IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 22, 2019, accepted October 15, 2019, date of publication October 21, 2019, date of current version October 30, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2948422*

# Improved Density Peaks Clustering Based on Shared-Neighbors of Local Cores for Manifold Data Sets

**DONGDONG CHENG**[1], **JINLONG HUANG**[1], **SULAN ZHANG**[1], **AND HUIJUN LIU**[2]
[1]College of Big Data and Intelligent Engineering, Yangtze Normal University, Chongqing 408100, China
[2]College of Computer Science, Chongqing University, Chongqing 400044, China

Corresponding author: Huijun Liu (lhjlcr@cqu.edu.cn)

This work was supported in part by the Project of Chongqing Natural Science Foundation under Grant cstc2019jcyj-msxmX0683, in part by the National Natural Science Foundation of China under Grant 61702060 and Grant 61601060, and in part by the Project of Chongqing Education Commission under Grant KJZH17104.

**ABSTRACT** A novel clustering algorithm by fast search and find of density peaks (DP) was proposed in Science, 2014. It has attracted much attention from researchers. It can easily select clusters centers with decision graph. However, it cannot be used to cluster manifold data sets as the existing distance measurement is not suitable to evaluate the dissimilarity between objects on manifold structure. Some researchers use graph-based distance to measure the dissimilarity between objects on manifold clusters, but computing the graph-based distance on the original data set is time consuming. An improved density peaks clustering algorithm based on shared-neighbors between local cores, SLORE-DP, is proposed in this paper. First, it finds local cores to represent the data set and redefines the graph-based distance between local cores with shared-neighbors-based distance. Then natural neighbor-based density and the new defined graph-based distance are used to construct decision graph on local cores and DP algorithm is employed to cluster local cores. Finally, the remaining points are assigned to the same cluster as their local cores belong to. Since we use the new defined graph-based distance to estimate the dissimilarity between local cores, SLORE-DP can be used to cluster manifold data sets and at the same time it only calculates the shortest path between local cores, which greatly reduces the running time of the algorithm. We do experiments on several synthetic data sets containing manifold clusters and several real data sets from UCI. The results show that SLORE-DP is more effective and efficient than other algorithms when clustering manifold data sets.

**INDEX TERMS** Shared-neighbors, local cores, density peaks, clustering.

## I. INTRODUCTION

As an unsupervised learning, clustering is an important method for data analysis. It has been widely used in the field of pattern recognition, image processing, and information retrieval. It is designed to divide objects into multiple clusters, so that similar objects are in the same cluster while different objects are in different clusters.

Many clustering algorithms have been proposed over the past few decades. According to different strategies, these algorithms can be roughly grouped into partitioning methods, density-based methods, hierarchical methods, model-based methods and grid-based methods. Among them, partitioning, density-based and hierarchical algorithms, due to their simple principle, are the most popular.

K-means [1] and K-medoids [2] are typical partitioning algorithms. However, their performance depends on the selection of initial cluster centers. To avoid selecting cluster centers, AP algorithm [3] treats all objects as potential centers. K-AP [4] is an improved AP algorithm. It uses the immediate result of K clusters by introducing a constraint in the process of message passing. However, since each point is always allocated to the nearest center, these algorithms cannot discover arbitrary-shaped clusters.

DBSCAN [5] is a typical density-based clustering algorithm. It defines clusters as dense regions separated by sparse regions. Dcore [6] is a hybrid decentralized approach which is based on finding density cores instead of centroids.

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

Its main idea is to find objects with higher density than their surroundings and these loosely connected objects compose density cores that roughly retains the shapes of clusters. RECOME [7] employs relative k-nearest neighbor kernel density (RNKD) to cluster. RNKD is used to determin core objects. non-core objects are partitioned into atom clusters by successively following higher density neighbor relations toward core objects. Core objects and their corresponding atom clusters are then merged through $\alpha$-reachable paths on k-nearest neighbor graph. DBSCAN, Dcore and RECOME can discover arbitrary-shaped clusters, but they have to set parameters without prior knowledge.

In 2014, Rodriguez and Laio reported a clustering algorithm by fast search and find of density peaks (DP for short) [10] in Science, which is based on the assumption that cluster centers tend to have a higher density than their neighbors and a relatively large distance from points with higher densities. It is able to quickly and effectively identify cluster centers by projecting the original data set into decision graph. However, there are some disadvantages in DP algorithm. First, it has to set cutoff distance to calculate densities of each point. Second, the decision graph based on Euclidean distance cannot properly reflect the relationship between objects on manifold. Third, there is a Domino Effect with the assignment strategy for the remaining points, because once one point is wrongly assigned, there will be more points assigned incorrectly, which makes it unable to discover clusters with complex structures. SNN-DPC [11] is proposed to solve these problems, but it has to take a lot of time to obtain the shared-neighbors of all data pairs.

In [8], the authors present a novel hybrid hierarchical clustering based on local cores, HCLORE. In the partition step, it divides the data set into several clusters by finding local cores. After that, it temporarily removes points with lower local density, so that the boundary between clusters becomes clearer. In the merging process, a newly defined similarities between clusters ensures the most similar clusters are merged. In [9], a novel minimum spanning tree clustering algorithm with local density peaks (LDP-MST) is proposed. HCLORE and LDP-MST both do well in discovering clusters with complex structures, but they have to predefine the number of clusters.

To solve the above problems, a shared-neighbor of local cores-based clustering algorithm is proposed, called SLORE-DP. The main innovations of this method include: (1) it redefines the graph-based distance between local cores with their shared-neighbors, which greatly reduces the running time compared with calculating the distance between all objects in a data set; (2) it constructs decision graph only on local cores according to the new distance, and uses DP algorithm to cluster local cores instead of the whole data set, which enables the proposed method to identify manifold clusters. The main steps of SLORE-DP is: first, we introduce natural neighbor to calculate the density of each object and obtain local cores with local maximum density than their neighbors, which does not need to set parameters; after that, we exploit the shared-neighbors of local cores to redefine the graph-based distance between local cores; then, we use the new density and the redefined distance to construct decision graph on local cores and cluster local cores by employing DP algorithm; finally, the non-local cores are assigned to the clusters their local cores belong to. The experiments by comparing our method with DP and SNN-DPC algorithms on synthetic data sets and real data sets show that our algorithm can discover manifold clusters faster, better and without any parameters.

The remaining content is organized as follows. Section II reviews the related works about DP algorithm and improved DP algorithms. Section III introduces natural neighbor and natural neighbor-based local density. Section IV presents the proposed clustering algorithm SLORE-DP and Section V shows the experimental results and analysis. Finally, we make a conclusion in Section VI.

## II. RELATED WORKS

### A. DP ALGORITHM

DP algorithm [10] assumes that cluster centers are characterized by a local maximum density and a relatively large distance from any points with a higher density. For each point $i$, the authors define its local density $\rho(i)$ and distance $\delta(i)$ from points of higher density. For each point $i$, the local density is computed with cutoff distance method shown in Eq. 1 or kernel distance method shown in Eq. 2:

$$\rho(i) = \sum_{j} \chi(d(i,j) - d_c) \tag{1}$$

$$\rho(i) = \sum_{i \neq j} \exp(-(\frac{d(i,j)}{d_c})^2) \tag{2}$$

where $\chi(x) = 1$ if $x < 0$ and $\chi(x) = 0$ otherwise, $d(i,j)$ is the Euclidean distance between points $i$ and $j$ and $d_c$ is a cutoff distance. Eq. 1 means that the local density of point $i$ equals to the number of points whose distances to $i$ are closer than $d_c$. The $\delta$ distance is defined as follows:

$$\delta(i) = \min_{j:\rho_j > \rho_i} (d(i,j)) \tag{3}$$

For the point $i$ with the maximum density, its $\delta(i)$ is defined as follows:

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

According to the definition of $\delta$ distance, only points with local or global maximum density have much larger $\delta(i)$.

DP algorithm computes the the local density $\rho$ and $\delta$ distance for each point and constructs the decision graph to map the data set into a two-dimensional graph w. r. t. the local density $\rho$ and $\delta$ distance. In decision graph, cluster centers stand out with anomalously large value of $\rho$ and $\delta$.

After finding cluster centers, we assign each remaining point to the same cluster which its nearest neighbor of higher density belong to. We do not consider finding noise points here. The DP algorithm is described in Algorithm 1.

When giving the density of each point and the distance matrix, the time complexity of DP is $O(N)$, where $N$ is the number of points in a data set.

**Algorithm 1** DP

```
Input: $\rho$: the density, $d$: the distance matrix
Output: $CL$: the cluster label of each point
$N = \text{size}(\rho)$;
$(sorted\rho, Index\rho) = \text{sort}(\rho, \text{'descend'})$;
for $i = 2 : N$ do
    $x = Index\rho(i)$;
    $p = \arg\min_{o \in Index\rho(1:i-1)} (d(x, o))$;
    $\delta(x) = d(x, p)$;
    $NNeighbor(x) = p$;
end
$\delta(Index\rho(1)) = \max(\delta)$;
Construct decision graph;
Determine $\rho_{min}$ and $\delta_{min}$ according to the decision graph by manually selection;
$Ncluster = 1$;
for each point $p$ do
    if $\rho(p) > \rho_{min}$ and $\delta(p) > \delta_{min}$ then
        $CL(p) = Ncluster$;
        $Ncluster = Ncluster + 1$;
    end
end
for each point $p$ do
    $CL(p) = CL(NNeighbor(p))$;
end
Return $CL$;
```

### B. IMPROVED DP ALGORITHMS

Decision graph gives us a better chance to understand the data, so that DP algorithm can choose the preferred clustering result quickly. However, it has to set cutoff distance to compute the density and the assignment strategy cannot process clusters with complex manifold structures.

In order to improve the density measure, some improved DP algorithms employ k-nearest neighbors to solve the problem. FKNN-DPC [12] is a robust clustering method by finding density peaks. It computes the local density according to k-nearest neighbors and the rest of the points are assigned to the most probable clusters in two steps. In the first step, it starts from cluster centers and breadth-first searches the k-nearest neighbors of a point to assign the non-outliers. In the second step, outliers and the points unassigned by the technique of fuzzy weighted k-nearest neighbors are clustered according to the result of the first step. In [13], the authors propose a clustering algorithm based on k-nearest neighbors and principal component analysis (PCA), named DPC-KNN-PCA, in which, k-nearest neighbors is introduced to compute the local density of each point and PCA is used to process high-dimensional data sets. In [14], two improved algorithms are presented, which adopt different ways to utilize the dissimilarity based on Transitive closure and Shared Nearest Neighbors.

Some algorithms are proposed to optimize the choice of cluster centers and automatically determine the number of clusters. STclu [15] defines k-density to measure the local density of points and detects cluster centers automatically via outward statistical testing. DenPEHC [16] is a hierarchical clustering algorithm based on DP algorithm. It uses decision graph to generate clustering layers and obtains clusters on each possible clustering layer. It also introduces a grid granulation framework to enable DenPEHC to process large-scale and high-dimensional data sets. 3DC algorithm [17] automatically detects the number of clusters by employing a divide-and-conquer strategy and the definition of density-reachable in the DBSCAN framework. A novel method to fast determine cluster centers is proposed in [18], which proves that the singular points outside the confidence interval by setting the confidence interval are cluster centers through theory analysis and simulations. QCC [19] is also a hierarchical clustering algorithm. It first finds the quasi-cluster centers which correspond to initial clusters and the density of a quasi-cluster center is the highest among its k-nearest neighbors or reverse nearest neighbors. Then, it defines a new metric to evaluate the similarity between initial clusters and obtains the clustering results by continually merging the most similar initial clusters.

Some improved DP algorithms are proposed to process data sets containing complex structured clusters. SNN-DPC [11] defines SNN similarity. When computing the local density and $\delta$ distance, it considers the information of the nearest neighbor and the shared neighbor between different points. Then it introduces a two-step assignation way: inevitable subordinate and possible subordinate. Inevitable subordinate accurately and quickly identifies and allocates points certainly belong to a cluster through counting the number of shared neighbors between two points. Possible subordinate allocates the remaining points through finding the clusters to which more neighbors belong. NaNLORE [20] improves DP algorithm by introducing the definitions of local representatives and density-adaptive distance. Local representatives are used to represent the whole data set and the density-adaptive distance helps to measure the dissimilarity between local representatives. It constructs decision graph on local representatives according to the density-adaptive distance, which makes it applicable to cluster data sets with complex structures.

There are also some researchers apply DP algorithm to image processing [21], community detection [22]–[25], extracting multi-document abstracts [26] and noise removal [27].

## III. PRELIMINARIES

### A. NATURAL NEIGHBOR

Many algorithms employ k-nearest neighbors to evaluate the local density of points, but they have to set $k$ value. Natural neighbor [28] can adapt to the distribution of data sets and obtain the $k$ value automatically. The performance in [20], [28]–[30] demonstrates its effectiveness. It is inspired by the reality that the friendship between two objects should be mutual. If everyone has at least one friend or the number of persons who take him or her as friends does not change, we call it reaches natural stable state.

For a data set $D$, the Euclidean distance between point $p$ and $q$ is denoted as $d(p, q)$. We assume that $o$ is the $k$-th nearest neighbor of point $p$. Then, k-nearest neighbors and reverse k-nearest neighbors are defined as shown in Definition 1 and 2, respectively.

*Definition 1:* (k-nearest neighbors) The k-nearest neighbors of point $p$ are a set of points whose distance to $p$ are less than or equal to $d(p, o)$, that is, $NN_k(p) = \{x \in D|d(p, x) \leq d(p, o)\}$

*Definition 2:* (Reverse k-nearest neighbors) The reverse k-nearest neighbors of point $p$ are a set of points who consider $p$ as one of its k-nearest neighbors, that is, $RNN_k(p) = \{x \in D|p \in NN_k(x)\}$.

Natural neighbor information is obtained by the following steps: we initialize $r$ with 1 and add one every time; we search $r$-th-nearest neighbor and count the number of reverse neighbors $nb(p)$ for each point $p$ in each iteration; when the number of points who do not have reverse neighbors is constant, it terminates and $r$ at this moment is called natural characteristic value $\lambda$; finally, it returns $\lambda$ and $nb$-nearest neighbors of points. The detailed process is shown in Algorithm 2.

**Algorithm 2** NaN-Searching

```
Input: D: the data set
Output: $\lambda, NN_{nb}$
Initializing: $r = 1, nb(i) = 0, NN_0(i) = \phi, RNN_0(i) = \phi,$
$Numb(0) = N$;
while true do
    for each data point $p \in D$ do
        Use kdtree to find the $r$-th neighbor $q$ of $p$;
        $nb(q) = nb(q) + 1$;
        $NN_r(p) = NN_{r-1}(p) \cup \{q\}$;
        $RNN_r(q) = RNN_{r-1}(q) \cup \{p\}$;
    end
    Compute the number of points which do not have
    reverse neighbors (i.e., $nb(p) = 0$) $Numb(r)$;
    if $Numb(r) == Numb(r-1)$ then
        Break;
    end
    r=r+1;
end
$\lambda = r$;
Output the $\lambda, NN_{nb}$;
```

In Algorithm 2, KD-tree is introduced to search $r$-nearest neighbors, which will reduce the time complexity of Algorithm 2 to $O(N\log N)$, otherwise, its time complexity is $O(N^2 \log N)$. ($N$ is the number of objects in a data set). Intuitively, points in dense regions have more neighbors than that in sparse regions. $nb(x)$ is greater for points in dense regions than that in sparse regions according to Algorithm 2. It well reflects the local characteristic of points. It is reasonable to use $nb(x)$ to define the local neighbor of point $x$. Thus, we give the definition of local neighbor as follows.

*Definition 3:* (Local neighbor) Local neighbor of $p$ is the $nb(p)$-nearest neighbors, that is, $LN(p) = NN_{nb(p)}(p)$.

### B. NATURAL NEIGHBOR-BASED LOCAL DENSITY

The density of points is used to measure the intensity and sparsity of the space in which it is located. Points in dense regions obviously have larger density than that in sparse regions. The key is to quantify the density. We find that the sum of the distances between a point and its $k$ nearest neighbors is smaller in a dense region than that in a sparse region. STclu [15] defines k-density as the ratio of k and the sum of distances with its k-nearest neighbors. The authors have proven that k-density performs better in discovering cluster centers than that in [10]. Similar to k-density, we define natural neighbor-based local density as shown in Eq. 5:

$$Den(p) = \frac{\lambda}{\sum_{x \in NN_\lambda(p)} d(p, x)} \tag{5}$$

where $\lambda$ is natural characteristic value. Different from the definition in [15], the new defined local density does not need to set parameter $k$ by manually, instead, we use natural characteristic value $\lambda$ to set parameter $k$.

## IV. THE PROPOSED ALGORITHM

### A. LOCAL CORES

In [20], local representatives are used to represent the whole data set. However, when searching local representatives, the authors take $\lambda$ nearest neighbors of each point into account. The same number of neighbors for each point makes the method ignore small clusters, so that the local feature of the data set is not well reflected. Local cores in [8], [30] are proposed to solve the existing problem in local representatives. The detailed definitions are as follows.

*Definition 4:* (Representative) Among point $p$ and its the local neighbors, $q$'s local density is the largest, then we say that $q$ is the representative of $p$ and its local neighbors. We denote it as $Rep(p) = q$.

Since we search the local neighbors for each point, there will be a situation that a point is in the local neighbors of two different representatives at the same time. Which representatives should be chosen to be the final representative of the point is what the representative competition rule to do. Additionally, there will also be a situation that if point $p$'s representative is $q$ and $q$'s is $r$, then how to determine the final representative of $p$ is what the representative transfer rule to do. They are formally defined as follows.

**Representative competition rule (RCR)** For point $p$, if $Rep(p) = R1$ and $Rep(p) = R2$ at the same time, then

$Rep(p) = \mathop{\arg\min}_{x\in\{R1,R2\}} \{d(p,x)\}$, that is, the representative closer to point $p$ will be the selected to be the final representative of $p$.

**Algorithm 3** LORE-Searching

```
Input: $LN$: the local neighbors, $Den$: the local density
        computed with Eq. 5
Output: $LORE$: the local cores, $MLORE$: the members
         of local cores, $Rep$: the representative
Initializing: $Rep = \phi$, $LORE = \phi$;
for each point i in the data set do
    $y = \mathop{\arg\max}_{x\in LN(i)}(Den(x))$;
    for each point $x$ in $LN(i)$ do
        if $Rep(x) == \phi$ then
            $Rep(x) = y$;
        end
        if $Rep(x) == z$ and $z \neq y$ then
            //Determine $Rep(x)$ according to RCR;
            if $d(x, y) \leq d(x, z)$ then
                $Rep(x) = y$;
            end
        end
        for each point $p$ in the data set do
            if $Rep(p) == x$ then
                $Rep(p) = Rep(x)$;//Determine $Rep(z)$
                according to RTR;
            end
        end
    end
end
K=1;
for each point $x$ in the data set do
    if $Rep(x) == x$ then
        $LORE(K) = x$;
        $K = K + 1$;
    end
end
for $i$=1:$K$ do
    $MLORE(i)$ = find ($Rep = LORE(i)$);
end
```

**Representative transfer rule (RTR)** If $Rep(p) = q$, and $Rep(q) = r$, then $Rep(p) = r$.

*Definition 5:* (Local Core) After changing the representative of each point according to RCR and RTR, a point $p$ is a local core if $Rep(p) = p$. We denote it as $LORE = \{p|Rep(p) = p\}$.

RCR ensures that each object finds a more reasonable representative. RTR ensures that each representative can represent as large neighborhood as possible, which reduces the number of local cores. local cores are obtained by LORE-Searching algorithm which is described in Algorithm 3. The definition of members of a local core is as follows.

*Definition 6:* (Members of local core) For a local core $p$, the point is one of the members of $p$ if its final representative is $p$. We denote it as $MLORE(p) = \{x|Rep(x) = p\}$.

### B. GRAPH-BASED DISTANCE WITH SHARED-NEIGHBORS OF LOCAL CORES

Euclidean distance cannot well represent the relationship between objects on manifold clusters. Some researchers [31] suggest employing the geodesic distance. However, the exact geodesic distance is difficult to get, because we do not know the prior information about the underlying manifolds. Nevertheless, the authors in [32] have pointed out that if there are sufficient samples from the manifold, then we can use the graph-based distance to approximate the geodesic distance. The graph-based distance can be calculated with the shortest path. The algorithms in [20], [31], [33] mainly use k-nearest neighbor graph to compute the graph-based distance. They construct k-nearest neighbor graph on the whole data set and employ Euclidean distance, density sensitive distance or Gaussian kernel function to weight the edges. However, calculating the shortest path on the whole data set is time consuming. In order to overcome this shortcoming, we define shared-neighbors distance between local cores and only compute the shortest path between local cores on the basis of the new defined distance.

*Definition 7:* (Neighbors of local cores) The neighbors of a local core $p$ are the union of its members(including itself)'s $\lambda$ nearest neighbors. We denote it as $NL(p) = \bigcup_{q\in MLORE(p)} NN_{\lambda}(q)$.

*Definition 8:* (Shared-neighbors between local cores) For two different local cores $p$ and $q$, the shared-neighbors between them are the intersection of their neighbors. We denote it as $SL(p, q) = NL(p) \cap NL(q)$.

*Definition 9:* (Shared-neighbors-based distance) Fore local cores $p$ and $q$, their shared-neighbors-based distance is computed as Eq. 6.

$$SD(p,q) = \begin{cases} \dfrac{d(p,q)}{|SL(p,q)| \times \sum\limits_{o\in SL(p,q)} Den(o)}, & if\,|SL(p,q)| \neq 0 \\ maxd, & if\,|SL(p,q)| = 0 \end{cases} \tag{6}$$

In Eq. 6, $d(p, q)$ represents the Euclidean distance between the local cores $p$ and $q$, $Den(o)$ represents the natural neighbor-based local density of the point $o$ and $maxd$ represents the maximum value of Euclidean distance among all pairs of local cores.

Because of the variation of distribution of the data set, the local cores are unevenly dispersed in the data set. Therefore, it is not suitable for using Euclidean distance to measure the dissimilarity between local cores. Since the shared-neighbors-based distance utilizes the neighbor information between local cores, it shortens the distance between local cores that are closely connected by dense regions and amplifies the distance between local cores that are separated by sparse regions.

In order to obtain the intrinsic geometric features of manifold data sets, the graph-based distance computed with shared-neighbors-based distance between local cores is defined as follows.

Given the shared-neighbors-based distance between local cores, let $P = \{p_1, p_2, ..., p_m\}$ represent the shortest path from $p_1(i.e., p)$ to $p_m(i.e., q)$. Then, the graph-based distance between local cores $p$ and $q$ is computed as Eq 7.

$$GD(p, q) = \sum_{k=1}^{m-1} SD(p_k, p_{k+1}) \tag{7}$$

where $p_k$ and $p_{k+1}$, $1 \leq k < m$ are the local cores along the shortest path $P$.

### C. CLUSTERING LOCAL CORES WITH DP

First, we construct decision graph on local cores. For a local core $p$, we redefine its density $\rho(p)$ and $\delta(p)$ distance. Its density $\rho(p)$ is its natural neighbor-based local density, which is computed as

$$\rho(p) = Den(p) \tag{8}$$

Its $\delta(p)$ distance is computed as shown in Eq. 9

$$\delta(p) = \min_{q \in LORE, \rho(q) > \rho(p)} GD(p, q) \tag{9}$$

For the local cores $p$ with the highest density, its $\delta$ distance is computed as :

$$\delta(p) = \max_{o \in LORE, o \neq p} (\delta(o)) \tag{10}$$

We construct decision graph on local cores according to $\rho(p)$ and $\delta(p)$ for each local core $p$. Since the decision graph gives us a chance to better access to the information contained in the data set and select the preferred clustering result, it is better to keep this kind of user-algorithm interaction.

According to the new decision graph, local cores with relatively large density $\rho$ and $\delta$ distance are selected as the final cluster centers. After the cluster centers have been found, we assign each remaining local core to the same cluster that the local core with higher density and the minimum graph-based distance belongs to.

### D. SHARED-NEIGHBORS OF LOCAL CORES-BASED DP ALGORITHM (SLORE-DP)

On the basis of the above definitions, we propose a novel shared-neighbor of local cores-based DP algorithm (SLORE-DP). The basic idea of SLORE-DP is: first, we find local cores, define the distance between local cores on the basis of shared-neighbor of local cores and calculate the graph-based distance between local cores based on the new defined distance; then, we use the redefined density and $\delta$ distance to construct decision graph and exploit DP to cluster local cores; finally, the remaining points are assigned to the cluster their corresponding local cores belong to. The proposed algorithm SLORE-DP is detailed in Algorithm 4, in which NaN-Searching() is the natural neighbor searching algorithm described in Algorithm 2, LORE-Searching() is to search the local cores and it is described in Algorithm 3 and DP() is the DP algorithm detailed in Algorithm 1.

The steps of the proposed algorithm SLORE-DP includes: (1) find local cores according to the result of NaN-Seaching algorithm, (2) compute the shared-neighbors-based graph-based distance between local cores and (3) employ DP algorithm to cluster local cores. The time complexity of searching natural neighbor information is $O(N\log N)$ when introducing KD-tree and searching local cores is $O(N)$. Therefore, the first step's time complexity is $O(N\log N)$. Assuming the number of local cores is $N_l$ ($N_l \ll N$), then, the time complexity of computing the new graph-based distance between local cores is $O((N_l)^2)$. Since we only cluster local cores with DP algorithm, its time complexity is $O(N_l)$. The overall time complexity of SLORE-DP algorithm is $O(N\log N)$.

**Algorithm 4** SLORE-DP

```
Input: $D$: the data set
Output: $CL$: the cluster label
$(\lambda, LN)$ = NaN-Searching($D$);
for each point $p$ in the data set $D$ do
    Compute the density $Den(p)$ for each point $p$
    according to Eq. 5;
end
$(LORE, MLORE, Rep)$ = LORE-Searching($LN, Den$);
Compute the shared-neighbors-based distance matrix $SD$
according to Eq. 6;
Compute the graph-based distance matrix $GD$ between
local cores according to Eq. 7;
$LORE\_CL$ = DP($Den(LORE), GD$);
for each non-local core $x$ in $D$ do
    $CL(x) = LORE\_CL(Rep(x))$;
end
```

## V. EXPERIMENTAL ANALYSIS

We evaluate the performance of the proposed algorithm SLORE-DP by comparing it with DP and SNN-DPC algorithms. For DP algorithm, to ensure average number of neighbors is around 1% to 2% of the total number of points in the data set, the the authors in [10] suggest setting the cutoff distance as the 2%-th shortest distance and computing the local density with the kernel distance method. For SNN-DPC, $k$ is set as 9, which is also suggested by the authors.

### A. THE COMPARISON OF DECISION GRAPHS ON MANIFOLD DATA SETS

We first compare the decision graphs of DP, SNN-DPC with that of SLORE-DP on two synthetic data sets (Jain and Db2). Jain is from [34], including two moon shaped-clusters with large variation in density, a total of 373 points and Db2 is from [5], consists of four manifold clusters, a total of 315 points.

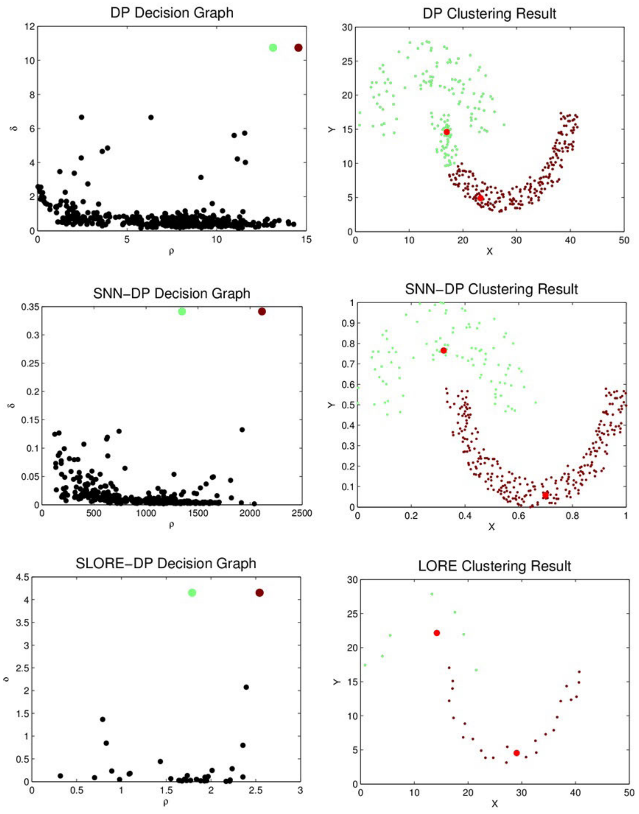


**FIGURE 1.** The comparison of decision graphs on jain.

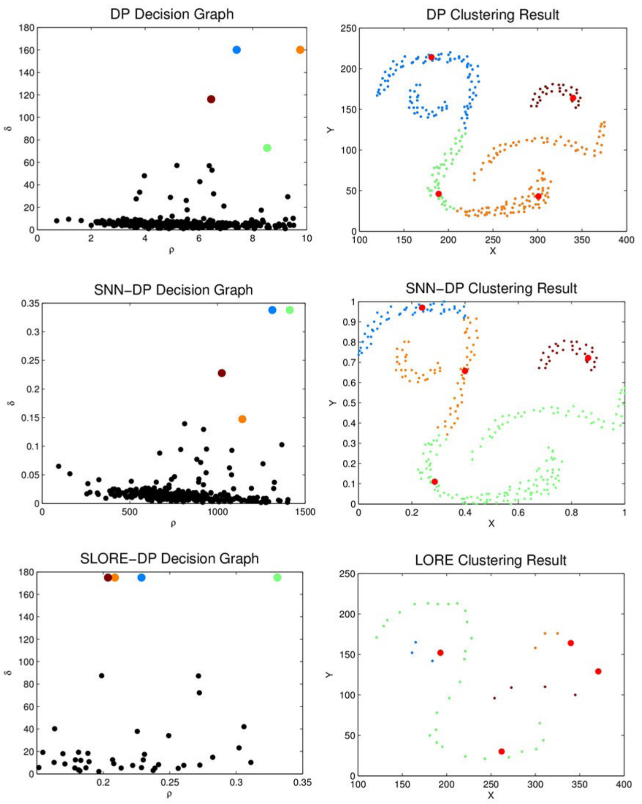


**FIGURE 2.** The comparison of decision graphs on Db2.

The decision graphs are shown in Fig. 1 and Fig. 2, where the red points are selected cluster centers.

As for Jain, DP selects two centers in the dense cluster and no center in the sparse cluster, thus it does not obtain the desired clustering result; SNN-DPC and SLORE-DP correctly find two centers and get the right clustering results. As for Db2, DP and SNN-DPC both choose three centers in the longest cluster and the longest cluster is divided into three clusters; SLORE-DP correctly identifies four centers in the four clusters and obtains the correct clustering result. The results show that DP cannot apply to manifold data sets; SNN-DPC redefines local density and $\delta$ distance using shared-neighbors, which helps it apply to manifold data sets with simple structures, but it still cannot be used to process data sets with complex structures; SLORE-DP uses the shared-neighbors-based distance to calculate the graph-based distance, which preserves the dissimilarity between points on manifold clusters and helps it apply to complex manifold data sets.

### B. PERFORMANCE ON SYNTHETIC DATA SETS AND REAL DATA SETS

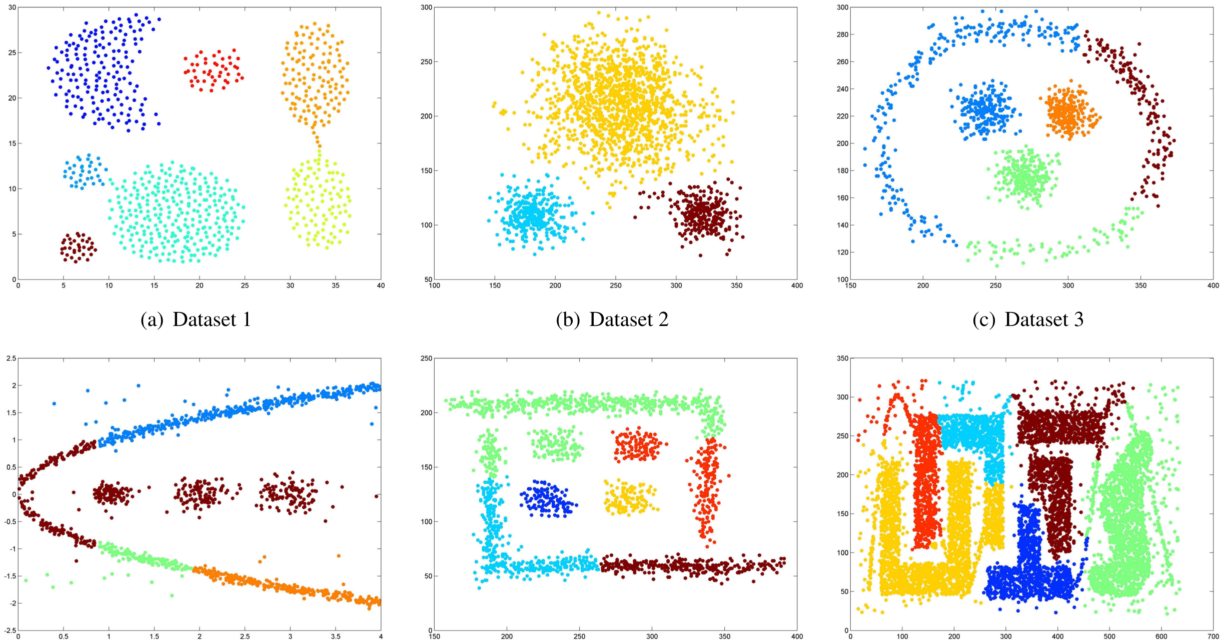
To prove the effectiveness of SLORE-DP, we also compare the performance of the proposed method SLORE-DP, DP and SNN-DP algorithms on more synthetic data sets and real data sets. In this section, the accuracy (ACC) and the normalized mutual information (NMI) [35] are employed to evaluate the clustering performance. They both range from 0 to 1, and the larger the value is, the better the clustering result means. The value of 0 means that the clustering result is useless and the value of 1 tells that the clustering result perfectly matches the class label vector. The configuration of the computer used in our experiment is as follows: processor is Intel Core i7 3.6GHz; memory size is 16GB; programming environment is MATLAB R2013a.

**TABLE 1.** The synthetic data sets.

| Dataset | Instances | Attributes | Clusters | Source |
|---|---|---|---|---|
| Dataset 1 | 788 | 2 | 7 | [36] |
| Dataset 2 | 1873 | 2 | 3 | [30] |
| Dataset 3 | 1156 | 2 | 4 | [30] |
| Dataset 4 | 1368 | 2 | 4 | [37] |
| Dataset 5 | 1741 | 2 | 6 | [30] |
| Dataset 6 | 8000 | 2 | 6 | [38] |

Table 1 has listed the detailed information of synthetic data sets. The clustering results of DP, SNN-DPC and SLORE-DP are shown in Fig. 3-5 and the comparison of ACC, NMI and Time is presented in Table 2.

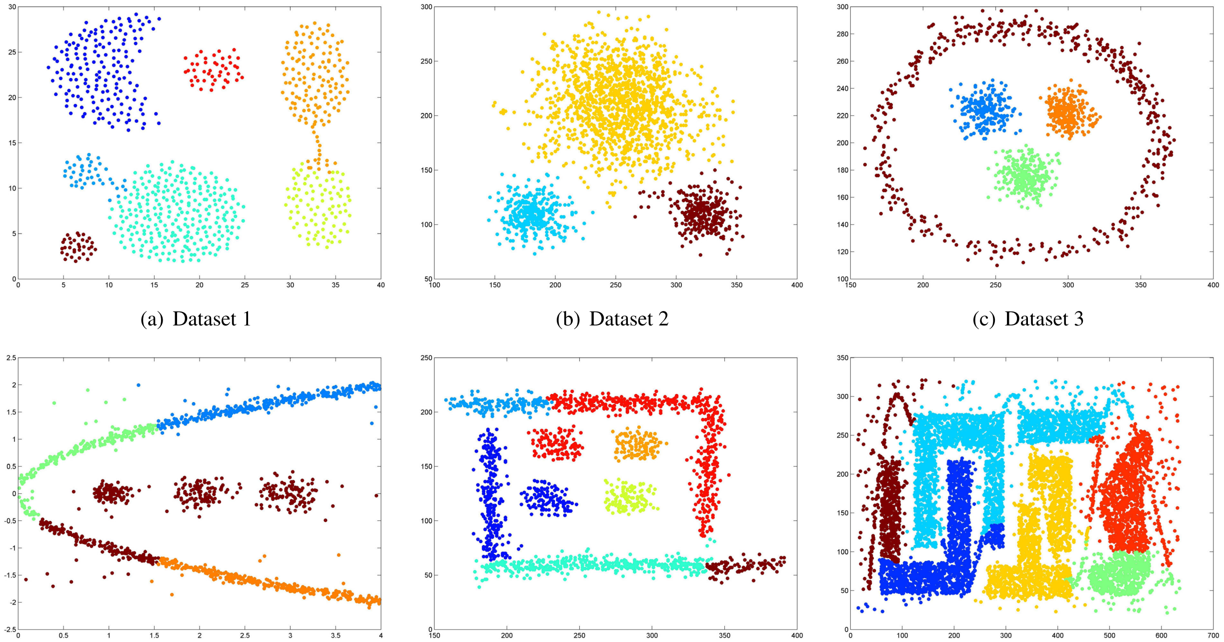
Seen from the results, DP, SNN-DPC and SLORE-DP are all effective to detect spherical clusters, thus ACC and NMI scores of the three algorithms for the first two data sets are equal or close to 1. However, DP cannot process data sets with manifold structures, like Dataset 3-6, because it cannot choose the correct cluster centers according to the decision



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

(d) Dataset 4 (e) Dataset 5 (f) Dataset 6

**FIGURE 3.** **The clustering results of DP.**



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

(d) Dataset 4 (e) Dataset 5 (f) Dataset 6

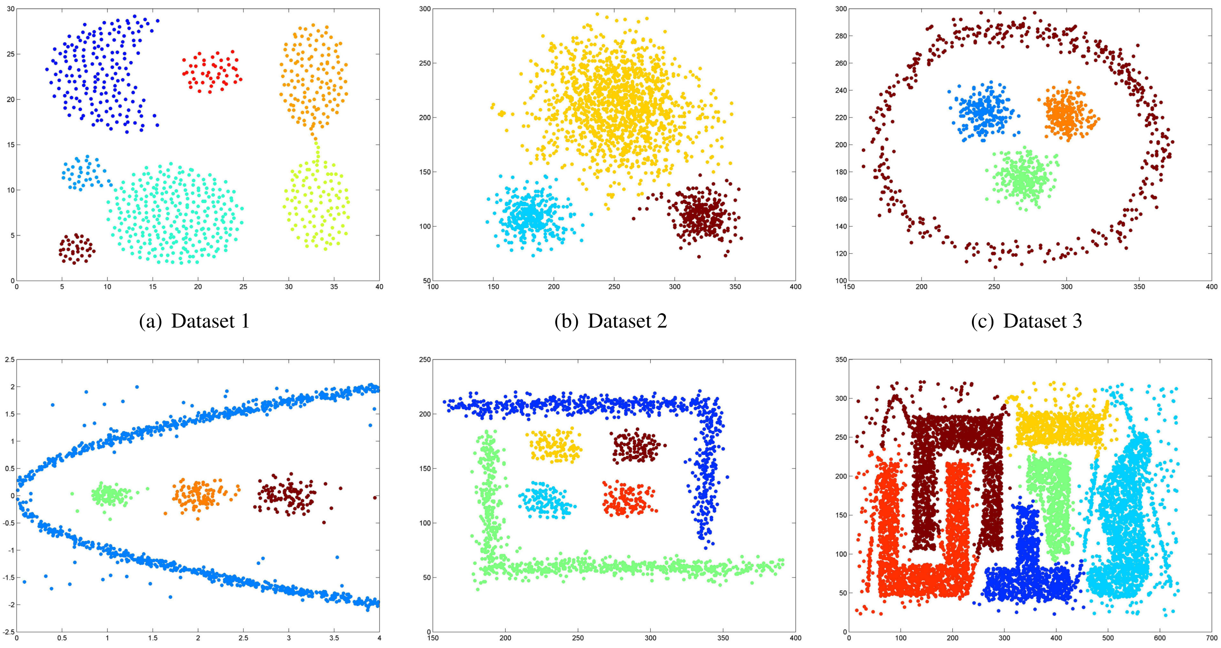
**FIGURE 4.** **The clustering results of SNN-DPC.**

graph. SNN-DPC can be used to cluster manifold data sets, like Dataset 4, but when clustering data sets with long concave clusters or other complex structures, it makes mistakes. The clustering results of ACC and NMI scores show that SLORE-DP performs well when discovering clusters with complex structures. Besides, the running time of SLORE-DP is far less than that of SNN-DP. In terms of the performance on complex manifold clusters and running time, SLORE-DP outperforms DP and SNN-DP algorithms.

We also do experiments on several benchmarking real data sets from UCI, which include Iris, Wine, Seed, Ionosphere, Cancer and Control. Table 3 has shown the detailed

**TABLE 2.** The comparison of ACC, NMI and time on synthetic data sets.

| | | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 | Dataset 6 |
|---|---|---|---|---|---|---|---|
| DP | ACC | **1.00** | **1.00** | 0.72 | 0.39 | 0.65 | 0.84 |
| | NMI | **1.00** | **1.00** | 0.68 | 0.28 | 0.68 | 0.78 |
| | Time(s) | 0.16 | 0.69 | 0.26 | 0.38 | 0.59 | 13.81 |
| SNN-DPC | ACC | 0.97 | **1.00** | **1.00** | 0.37 | 0.67 | 0.64 |
| | NMI | 0.95 | 0.99 | **1.00** | 0.21 | 0.72 | 0.73 |
| | Time(s) | 32.57 | 183.37 | 69.09 | 99.56 | 158.83 | 3361.15 |
| SLORE-DP | ACC | 0.99 | **1.00** | **1.00** | **1.00** | **1.00** | **0.99** |
| | NMI | 0.99 | 0.98 | **1.00** | **1.00** | **1.00** | **0.97** |
| | Time(s) | 1.57 | 3.46 | 1.37 | 1.83 | 2.67 | 40.05 |



(a) Dataset 1

(b) Dataset 2

(c) Dataset 3

(d) Dataset 4

(e) Dataset 5

(f) Dataset 6

**FIGURE 5.** The clustering results of SLORE-DP.

**TABLE 3.** The real data sets from UCI.

| Datasets | Instances | Attributes | Clusters |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Wine | 178 | 13 | 3 |
| Seed | 210 | 7 | 3 |
| Ionosphere | 351 | 34 | 2 |
| Cancer | 569 | 30 | 2 |
| Control | 600 | 60 | 6 |

information of these real data sets. The comparison of ACC, NMI and Time is shown in Table 4. The best results are shown in bold. From the results, we can learn that the ACC and NMI scores of SLORE-DP are the highest for Wine, Seed, Ionosphere and control. For Iris and Cancer, SLORE-DP gets the second best results. Both DP and SNN-DPC achieve the best results on only two data sets. The running time of SLORE-DP is much less than that of SNN-DPC. Therefore, in terms of the performance and running time, SLORE-DP is more effective and efficient than other algorithms.

**TABLE 4.** The comparison of ACC, NMI and time on real data sets.

| | | Iris | Wine | Seed | Ionosphere | Cancer | Control |
|---|---|---|---|---|---|---|---|
| DP | ACC | 0.91 | **0.98** | **0.89** | 0.68 | 0.80 | 0.56 |
| | NMI | 0.81 | **0.91** | 0.70 | 0.09 | 0.35 | 0.75 |
| | Time(s) | 0.04 | 0.04 | 0.04 | 0.05 | 0.12 | 0.18 |
| SNN-DPC | ACC | **0.95** | 0.71 | 0.78 | 0.56 | **0.94** | 0.44 |
| | NMI | **0.84** | 0.63 | 0.63 | 0.07 | **0.68** | 0.67 |
| | Time(s) | 1.21 | 1.71 | 2.36 | 6.54 | 17.20 | 19.18 |
| SLORE-DP | ACC | 0.91 | **0.98** | **0.89** | **0.70** | 0.86 | **0.60** |
| | NMI | 0.81 | **0.91** | **0.71** | **0.12** | 0.49 | **0.82** |
| | Time(s) | 0.17 | 0.18 | 0.21 | 0.34 | 0.57 | 0.67 |

## VI. CONCLUSION

In this work, an improved density peaks clustering algorithm SLORE-DP is proposed, which is based on shared-neighbors of local cores. Its main idea is that it employs natural neighbor-based local density and shared-neighbors-based graph distance between local cores to construct the decision graph in the DP framework to make it applicable to cluster manifold data sets. First, we get local cores and compute the shared-neighbors-based graph distance between local cores. Then, we employ DP algorithm to cluster local cores.

Finally, the non-local cores are assigned to the clusters their local cores belong to. Since we utilize the graph distance with the shared-neighbors-based distance to assess the dissimilarity between local cores, SLORE-DP is effective for clustering manifold data sets and at the same time it avoids calculating the shortest path on the whole data set, reducing the running time. The experimental results on both synthetic and real data sets demonstrate that SLORE-DP is significantly more effective and efficient than DP and SNN-DPC when clustering manifold data sets.

## REFERENCES

[1] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.

[2] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Hoboken, NJ, USA: Wiley, 1990.

[3] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[4] X. Zhang, W. Wang, K. Nørvåg, and M. Sebag, "K-AP: Generating specified $K$ clusters by efficient affinity propagation," in *Proc. 10th IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2010, pp. 1187–1192.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, 1996, pp. 226–231.

[6] Y. Chen, S. Tang, L. Zhou, C. Wang, J. Du, T. Wang, and S. Pei, "Decentralized clustering by finding loose and distributed density cores," *Inf. Sci.*, vols. 433–434, no. 4, pp. 510–526, Apr. 2018.

[7] Y.-A. Geng, Q. Li, R. Zheng, F. Zhuang, R. He, and N. Xiong, "RECOME: A new density-based clustering algorithm using relative KNN kernel density," *Inf. Sci.*, vols. 436–437, pp. 13–30, Apr. 2018.

[8] D. Cheng, Q. Zhu, J. Huang, Q. Wu, and Y. Lijun, "A local cores-based hierarchical clustering algorithm for data sets with complex structures," *Neural Comput. Appl.*, pp. 1–18, Jul. 2018. doi: 10.1007/s00521-018-3641-8.

[9] D. Cheng, Q. Zhu, J. Huang, Q. Wu, and Y. Lijun, "Clustering with local density peaks-based minimum spanning tree," *IEEE Trans. Knowl. Data Eng.*, to be published. doi: 10.1109/TKDE.2019.2930056.

[10] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

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

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

[13] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on $K$-nearest neighbors and principal component analysis," *Knowl.-Based Syst.*, vol. 99, pp. 135–145, May 2016.

[14] L. Ni, W. Luo, C. Bu, and Y. Hu, "Improved CFDP algorithms based on shared nearest neighbors and transitive closure," in *Trends and Applications in Knowledge Discovery and Data Mining—PAKDD* (Lecture Notes in Computer Science), vol. 10526, U. Kang, E. P. Lim, J. Yu, and Y. S. Moon, Eds. Cham, Switzerland: Springer, 2017, pp. 79–93.

[15] G. Wang and Q. Song, "Automatic clustering via outward statistical testing on density metrics," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 1971–1985, Aug. 2016.

[16] J. Xu, G. Wang, and W. Deng, "DenPEHC: Density peak based efficient hierarchical clustering," *Inf. Sci.*, vol. 373, pp. 200–218, Dec. 2016.

[17] Z. Liang and P. Chen, "Delta-density based clustering with a divide-and-conquer strategy: 3DC clustering," *Pattern Recognit. Lett.*, vol. 73, pp. 52–59, Apr. 2016.

[18] J. Chen, X. L. Xiang, H. Zheng, and X. Bao, "A novel cluster center fast determination clustering algorithm," *Appl. Soft Comput.*, vol. 57, pp. 539–555, Aug. 2017.

[19] J. Huang, Q. Zhu, L. Yang, D. Cheng, and Q. Wu, "QCC: A novel clustering algorithm based on quasi-cluster centers," *Mach. Learn.*, vol. 106, no. 3, pp. 337–357, 2017.

[20] D. Cheng, Q. Zhu, J. Huang, L. Yang, and Q. Wu, "Natural neighbor-based clustering algorithm with local representatives," *Knowl.-Based Syst.*, vol. 123, pp. 238–253, May 2017.

[21] S. Yong, Z. Chen, Z. Qi, F. Meng, and L. Cui, "A novel clustering-based image segmentation via density peaks algorithm with mid-level feature," *Neural Comput. Appl.*, vol. 28, no. S1, pp. 29–39, 2016.

[22] D. Liu, Y. Su, X. Li, and Z. Niu, "A novel community detection method based on cluster density peaks," in *Natural Language Processing and Chinese Computing*, X. Huang, J. Jiang, D. Zhao, Y. Feng, and Y. Hong, Eds. Cham, Switzerland: Springer, 2018, pp. 515–525.

[23] X. Bai, P. Yang, and X. Shi, "An overlapping community detection algorithm based on density peaks," *Neurocomputing*, vol. 226, pp. 7–15, Feb. 2017.

[24] M. Wang, W. Zuo, and Y. Wang, "An improved density peaks-based clustering method for social circle discovery in social networks," *Neurocomputing*, vol. 179, pp. 219–227, Feb. 2016.

[25] W. Luo, Z. Yan, C. Bu, and D. Zhang, "Community detection by fuzzy relations," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[26] B. Wang, J. Zhang, Y. Liu, and Y. Zou, "Density peaks clustering based integrate framework for multi-document summarization," *CAAI Trans. Intell. Technol.*, vol. 2, no. 1, pp. 26–30, 2017.

[27] S. Wang, D. Wang, C. Li, Y. Li, and G. Ding, "Clustering by fast search and find of density peaks with data field," *Chin. J. Electron.*, vol. 25, no. 3, pp. 397–402, 2016.

[28] Q. Zhu, J. Feng, and J. Huang, "Natural neighbor: A self-adaptive neighborhood method without parameter $K$," *Pattern Recognit. Lett.*, vol. 80, pp. 30–36, Sep. 2016.

[29] L. Yang, Q. Zhu, J. Huang, and D. Cheng, "Adaptive edited natural neighbor algorithm," *Neurocomputing*, vol. 230, pp. 427–433, Mar. 2017.

[30] D. Cheng, Q. Zhu, J. Huang, Q. Wu, and L. Yang, "A novel cluster validity index based on local cores," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 985–999, Apr. 2018. doi: 10.1109/TNNLS.2018.2853710.

[31] E. Tu, L. Cao, J. Yang, and N. Kasabov, "A novel graph-based $K$-means for nonlinear manifold clustering and representative selection," *Neurocomputing*, vol. 143, pp. 109–122, Nov. 2014.

[32] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[33] P. Yang, Q. Zhu, and B. Huang, "Spectral clustering with density sensitive similarity function," *Knowl.-Based Syst.*, vol. 24, no. 5, pp. 621–628, 2011.

[34] A. K. Jain and M. H. C. Law, "Data clustering: A user's dilemma," in *Pattern Recognition and Machine Intelligence*, vol. 3776. Berlin, Germany: Springer, 2005, pp. 1–10.

[35] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 568–586, Mar. 2011.

[36] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, 2007, Art. no. 4.

[37] J. Ha, S. Seok, and J.-S. Lee, "Robust outlier detection using the instability factor," *Knowl.-Based Syst.*, vol. 63, pp. 15–23, Jun. 2014.

[38] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.

**DONGDONG CHENG** received the bachelor's degree in computer science from Chongqing Normal University, in 2013, and the Ph.D. degree from Chongqing University, in 2018. She is currently a Lecturer with the College of Big Data and Intelligent Engineering, Yangtze Normal University. Her research interests are clustering analysis and data mining.

**JINLONG HUANG** received the Ph.D. degree from Chongqing University, in 2017. He is currently a Lecturer with the College of Big Data and Intelligent Engineering, Yangtze Normal University. His research interests include outlier detection and clustering analysis.

**SULAN ZHANG** received the B.S. degree from the Department of Computer Science and Technology, Southwest University, China, in 2006, and the master's degree in computer software and theory and the Ph.D. degree in computer science and technology from Chongqing University, China, in 2009 and 2013, respectively. She is currently an Associate Professor with the College of Big Data and Intelligent Engineering, Yangtze Normal University, China. Her main research interests include data mining, data analysis, and computer modeling and simulation.

**HUIJUN LIU** received the B.E., M.S., and Ph.D. degrees in computer science, Chongqing University, Chongqing, China, in 1999, 2004, and 2010, respectively. He is currently an Associate Professor with the College of Computer Science, Chongqing University. His current research interests include data mining and complex networks.