranges from 5 to 50, which is a reasonable range for all the datasets since a very small k results in all parts having very few shared neighbors, and an excessive k results in all the points having many neighbors, both of which will make the SNN strategy less meaningful and will negatively affect its performance.



(a) SNN-DPC algorithm (b) Traditional DPC algorithm

Fig. 16. The clustering results on olivetti faces by SNN-DPC and traditional DPC.



(a) AMI (b) ARI (c) FMI

Fig. 17. Results on different datasets with different k arguments.

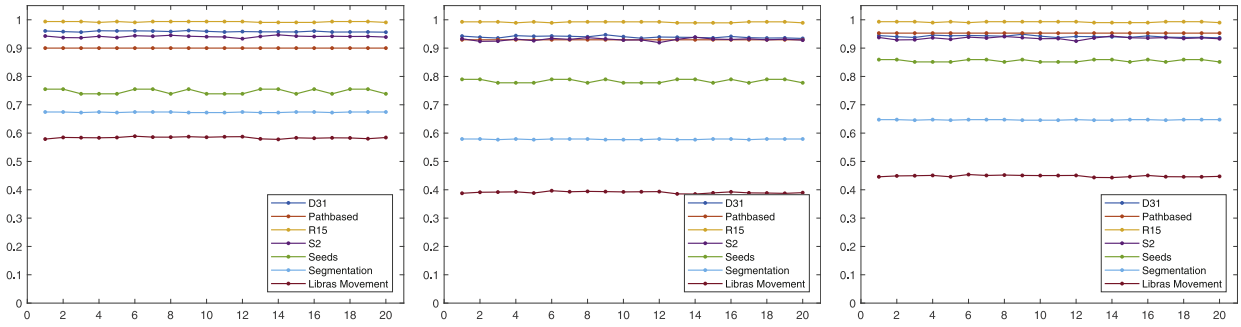
As shown in Fig. 17, the general trend for all datasets is that a larger k results in more stable metric values. Most datasets, regardless of whether they are synthetic or real-world datasets, fluctuate severely before $k = 13$; subsequently, the fluctuation becomes less conspicuous. This is particularly true for the FMI metric value.

Note that for the three large-scale synthetic datasets, there is almost no fluctuation, and the same metric values are kept from $k = 5$ to $k = 50$, which partially indicates that, for the large-scale dataset, the performance of the SNN-DPC algorithm is robust to changes in k. In terms of the dimension, from the highest dimensional dataset DIM512 and the second-highest dimensional dataset Libras Movement, we can observe that the result is more stable on the high-dimensional datasets.

Meanwhile, the datasets that are of both small scale and low dimension, such as the Wine and Seeds datasets, tend to be unstable when k is relatively small, but they eventually return to a normal level with increasing k. Both datasets demonstrate the robustness of the SNN-DPC algorithm.

6.2. Order sensitivity

The order sensitivity considers whether, for a single clustering algorithm, the clustering result will be significantly affected by the order of cases in the datasets. In general, unless there are specific reasons, an ideal clustering algorithm should not be order sensitive.



(a) AMI (b) ARI (c) FMI

Fig. 18. Results of different datasets in random order.

Table 7
Running time of 3 density peak clustering algorithms (Unit: second).

Name	SNN-DPC	FKNN-DPC	DPC	Name	SNN-DPC	FKNN-DPC	DPC
Aggregation	0.6222	0.1597	0.0697	Seeds	0.0559	0.0219	0.0069
Flame	0.0611	0.0206	0.0084	Segmentation	4.7392	0.7737	0.7349
Jain	0.1554	0.0430	0.0174	Libras Movement	0.1502	0.0481	0.0211
Pathbased	0.1083	0.0391	0.0131	Ionosphere	0.1062	0.0447	0.0141
R15	0.3697	0.0809	0.0456	Waveform	24.0300	3.0256	3.5444
Spiral	0.1053	0.0264	0.0116	Waveform(noise)	23.8030	3.0709	3.7184
D31	9.5172	1.1664	1.3218	Ecoli	0.1601	0.0363	0.0146
DIM512	0.9410	0.1888	0.2268	Dermatology	0.1764	0.0442	0.0204
S2	26.4790	3.2968	3.3741	Parkinsons	0.1752	0.0498	0.0137
Iris	0.0414	0.0181	0.0072	Balance Scale	0.4445	0.0455	0.0545
Wine	0.0534	0.0191	0.0055	Spectrometer	0.3364	0.1737	0.0516
WDBC	0.3527	0.0945	0.0437				

In this part, we will discuss the order sensitivity of the SNN-DPC algorithm. We choose several representative datasets from Tables 2 and3, and we randomize the order of cases in one of the datasets. Then, according to the argument information shown in Tables 4 and5, we apply SNN-DPC to the randomized dataset with the best argument. After obtaining the result, we randomize it and apply SNN-DPC to it again. For each dataset, we repeat this process 20 times before proceeding to next dataset. If there are significant differences among the results of repeated experiments, we believe that the algorithm is order sensitive; otherwise, we believe that the algorithm is not order sensitive.

Fig. 18 shows the results of all experiments. Each line represents a single dataset, and each point on the line is one experiment. Apart from the difference in values, it is clear that, for the same dataset, the metric values are stable, without an abrupt change or severe fluctuations. In this regard, it is convincing that the SNN-DPC algorithm is not order sensitive.

6.3. Running time

In this part, we will compare the running time of our SNN-DPC algorithm with those of the FKNN-DPC and traditional DPC algorithms on all the datasets used in the experiments. From Section 4.3.1, we have learned that the time complexity of traditional DPC is $O(n^2)$, and the time complexity of our method is $O((k+m)n^2)$, where n is the number of cases in the dataset and k represents how many neighbors will be considered in the SNN-DPC algorithm. The main time consumption is during the process of obtaining the shared neighbors, the essence of which is the intersection of two sets. Another process that is also time consuming is the allocation of possible subordinate points whenever the algorithm cannot assign any point to an existing cluster, i.e., when it reaches line 14, it will add 1 to k and repeat the previous steps. This also consumes extra time, but it allows the algorithm to be more accurate. To compensate for the large time consumption, we apply some optimizations to the algorithm. Therefore, even though the general time complexity of the SNN-DPC is larger than that of the traditional DPC, the actual execution times on datasets are not as slow as expected.

We perform experiments on a computer with an Intel Core i5-6300HQ 2.30 GHz CPU and 12.0 GB of RAM running MATLAB 2017a (for SNN-DPC, FKNN-DPC, and DPC) and Python 3.6.2 (for the other algorithms). Table 7 shows the average running time of each algorithm on the datasets provided in Tables 2 and3. Since only the SNN-DPC, FKNN-DPC and traditional DPC algorithms are implemented in MATLAB, the running times of the other algorithms are omitted from the table.

To make the results less sensitive to unexpected incidents, for each dataset, we apply its best argument and perform the same process 50 times. All values in Table 7 are the average running time. As shown, the running time of SNN-DPC is approximately seven or eight times that of DPC, but the values of k range between 5 and 30. It is clear that even though the argument k has some negative influence on the running time, this influence is a relatively constant multiplier to the running

time. Additionally, for the larger datasets, the differences are less than the average level; for example, on the "Segmentation" dataset, SNN-DPC's running time is 6.44 times that of traditional DPC, and on the "Waveform (noise)" dataset, it is 6.40 times that of traditional DPC.

7. Conclusion

In this paper, we improve the traditional DPC algorithm and propose a shared-nearest-neighbor-based clustering by fast search and find of density peaks, namely, SNN-DPC. SNN-DPC redefines the local density and distance from the nearest larger density point and introduces the shared neighbors and local density information between points; therefore, the newly defined ρ and δ can reflect the attributes of points more objectively and avoid the problem that the traditional DPC algorithm cannot effectively address variable-density clusters. In addition, two types of point allocation strategies based on shared neighbor information are proposed, the allocation of inevitable subordinate points and of possible subordinate points, which avoid the problem of joint allocation errors caused by the one-step allocation strategy in the traditional DPC algorithm.

The experimental results on classical synthetic datasets, UCI real-world datasets and the Olivetti Faces dataset show that the SNN-DPC algorithm can find cluster centers more accurately and is able to effectively work with a variety of forms and distributions of datasets. Moreover, the algorithm can assign non-center points to the appropriate cluster. SNN-DPC is an effective adaptive clustering algorithm, is applicable to any dimension and any size of dataset, and is robust to noise and differences in cluster density.

For future work, we divide the work into two aspects. For the theoretical aspect, the first part is to continue to explore the clustering algorithm based on shared neighbors, find a way to automatically determine the value of k, and simplify the process of determining the arguments of the algorithm. The second part is continuing to improve the definition of ρ and δ, making it easier to manually find the centers of clusters. The third part is to combine SNN-DPC with other algorithms to play on the advantages of other algorithms and to compensate for SNN-DPC in specific problems. At the application level, we will attempt to apply the SNN-DPC algorithm to production environments, solve practical problems, and promote production efficiency in related fields.

Acknowledgment

This work is partly funded by the National Natural Science Foundation of China (Nos. 61672329, 61373149, 61472233, 61572300, and 81273704), Shandong Provincial Project for Science and Technology Development (2014GGX101026), Shandong Provincial Project of Education Scientific Plan (No. ZK1437B010), Taishan Scholar Program of Shandong Province (Nos. TSHW201502038 and 20110819), and Shandong Provincial Project of Exquisite course (Nos. 2012BK294, 2013BK399, and 2013BK402).

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, 27, ACM, 1998.
[2] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: Proceedings of the ACM Sigmod Record, 28, ACM, 1999, pp. 49–60.
[3] D. Dua, E. Karra Taniskidou, UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2017 [http://archive.ics.uci.edu/ml].
[4] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.
[5] H. Chang, D.-Y. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (1) (2008) 191–203.
[6] M. Charytanowicz, J. Niewczas, P. Kulczycki, P.A. Kowalski, S. ?ukasik, S. ?ak, Complete Gradient Clustering Algorithm for Features Analysis of X-ray Images, Springer, pp. 15–24.
[7] Y.-W. Chen, D.-H. Lai, H. Qi, J.-L. Wang, J.-X. Du, A new method to estimate ages of facial image for large database, Multimed. Tools Appl. 75 (5) (2016) 2877–2895.
[8] A. Dempster, N. Laird, D. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, J. R. Stat. Soc. Ser. B Methodol. 39 (1) (1977) 1–38.
[9] D.B. Dias, R.C. Madeo, T. Rocha, H.H. Bķscaro, S.M. Peres, Hand movement recognition for brazilian sign language: a study using distance-based neural networks, in: Proceedings of the International Joint Conference on Neural Networks, IEEE, 2009, pp. 697–704.
[10] M. Du, S. Ding, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, Knowl. Based Syst. 99 (2016) 135–145.
[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the KDD, 96, 1996, pp. 226–231.
[12] E.B. Fowlkes, C.L. Mallows, A method for comparing two hierarchical clusterings, J. Am. Stat. Assoc. 78 (383) (1983) 553–569.
[13] P. Franti, O. Virmajoki, V. Hautamaki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Trans. Pattern Anal. Mach. Intell. 28 (11) (2006) 1875–1881.
[14] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.
[15] P. Frnti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recognit. 39 (5) (2006) 761–775.
[16] L. Fu, E. Medico, Flame, a novel fuzzy clustering method for the analysis of dna microarray data, BMC Bioinform. 8 (1) (2007) 3.
[17] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Discov. Data 1 (1) (2007) 4.
[18] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, in: Proceedings of the ACM Sigmod Record, 27, ACM, 1998, pp. 73–84.
[19] L. Huang, Y. Li, G. Wang, Y. Wang, Community detection method based on vertex distance and clustering of density peaks, J. Jilin Univ. Eng. Technol. Ed. 46 (6) (2016) 2042–2051.
[20] A.K. Jain, M.H. Law, Data clustering: a user's dilemma, PReMI 3776 (2005) 1–10.
[21] R.A. Jarvis, E.A. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Trans. Comput. 100 (11) (1973) 1025–1034.
[22] Y. Ju, Research on manifold-based density peaks clustering algorithm, (Thesis), 2016.

[23] L. Kaufman, P. Rousseeuw, Clustering by Means of Medoids, North-Holland, 1987.
[24] T. Li, H. Ge, S. Su, Density peaks clustering by automatic determination of cluster centers, J. Front. Comput. Sci. Technol. 10 (11) (2016) 1614–1622.
[25] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, Oakland, CA, USA., 1967, pp. 281–297.
[26] R. Mehmood, G. Zhang, R. Bie, H. Dawood, H. Ahmad, Clustering by fast search and find of density peaks via heat diffusion, Neurocomputing 208 (2016) 210–217.
[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (Oct) (2011) 2825–2830.
[28] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496, doi:10.1126/science.1242072.
[29] F.S. Samaria, A.C. Harter, Parameterisation of a stochastic model for human face identification, in: Proceedings of the Second IEEE Workshop on Applications of Computer Vision, IEEE, 1994, pp. 138–142.
[30] G. Sheikholeslami, S. Chatterjee, A. Zhang, Wavecluster: a wavelet-based clustering approach for spatial data in very large databases, VLDB J. Int. J. Very Large Data Bases 8 (3–4) (2000) 289–304.
[31] X. Shi, G. Feng, M. Li, Y. Li, C. Wu, Overlapping community detection method based on density peaks, J. Jilin Univ. Eng. Technol. Ed. 47 (1) (2017) 242–248.
[32] 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 Comput. Appl. (2016) 1–11.
[33] V.G. Sigillito, S.P. Wing, L.V. Hutton, K.B. Baker, Classification of radar returns from the ionosphere using neural networks, Johns Hopkins APL Tech. Dig. 10 (3) (1989) 262–266.
[34] D.B. Goldgof, Nuclear feature extraction for breast tumor diagnosis, Proc Spie 1993 (1992) 861–870.
[35] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (Dec) (2002) 583–617.
[36] P. Tan, Introduction to Data Mining, Pearson Education India, 2006.
[37] C.J. Veenman, M.J.T. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1273–1280.
[38] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance, J. Mach. Learn. Res. 11 (Oct) (2010) 2837–2854.
[39] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416.
[40] S. Wang, D. Wang, C. Li, Y. Li, G. Ding, Clustering by fast search and find of density peaks with data field, Chin. J. Electron. 25 (3) (2016) 397–402.
[41] W. Wang, J. Yang, R. Muntz, Sting: a statistical information grid approach to spatial data mining, in: Proceedings of the VLDB, 97, 1997, pp. 186–195.
[42] J. Xie, H. Gao, W. Xie, X. Liu, P.W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors, Inf. Sci. (Ny) 354 (2016) 19–40.
[43] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 478–487.
[44] J. Xu, G. Wang, W. Deng, Denpehc: density peak based efficient hierarchical clustering, Inf. Sci. (Ny) 373 (2016) 200–218.
[45] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: Proceedings of the ACM Sigmod Record, 25, ACM, 1996, pp. 103–114.
[46] Y. Zhang, Y. Xia, Y. Liu, W. Wang, Clustering sentences with density peaks for multi-document summarization, in: Proceedings of the HLT-NAACL, 2015, pp. 1262–1267.
[47] J. Zhong, W.T. Peter, Y. Wei, An intelligent and improved density and distance-based clustering approach for industrial survey data classification, Expert Syst. Appl. 68 (2017) 21–28.