Pattern Recognition 71 (2017) 375–386

Contents lists available at ScienceDirect

Pattern Recognition

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

Fast density clustering strategies based on the *k*-means algorithm

Liang Bai[a,b,c,*], Xueqi Cheng[b], Jiye Liang[a], Huawei Shen[b], Yike Guo[c]

[a] School of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi, 030006, China
[b] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
[c] Department of Computing, Imperial College London, SW7, London, United Kingdom

ARTICLE INFO

Article history:
Received 21 September 2016
Revised 26 April 2017
Accepted 16 June 2017
Available online 17 June 2017

Keywords:
Cluster analysis
Density-based clustering
Acceleration mechanism
Approximate algorithm
k-means

ABSTRACT

Clustering by fast search and find of density peaks (CFSFDP) is a state-of-the-art density-based clustering algorithm that can effectively find clusters with arbitrary shapes. However, it requires to calculate the distances between all the points in a data set to determine the density and separation of each point. Consequently, its computational cost is extremely high in the case of large-scale data sets. In this study, we investigate the application of the *k*-means algorithm, which is a fast clustering technique, to enhance the scalability of the CFSFDP algorithm while maintaining its clustering results as far as possible. Toward this end, we propose two strategies. First, based on concept approximation, an acceleration algorithm (CFSFDP+A) involving fewer distance calculations is proposed to obtain the same clustering results as those of the original algorithm. Second, to further expand the scalability of the original algorithm, an approximate algorithm (CFSFDP+DE) based on exemplar clustering is proposed to rapidly obtain approximate clustering results of the original algorithm. Finally, experiments are conducted to illustrate the effectiveness and scalability of the proposed algorithms on several synthetic and real data sets.

1. Introduction

Cluster analysis is a statistical multivariate analysis technique involving pattern recognition based on unsupervised learning. It has extensive applications in various domains, including financial fraud, medical diagnosis, image processing, information retrieval, and bioinformatics [1]. Clustering is the process of grouping a set of points into clusters such that the points in the same cluster have high similarity but are significantly dissimilar from the points in other clusters.

Various types of clustering algorithms have been proposed and developed in the literature (e.g., [2,3] and the references therein). In general, these algorithms are categorized into four types: partitional, hierarchical, density-based, and grid-based clustering. Representative algorithms can be found in [4–17]. The advantages of density-based clustering algorithms over other types of clustering algorithms are that they can find clusters with arbitrary shapes and they do not require the number of clusters as input. However, their disadvantage is that they involve high computational costs. Representative algorithms include density-based spatial clustering of applications with noise (DBSCAN) [12], ordering points to identify the clustering structure (OPTICS) [13], mean-shift clustering (MSC) [18,19], and clustering by fast search and find of density peaks (CFSFDP) [20]. The CFSFDP algorithm, which was published in [*Science, 2014*], is based on cluster centers. It needs to manually determine the number of clusters and select several points as cluster centers to represent the clusters according to the density and separation of each point. In the algorithm, a cluster is represented by a cluster center with higher local density and separation from other cluster centers, and it is surrounded by neighbors with lower local density. After the cluster centers have been obtained, each remaining point is assigned to the same cluster as its nearest neighbor with higher density. The CFSFDP algorithm determines the membership of a point to a cluster by considering not only the connectivity but also the separation of points. Thus, its performance is robust with respect to the radius of a neighborhood, compared to other density-based algorithms. However, as with other density-based algorithms, it requires to calculate the distances between all the points in a data set to determine the densities of the points and separations between the points. Consequently, its computational cost is extremely high in the case of large-scale data sets.

Many techniques have been proposed to reduce the unnecessary computations of distances and enhance the efficiency of clustering algorithms as follows. (1) *Spatial index structure*: An index structure, such as KD-tree [21], R*-tree [22], or X-tree [23], may

* Corresponding author at: School of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi, 030006, China.
E-mail addresses: sxbailiang@hotmail.com (L. Bai), cxq@ict.ac.cn (X. Cheng), ljy@sxu.edu.cn (J. Liang), shenhuawei@ict.ac.cn (H. Shen), y.guo@imperial.ac.uk (Y. Guo).

http://dx.doi.org/10.1016/j.patcog.2017.06.023
0031-3203/

be applied to a data set to efficiently find the neighbors of a point on the basis of a distance measure. However, the computational complexity of this solution increases exponentially with the data dimensions. As a general rule, if a data set has n points and m dimensions, n should be considerably greater than 2^m, which can enhance the search efficiency using index structures. Otherwise, the distances between most of the points will need to be computed in the search process. (2) *Grid-based clustering*: The density-based clustering algorithm DENCLUE [14] only maintains information about grid cells that actually contain data points, and it manages these cells in a tree-based access structure. It is significantly faster than other algorithms for high-dimensional data sets. The STING algorithm [15] uses grid-based clustering to enhance the efficiency of DBSCAN. However, the grid-based technique only obtains an approximate result compared to the original algorithms and is efficient for dealing with low-dimensional data sets. As the number of dimensions increases, the number of grids increases exponentially, resulting in high computational costs. (3) *Hybrid clustering*: Several algorithms, such as BRIDGE [24], I-DBSCAN [25], and rough-DBSCAN [26], integrate the k-means [4] and DBSCAN algorithms to cluster large-scale data sets. These algorithms first find suitable prototypes from the large-scale data set and then apply the clustering method using only the prototypes. Nanda et al. [27] proposed a modified DBSCAN algorithm that involves a new merging criterion of preliminary clusters and uses correlation coefficients to reduce the computational complexity of DBSCAN. (4) *Parallel clustering*: Parallel techniques are used to enhance the clustering speed of the original algorithms. For example, some scholars have proposed parallel density-based clustering algorithms using the MapReduce technique [28], such as MR-DBSCAN [29] and DBCURE-MR [30]. Chen et al. [31] proposed a parallel-computing solution for spectral clustering [17]. Despite their theoretical and practical advantages, the above-mentioned techniques cannot be directly applied to the CFSFDP algorithm, because each clustering algorithm has its own clustering criterion and searching strategy. Recently, several scholars have developed some improved CFSFDP algorithms [32–34]. For example, Wu et al. [32] proposed a density- and grid-based clustering (DGB) algorithm to improve the clustering speed of the CFSFDP algorithm. However, the DGB algorithm is only suitable for dealing with two-dimensional data. Zhang et al. [33] proposed a distributed CFSFDP algorithm based on the MapReduce technique. Gao et al. [34] proposed an improved CFSFDP algorithm that considers how to enhance the clustering accuracy of the original algorithm. This algorithm involves additional computational costs compared to CFSFDP. Although these algorithms improve the performance of the CFSFDP algorithm, they do not fully consider how to reduce redundant computations and accelerate the CFSFDP algorithm.

Therefore, in this paper, we use the k-means algorithm to enhance the scalability of the CFSFDP algorithm. The k-means algorithm [4] is well known for its efficiency in clustering large-scale data sets. However, it is not suitable for determining non-spherical clusters. We propose two strategies to integrate the advantages of the CFSFDP and k-means algorithms in order to rapidly discover any clusters with any shape. The first strategy is to design an acceleration mechanism based on concept approximation. In this mechanism, we use the estimation of distances to construct approximation spaces that are required for computing the densities and separations of points. This can significantly reduce the calculation of distances. The accelerated CFSFDP algorithm (i.e., CFSFDP+A) requires a shorter computational time and fewer distance calculations while providing the same clustering results. The second strategy is to design an approximate algorithm (i.e., CFSFDP+DE) of the CFSFDP algorithm on the basis of exemplar clustering in order to further expand the scalability of the original algorithm. This algorithm can rapidly obtain approximate clustering results of the original algorithm. We analyze the precision loss of the approximate algorithm experimentally.

The remainder of this paper is organized as follows. Section 2 reviews the CFSFDP and k-means algorithms. Section 3 presents the accelerated CFSFDP algorithm (CFSFDP+A). Section 4 presents the approximate algorithm (CFSFDP+DE). Section 5 illustrates the effectiveness and efficiency of the proposed algorithms on several data sets. Finally, Section 6 concludes the paper.

2. Preliminaries

The CFSFDP algorithm was proposed by Alex Rodriguez and Alessandro Laio [20]. It is based on the assumptions that cluster centers are surrounded by neighbors with lower local density and that they are at relatively large distances from points with higher local density. For each data point $\mathbf{x}_i$, we need to compute two quantities: its local density $\rho(\mathbf{x}_i)$ and its separation $\delta(\mathbf{x}_i)$. Let d be a distance measure that is assumed to satisfy the triangular inequality, let $B(\mathbf{x}_i) = \{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d_c, \mathbf{x}_j \in \mathbf{X}\}$ be the neighborhood of point $\mathbf{x}_i$, and let d_c be the radius of the neighborhood. The density and separation measures are defined as

$$\rho(\mathbf{x}_i) = |B(\mathbf{x}_i)|, \tag{1}$$

and

$$\delta(\mathbf{x}_i) = \begin{cases} \min\limits_{\rho(\mathbf{x}_j) > \rho(\mathbf{x}_i)} d(\mathbf{x}_i, \mathbf{x}_j) \\ \max\limits_{\mathbf{x}_j \in \mathbf{X}} d(\mathbf{x}_i, \mathbf{x}_j), \quad otherwise. \end{cases} \tag{2}$$

The local density measure ρ of a point is used to represent the number of points in its neighborhood. The separation measure δ of a point is used to evaluate its isolation from other points having higher density. The smaller the value of δ is, the greater is the possibility that the point is assigned to the same cluster as its nearest neighbor with higher density. The algorithm first manually selects several points with high δ and relatively high ρ as the cluster centers. After the cluster centers are found, each remaining point is assigned to the same cluster as its nearest neighbor with higher density. If some points have relatively high δ and low ρ, they are considered as clusters composed of a single point, namely, outliers. The cluster assignment is performed in a single step. Further details regarding the clustering process can be found in [20]. Although the algorithm can be used to effectively and simply find clusters with arbitrary shapes, it needs to compute the distances between all the points to determine the densities and separations of the points. Thus, the time complexity of the algorithm is $O(n^2)$ in terms of the number of distance calculations. However, in the case of large-scale data sets, the computational cost is extremely high. Therefore, how to enhance the scalability of the algorithm is a critical issue.

The k-means algorithm [4] is one of the most efficient clustering techniques, which begins with an initial set of cluster centers and iteratively refines this set so as to decrease the sum of squared errors. It has attracted considerable interest in the literature. The k-means algorithm minimizes the objective function F as follows:

$$F(\mathbf{S}, \mathbf{V}) = \sum_{l=1}^{k_m} \sum_{\mathbf{x}_i \in S_l} d(\mathbf{x}_i, \mathbf{v}_l),$$

where $\mathbf{S} = \{S_1, S_2, \cdots, S_{k_m}\}$ is a partition of $\mathbf{X}$ and $S_l \subset \mathbf{X}$, $S_l \cap S_q = \varnothing$, $\bigcup_{l=1}^{k_m} S_l = \mathbf{X}$, for $1 \le l \ne q \le k_m$, $\mathbf{V} = \{\mathbf{v}_l\}_{l=1}^{k_m}$, and $\mathbf{v}_l$ is the lth center of S_l. The optimal minimum value of F is normally obtained by an alternative optimization method. It relates to two updated equations for $\mathbf{S}$ and $\mathbf{V}$. The updated equations are expressed as follows. Given $\mathbf{V}$, $\mathbf{S}$ is updated by

$$\mathbf{x}_i \in S_l, \ if \ l = \arg\min_{l=1}^{k_m} d(\mathbf{x}_i, \mathbf{v}_l),$$

for $1 \le i \le N$, $1 \le l \le k_m$. Given **S, V** is updated by

$$\mathbf{v}_l = \frac{\sum_{\mathbf{x}_i \in S_l} \mathbf{x}_i}{|S_l|},$$

for $1 \le l \le k_m$. The time complexity of the k-means algorithm is $O(nk_mt)$, where t is the number of iterations. The algorithm only needs to compute the distances between points and centers, and its cost is very low compared to the cost of computing the distances between all the points. The performance of the k-means algorithm is affected by many factors. For example, the algorithm is very sensitive to the initial cluster centers. Different initial selections often lead to different clustering results. Moreover, the k-means algorithm tends to discover spherical clusters with relatively uniform sizes [35]. However, it is not suitable for other data distributions.

Therefore, we wish to integrate the advantages of the k-means and CFSFDP algorithms to rapidly find clusters with any shape. The CFSFDP algorithm searches the global space and compute the distances between all the points to determine δ and ρ. However, a cluster tends to exist in a local space, which can provide a smaller search range than the global space. Furthermore, to obtain an exact or approximate clustering result of the CFSFDP algorithm, we need not compute the distances between all the points. Therefore, in this paper, we first use the k-means algorithm to initially partition a given data set **X** into k_m subsets, which can provide the local space of each point and the distances between points and centers. We investigate how to enhance the scalability of the CFSFDP algorithm by using the local spaces instead of the global space and the distances between points and centers instead of the distances between all the points. The initial partition is required to not produce an "exact" clustering result but rapidly obtain k_m subsets such that data points close to each other in the geometrical space are likely to be placed in the same subset. Thus, we propose that the k-means algorithm be applied with k_m randomly selected initial centers to produce an initial partition. Here, k_m should be larger than the number of clusters k. The larger the value of k_m, the closer are the points in the same subsets in the geometrical space. However, if k_m is too large, the pre-processing cost will be high. Therefore, k_m should be considerably smaller than n.

3. Accelerated CFSFDP algorithm

In this section, we discuss how to rapidly obtain the "exact" clustering results of the CFSFDP algorithm by using the approximation description. The CFSFDP algorithm needs to compute the density $\rho(\mathbf{x}_i)$ and the separation $\delta(\mathbf{x}_i)$ of each $\mathbf{x}_i$ in **X**, in order to select the cluster centers and assign points to the clusters. This is a crucial step. For $\rho(\mathbf{x}_i)$, the CFSFDP algorithm needs to identify the neighborhood $B(\mathbf{x}_i)$. For $\delta(\mathbf{x}_i)$, it needs to find the point $\mathbf{y}_i$ because $\delta(\mathbf{x}_i) = d(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{y}_i = \arg\min_{\rho(\mathbf{x}_j)>\rho(\mathbf{x}_i), \mathbf{x}_j \in \mathbf{X}} d(\mathbf{x}_i, \mathbf{x}_j)$. We need to calculate the distance between each point $\mathbf{x}_j$ in **X** and $\mathbf{x}_i$ in order to obtain the neighborhood $B(\mathbf{x}_i)$ and the point $\mathbf{y}_i$. The entire process requires around $2n^2$ distance calculations and is hence very time-consuming when n is extremely large. Therefore, there are two key issues in terms of enhancing the efficiency of the CFSFDP algorithm:

- **How to reduce the distance calculations while computing the ρvalues of each point**.
- **How to reduce the distance calculations while computing the δvalues of each point**.

Given an initial partition **S** by the k-means algorithm, let $B_l(\mathbf{x}_i) = \{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d_c, \mathbf{x}_j \in S_l\}$ be a set including the points in S_l that belong to $B(\mathbf{x}_i)$. Then, we can rewrite $\rho(\mathbf{x}_i)$ and $B(\mathbf{x}_i)$ as

$$\rho(\mathbf{x}_i) = \sum_{l=1}^{k_m} |B_l(\mathbf{x}_i)|$$

and

$$B(\mathbf{x}_i) = \bigcup_{l=1}^{k_m} B_l(\mathbf{x}_i).$$

To determine the points in S_l that belong to $B_l(\mathbf{x}_i)$, we do not directly compute the distance between $\mathbf{x}_i$ in S_h and each $\mathbf{x}_j$ in S_l. Considering the triangle inequalities

$$\left|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{x}_j, \mathbf{v}_l)\right| \le d(\mathbf{x}_i, \mathbf{x}_j) \le d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{x}_j, \mathbf{v}_l)$$

and

$$\left|d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{x}_j, \mathbf{v}_h)\right| \le d(\mathbf{x}_i, \mathbf{x}_j) \le d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{x}_j, \mathbf{v}_h),$$

we can estimate $d(\mathbf{x}_i, \mathbf{x}_j)$ as follows.

$$\max\{\left|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{x}_j, \mathbf{v}_l)\right|, \left|d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{x}_j, \mathbf{v}_h)\right|\} \le d(\mathbf{x}_i, \mathbf{x}_j)$$
$$\le \min\{d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{x}_j, \mathbf{v}_l), d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{x}_j, \mathbf{v}_h)\}.$$

Base on the bounds, the following inequality rules are obtained to reduce several unnecessary operations when constructing $B(\mathbf{x}_i)$:

(a) If $\left|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{x}_j, \mathbf{v}_l)\right| \ge d_c$ or $\left|d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{x}_j, \mathbf{v}_h)\right| \ge d_c$ and $\mathbf{x}_j \in S_l$, the point $\mathbf{x}_j$ does not belong to $B_l(\mathbf{x}_i)$.
(b) If $d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{x}_j, \mathbf{v}_l) < d_c$ or $d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{x}_j, \mathbf{v}_h) < d_c$ and $\mathbf{x}_j \in S_l$, the point $\mathbf{x}_j$ belongs to $B_l(\mathbf{x}_i)$.
(c) If $d(\mathbf{x}_i, \mathbf{v}_l) - r_l \ge d_c$, where $r_l = \max_{\mathbf{x} \in S_l} d(\mathbf{x}, \mathbf{c}_l)$ is the radius of S_l, no point in S_l belongs to $B_l(\mathbf{x}_i)$.
(d) If $d(\mathbf{x}_i, \mathbf{v}_l) + r_l < d_c$, where $r_l = \max_{\mathbf{x} \in S_l} d(\mathbf{x}, \mathbf{c}_l)$, all the points in S_l belong to $B_l(\mathbf{x}_i)$.

Similar to the rough set theory [36], these rules are used to build the upper and lower approximations for each $B_l(\mathbf{x}_i)$, namely, $\overline{B_l}(\mathbf{x}_i)$ and $\underline{B_l}(\mathbf{x}_i)$, which are described as follows.

$$\overline{B_l}(\mathbf{x}_i) = \{\mathbf{x}_j | \left|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{x}_j, \mathbf{v}_l)\right| < d_c \wedge \left|d(\mathbf{x}_j, \mathbf{v}_h) - d(\mathbf{x}_i, \mathbf{v}_h)\right| < d_c, \mathbf{x}_j \in S_l\}$$

and

$$\underline{B_l}(\mathbf{x}_i) = \{\mathbf{x}_j | \ d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{x}_j, \mathbf{v}_l) < d_c \vee d(\mathbf{x}_j, \mathbf{v}_h) + d(\mathbf{x}_i, \mathbf{v}_h) < d_c, \mathbf{x}_j \in S_l\}.$$

We use these two sets to approximately describe $B_l(\mathbf{x}_i)$ (see Fig. 1). $\underline{B_l}(\mathbf{x}_i)$ denotes a set including the points that belong to $B_l(\mathbf{x}_i)$, while $\overline{B_l}(\mathbf{x}_i)$ denotes a set including the points that may belong to $B_l(\mathbf{x}_i)$. Further, $S_l - \overline{B_l}(\mathbf{x}_i)$ denotes a set including the points that do not belong to $B_l(\mathbf{x}_i)$. These sets have the following relation

$$\underline{B_l}(\mathbf{x}_i) \subseteq B_l(\mathbf{x}_i) \subseteq \overline{B_l}(\mathbf{x}_i).$$

Based on the above relation, an approximate boundary of $B_l(\mathbf{x}_i)$ can be obtained, which is defined as follows:

$$Bn_l(\mathbf{x}_i) = \overline{B_l}(\mathbf{x}_i) - \underline{B_l}(\mathbf{x}_i).$$

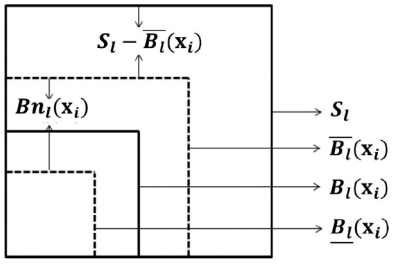

Fig. 1. Approximate description of $B_l(\mathbf{x}_i)$.

According to the approximate description, we can rewrite $B_l(\mathbf{x}_i)$ as follows:

$$B_l(\mathbf{x}_i) = \underline{B_l}(\mathbf{x}_i) \cup \{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d_c, \mathbf{x}_j \in Bn_l(\mathbf{x}_i)\}.$$

When computing the exact $B_l(\mathbf{x}_i)$, we make the following observations: (1) the points in $S_l - \overline{B_l}(\mathbf{x}_i)$ can be directly rejected to construct $B_l(\mathbf{x}_i)$; (2) the points in $\underline{B_l}(\mathbf{x}_i)$ can be directly accepted to construct $B_l(\mathbf{x}_i)$; and (3) $Bn_l(\mathbf{x}_i)$ is an uncertain region. We need to compute $d(\mathbf{x}_i, \mathbf{x}_j)$ for each $\mathbf{x}_j$ in $Bn_l(\mathbf{x}_i)$ to determine whether $\mathbf{x}_j$ belongs to $B_l(\mathbf{x}_i)$. This indicates that a smaller $|Bn_l(\mathbf{x}_i)|$ value corresponds to a smaller number of distances that are required to be computed. Therefore, a large number of unnecessary distance calculations can be reduced by $\bigcup_{l=1}^{k_m} \underline{B}(\mathbf{x}_i)$ and $\bigcup_{l=1}^{k_m} \overline{B}(\mathbf{x}_i)$ for $1 \le l \le k_m$ while computing the exact $\rho(\mathbf{x}_i)$.

After computing $\rho(\mathbf{x}_i)$ for $1 \le i \le n$, we discuss how to rapidly compute $\delta(\mathbf{x}_i)$. According to the definition of $\delta(\mathbf{x}_i)$, we need to find its nearest neighbor $\mathbf{y}_i$ with higher density from the search space. The searching cost of $\mathbf{y}_i$ depends on the size and construction cost of the search space. Thus, we will try to rapidly construct a very small space including $\mathbf{y}_i$ to reduce the searching cost. First, we define a search space including $\mathbf{y}_i$ as follows:

$$G(\mathbf{x}_i, d_m) = \{\mathbf{x}_j | \rho(\mathbf{x}_i) < \rho(\mathbf{x}_j) \wedge d(\mathbf{x}_i, \mathbf{x}_j) \le d_m, \mathbf{x}_j \in \mathbf{X}\},$$

where d_m is the distance between $\mathbf{x}_i$ and some point with higher density; it controls the size of $G(\mathbf{x}_i, d_m)$. If we rapidly construct $G(\mathbf{x}_i, d_m)$ with a smaller size, the computational time of $\mathbf{y}_i$ is reduced. Similar to computing $B(\mathbf{x}_i)$, the upper and lower approximations are constructed for each $G_l(\mathbf{x}_i, d_m)$, namely, $\overline{G_l}(\mathbf{x}_i, d_m)$ and $\underline{G_l}(\mathbf{x}_i, d_m)$, which are described as follows.

$$\begin{aligned}\overline{G_l}(\mathbf{x}_i, d_m) =& \{\mathbf{x}_j \big| \big| d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{x}_j, \mathbf{v}_l) \big| < d_m \\ & \wedge \big| d(\mathbf{x}_j, \mathbf{v}_h) - d(\mathbf{x}_i, \mathbf{v}_h) \big| < d_m \\ & \wedge \rho(\mathbf{x}_i) < \rho(\mathbf{x}_j), \mathbf{x}_j \in S_l\}\end{aligned}$$

and

$$\begin{aligned}\underline{G_l}(\mathbf{x}_i, d_m) = \quad & \{\mathbf{x}_j | \ (d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{x}_j, \mathbf{v}_l) < d_m \\ & \vee d(\mathbf{x}_j, \mathbf{v}_h) + d(\mathbf{x}_i, \mathbf{v}_h) < d_m) \\ & \wedge \rho(\mathbf{x}_i) < \rho(\mathbf{x}_j), \mathbf{x}_j \in S_l\}.\end{aligned}$$

$\underline{G_l}(\mathbf{x}_i, d_m)$ denotes a set including the points that belong to $G_l(\mathbf{x}_i, d_m)$, while $\overline{G_l}(\mathbf{x}_i, d_m)$ denotes a set including the points that may belong to $G_l(\mathbf{x}_i, d_m)$. The approximate boundary of $G_l(\mathbf{x}_i, d_m)$ is given as follows:

$$Gn_l(\mathbf{x}_i, d_m) = \overline{G_l}(\mathbf{x}_i, d_m) - \underline{G_l}(\mathbf{x}_i, d_m).$$

According to the approximate description, we can compute $G_l(\mathbf{x}_i, d_m)$ by the following relation

$$\begin{aligned}G_l(\mathbf{x}_i, d_m) =& \underline{G_l}(\mathbf{x}_i, d_m) \cup \{\mathbf{x}_j | d(\mathbf{x}_i, \mathbf{x}_j) < d_m, \\ & \mathbf{x}_j \in Gn_l(\mathbf{x}_i, d_m)\}.\end{aligned}$$

While computing $\delta(\mathbf{x}_i)$, we can directly reject the points in $S_l - \overline{G_l}(\mathbf{x}_i, d_m)$ and search the region $\overline{G_l}(\mathbf{x}_i, d_m)$ for $\mathbf{y}_i$. However, $\overline{G_l}(\mathbf{x}_i, d_m)$ is a large region. Therefore, we need to further reduce the size of the search region. We know that if

$$\begin{aligned}&\min\{d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{y}, \mathbf{v}_l), d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{y}, \mathbf{v}_h)\} < \\ &\max\{|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{z}, \mathbf{v}_l)|, |d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{z}, \mathbf{v}_h)|\},\end{aligned}$$

then $d(\mathbf{x}_i, \mathbf{y}) < d(\mathbf{x}_i, \mathbf{z})$. Thus, we define the following two sets:

$$\begin{aligned}\overline{D_l}(\mathbf{x}_i, d_m) =& \{\mathbf{y} | \min\{d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{y}, \mathbf{v}_l), d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{y}, \mathbf{v}_h)\} \\ & < \max\{|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{z}, \mathbf{v}_l)|, |d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{z}, \mathbf{v}_h)|\} \\ & \vee \max\{|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{z}, \mathbf{v}_l)|, |d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{z}, \mathbf{v}_h)|\} \\ & \not< \min\{d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{y}, \mathbf{v}_l), d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{y}, \mathbf{v}_h)\}, \\ & \forall \mathbf{y}, \mathbf{z} \in \overline{G_l}(\mathbf{x}_i, d_m)\}\end{aligned}$$

and

$$\begin{aligned}\underline{D_l}(\mathbf{x}_i, d_m) =& \{\mathbf{y} | \min\{d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{y}, \mathbf{v}_l), d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{y}, \mathbf{v}_h)\} \\ & < \max\{|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{z}, \mathbf{v}_l)|, |d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{z}, \mathbf{v}_h)|\} \\ & \vee \max\{|d(\mathbf{x}_i, \mathbf{v}_l) - d(\mathbf{z}, \mathbf{v}_l)|, |d(\mathbf{x}_i, \mathbf{v}_h) - d(\mathbf{z}, \mathbf{v}_h)|\} \\ & \not< \min\{d(\mathbf{x}_i, \mathbf{v}_l) + d(\mathbf{y}, \mathbf{v}_l), d(\mathbf{x}_i, \mathbf{v}_h) + d(\mathbf{y}, \mathbf{v}_h)\}, \\ & \forall \mathbf{y}, \mathbf{z} \in \underline{G_l}(\mathbf{x}_i, d_m)\}.\end{aligned}$$

According to the definition, we know that

$$\underline{D_l}(\mathbf{x}_i, d_m) \subseteq \overline{D_l}(\mathbf{x}_i, d_m) \subseteq \overline{G_l}(\mathbf{x}_i, d_m).$$

Because

$$\begin{aligned}&d(\mathbf{x}_i, \mathbf{y}) < d(\mathbf{x}_i, \mathbf{z}), \forall \mathbf{y} \in \overline{D_l}(\mathbf{x}_i, d_m), \\ &\quad \forall \mathbf{z} \in \overline{G_l}(\mathbf{x}_i, d_m) - \overline{D_l}(\mathbf{x}_i, d_m)\end{aligned}$$

and

$$\begin{aligned}&d(\mathbf{x}_i, \mathbf{y}) < d(\mathbf{x}_i, \mathbf{z}), \forall \mathbf{y} \in \underline{D_l}(\mathbf{x}_i, d_m), \\ &\quad \forall \mathbf{z} \in \underline{G_l}(\mathbf{x}_i, d_m) - \underline{D_l}(\mathbf{x}_i, d_m),\end{aligned}$$

we can conclude that

$$\mathbf{y}_i \in \bigcup_{l=1}^{k_m} \overline{D_l}(\mathbf{x}_i, d_m)$$

and

$$d(\mathbf{x}_i, \mathbf{y}_i) \le \min_{\mathbf{y} \in \bigcup_{l=1}^{k_m} \underline{D}(\mathbf{x}_i, d_m)} d(\mathbf{x}_i, \mathbf{y}).$$

The above-mentioned expressions represent the contributions of $\overline{D_l}(\mathbf{x}_i, d_m)$ and $\underline{D}_l(\mathbf{x}_i, d_m)$ to the search for $\mathbf{y}_i$. We can find $\mathbf{y}_i$ in $\overline{D_l}(\mathbf{x}_i, d_m)$, whose size is smaller than that of $\overline{G_l}(\mathbf{x}_i, d_m)$. $\underline{D}_l(\mathbf{x}_i, d_m)$ can be used to update the d_m value. We know that the size of $G_l(\mathbf{x}_i, d_m)$ depends on d_m. If $d_{m_1} < d_{m_2}$, then $G_l(\mathbf{x}_i, d_{m_1}) \subseteq G_l(\mathbf{x}_i, d_{m_2})$, $\overline{G_l}(\mathbf{x}_i, d_{m_1}) \subseteq \overline{G_l}(\mathbf{x}_i, d_{m_2})$, and $\underline{G_l}(\mathbf{x}_i, d_{m_1}) \subseteq \underline{G_l}(\mathbf{x}_i, d_{m_2})$. Thus, we will constantly update the d_m value in the process of finding $\mathbf{y}_i$ in $\overline{D_l}(\mathbf{x}_i, d_m)$. Before searching for it, if $\underline{D}_l(\mathbf{x}_i, d_m) \ne \emptyset$, we update

$$d_m = \min_{\mathbf{y} \in \underline{D_l}(\mathbf{x}_i, d_m)} \min\{d(\mathbf{x}_i, c_l) + d(\mathbf{y}, c_l), d(\mathbf{x}_i, c_h) + d(\mathbf{y}, c_h)\}.$$

In the search process, if we find a point whose distance from $\mathbf{x}_i$ is less than d_m, we update d_m with this distance. Thus, we can continuously reduce the d_m value and the size of the search space.

The accelerated CFSFDP algorithm is called CFSFDP+A, and it is described in Algorithm 1 . The selection of k_m initial cluster centers does not affect the clustering result of the CFSFDP+A algorithm but affects its efficiency. As the k_m value increases, most of the points in each subset partitioned by the k-means algorithm are more adjacent to each other in a local geometrical space.

Space complexity: In the CFSFDP+A algorithm, we need to save a partition vector $Pc = [p_1, p_2, \cdots, p_n]$, where $p_i = l$ if the point $\mathbf{x}_i$ belongs to S_l, and the distance matrix $D = [d(\mathbf{x}_i, \mathbf{v}_l)]_{n \times k_m}$. This procedure requires $O(nk_m)$ space. Given that $k_m \ll n$, $nk_m \ll n^2$.

Time complexity: First, we apply the k-means algorithm to partition the data set into k_m subsets, which requires $O(nk_mt)$ distance calculations, where t is the number of iterations. In this procedure, the center $\mathbf{v}_l$ of each subset $S_l (1 \le l \le k_m)$ and the distances between each point $\mathbf{x}_i (1 \le i \le n)$ and all the centers can be obtained. In the assignment of points to the clusters, $O(nn_1 + nn_2)$ distance calculations are required, where $n_1 (\ll n)$ is the average number of distance calculations in computing the densities of all the points and $n_2 (\ll n)$ is the average number of distance calculations in computing the separations of all the points. Therefore, the proposed algorithm requires $O(nk_mt + nn_1 + nn_2)$ distance calculations. Given that the computational complexity of the original algorithm is $O(n^2)$ in terms of the number of distance calculations, we may conclude that the proposed algorithm has lower computational complexity.

Algorithm 1: CFSFDP+A algorithm.

Input: $\mathbf{X}$, d_c, k_m
Output: A clustering result
Obtain an initial partition $\mathbf{S}$ by k-means;
Save $\mathbf{V}$ and $\mathbf{D} = [d(\mathbf{x}_i, \mathbf{v}_l)]_{n\times k_m}$;
for $i = 1 : n$ **do**
  $F(\mathbf{x}_i) = \arg(\mathbf{x}_i)$// It will be used to save the name of the point that $\mathbf{x}_i$ follows;
for $i = 1 : n$ **do**
  for $l = 1 : k_m$ **do**
    Compute $Bn_l(\mathbf{x}_i)$ and $\underline{B}_l(\mathbf{x}_i)$;
    $\rho(\mathbf{x}_i) = \rho(\mathbf{x}_i) + |\underline{B}_l(\mathbf{x}_i)|$;
    for *each* $\mathbf{z}_i \in Bn_l(\mathbf{x}_i)$ **do**
      Compute $d(\mathbf{x}_i, \mathbf{z}_i)$;
      if $d(\mathbf{x}_i, \mathbf{z}_i) < d_c$ **then**
        $\rho(\mathbf{x}_i) = \rho(\mathbf{x}_i) + 1$;
for $i = 1 : n$ **do**
  Randomly select $\mathbf{z}_i$ where $\rho(\mathbf{z}_i) > \rho(\mathbf{x}_i)$;
  Set $d_m = \min\{d(\mathbf{x}_i, c_r) + d(\mathbf{z}_i, c_r), d(\mathbf{x}_i, c_h) + d(\mathbf{z}_i, c_h)\}$, $\mathbf{z}_i \in c_r, \mathbf{x}_i \in c_h$;
  for $l = 1 : k_m$ **do**
    Compute $\overline{D}_l(\mathbf{x}_i, d_m)$ and $\underline{D}_l(\mathbf{x}_i, d_m)$;
    if $\underline{D}_l(\mathbf{x}_i, d_m) \neq \emptyset$ **then**
      Update d_m;
    for *each* $\mathbf{z}_i \in \overline{D}_l(\mathbf{x}_i, d_m)$ **do**
      if $min\{d(\mathbf{x}_i, c_l) + d(\mathbf{z}_i, c_l), d(\mathbf{x}_i, c_h) + d(\mathbf{z}_i, c_h)\} \leq d_m$ **then**
        Compute $d(\mathbf{x}_i, \mathbf{z}_i)$;
        if $d(\mathbf{x}_i, \mathbf{z}_i) < d_m$ **then**
          $d_m = d(\mathbf{x}_i, \mathbf{z}_i)$;
          $F(\mathbf{x}_i) = \arg(\mathbf{z}_i)$;
          $\delta(\mathbf{x}_i) = d_m$;
Determine the number of clusters k;
Select a set of cluster centers including the first k points with higher ρ and δ;
Assign each remaining point $\mathbf{x}_i$ to the same cluster as its nearest neighbor with higher density;

4. Approximate CFSFDP algorithm

Although the CFSFDP+A algorithm reduces the computational cost of the original algorithm, it still involves a large number of distance calculations in the case of a large-scale data set. To further enhance the scalability of the original algorithm, we propose an approximate CFSFDP algorithm based on exemplar clustering. In this algorithm, the centers $\mathbf{V}$ obtained by the k-means algorithm are selected as exemplars to replace all the points. The key issue is how to simulate the clustering result of the CFSFDP algorithm by only computing the densities and separations of these exemplars.

We use the density measure ρ of the original algorithm to evaluate the density of an exemplar. The separation measure δ of an exemplar is different from that in the original algorithm; hence, it needs to be redefined. The original algorithm assumes that a point $\mathbf{x}_i$ may be assigned to the same cluster as its nearest point with higher local density. Thus, the algorithm uses their distance to represent the separation. However, there may be several points between any two exemplars. Therefore, before evaluating the separation of an exemplar, we need to consider the density connection between two exemplars (see Fig. 2). The density connection is an important factor that is used to determine whether the two exemplars represent the same cluster. We assume that only density-connected exemplars maybe in the same cluster. The density connection can be defined as follows:

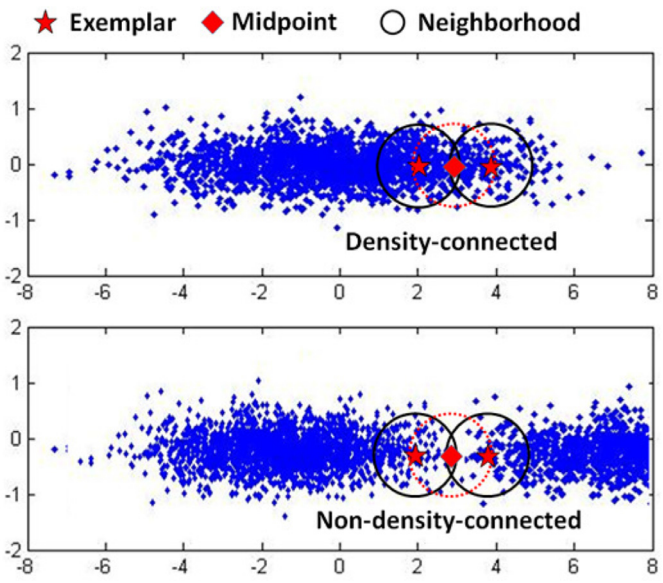

Fig. 2. Density-connectivity between exemplars.

Definition 1. For any two exemplars $\mathbf{v}_i$ and $\mathbf{v}_j$, if they are within the neighborhoods of each other, i.e., $d(\mathbf{v}_i, \mathbf{v}_j) < d_c$, they are assumed to be directly density-connected.

Definition 2. For any two exemplars $\mathbf{v}_i$ and $\mathbf{v}_j$, if $d_c \leq d(\mathbf{v}_i, \mathbf{v}_h) < 2d_c$ and the density of their midpoint $\frac{\mathbf{v}_i+\mathbf{v}_j}{2}$ is no less than their minimum density, they are assumed to be indirectly density-connected.

Definition 1 is an assumption of the approximate algorithm with regard to the density connection. Here, d_c denotes the error tolerance of the density connection. For any two exemplars, if their distance is less than d_c, they are density-connected by default. Otherwise, we need to evaluate their connection, according to Definition 2. For two exemplars, if their distance is in the internal $[d_c, 2d_c)$, their neighborhoods or borders are overlapped. In this case, we introduce the midpoint between the two exemplars as a latent exemplar. The midpoint and each exemplar are directly density-connected. To judge whether they are density-connected, we need to consider the density of their midpoint. In the original algorithm, a cluster is defined to have a center with higher local density and separation, and it is surrounded by neighbors with lower local density. This implies that the density of points decreases from the center to the borders in a cluster. If the two exemplars whose midpoint has lower density than that of the exemplars are assumed to represent the same cluster, the assumption does not fit the definition of a cluster in the original algorithm. Therefore, we require that if two exemplars are indirectly density-connected, the density of their midpoint should be no less than their minimum density. Therefore, we present a formal description of the density connection as follows.

$$\zeta(\mathbf{v}_i, \mathbf{v}_j) = \begin{cases} 'cn', & if \quad d(\mathbf{v}_i, \mathbf{v}_j) < d_c, \\ 'icn', & if \quad d_c \leq d(\mathbf{v}_i, \mathbf{v}_j) < d_c \\ & \wedge \rho(\frac{\mathbf{v}_i + \mathbf{v}_j}{2}) \geq \min(\rho(\mathbf{v}_i), \rho(\mathbf{v}_i)), \\ 'non', & otherwise, \end{cases}$$

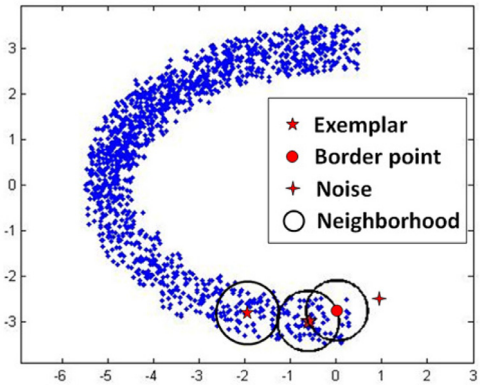

Fig. 3. Assignation of a point.

where $'cn'$ denotes direct connection, $'icn'$ denotes indirect connection, and $'non'$ denotes non-connection. Here, while computing the density of the midpoint between two exemplars, we need not directly compute the distances between the points and the midpoint, owing to the triangle median theorem:

$$d(\mathbf{x}_q, \frac{\mathbf{v}_i+\mathbf{v}_j}{2}) = \sqrt{\frac{1}{2}d^2(\mathbf{x}_q,\mathbf{v}_i)+\frac{1}{2}d^2(\mathbf{x}_q,\mathbf{v}_j)-\frac{1}{4}d^2(\mathbf{v}_i,\mathbf{v}_j)}.$$

On the basis of the above-mentioned definitions, for an exemplar, we will use the distance between it and its nearest exemplar with higher density and density connection to represent its separation. Therefore, the separation measure is defined as follows:

$$\delta(\mathbf{v}_i) = \begin{cases} \min\limits_{\rho(\mathbf{v}_j)>\rho(\mathbf{v}_i)\wedge\zeta(\mathbf{v}_i,\mathbf{v}_j)\neq' non'} d(\mathbf{v}_i,\mathbf{v}_j), \\ \max\limits_{\mathbf{v}_j\in\mathbf{V}} d(\mathbf{v}_i,\mathbf{v}_j), otherwise. \end{cases} \quad (3)$$

After the exemplars are organized, we will assign the points to clusters by their minimum distance with exemplars. For a point, if the distance between it and its nearest exemplar is less than $2d_c$, it is in the neighborhood of the border point that the exemplar represents. Thus, it is assigned to the cluster that the exemplar represents. Otherwise, the point is assumed as noise (see Fig. 3).

The density-exemplar algorithm (CFSFDP+DE) is described in Algorithm 2. The random selection of k_m initial cluster centers does not affect the clustering efficiency of the algorithm but affects its clustering effectiveness.

Algorithm 2: CFSFDP+DE algorithm.

Input: $\mathbf{X}$, d_c, k_m
Output: A clustering result
Obtain an initial partition $\mathbf{S}$ by k-means with the radius constraint;
Save $\mathbf{V}$ and $\mathbf{D} = [d(\mathbf{x}_i,\mathbf{v}_l)]_{n\times k_m}$;
for $i = 1 : n$ **do**
  $F(\mathbf{v}_l) = \arg(\mathbf{v}_l)$// It will be used to save the name of the exemplar that $\mathbf{v}_l$ follows;
Compute $\rho(\mathbf{v}_l)$ for each exemplar $\mathbf{v}_l$;
Compute $\delta(\mathbf{v}_l)$ and determine $F(\mathbf{v}_l)$ for each exemplar $\mathbf{v}_l$;
Determine the number of clusters k;
Select the first k exemplars with higher ρ and δ as cluster centers;
Assign each remaining $\mathbf{v}_l$ to the same cluster as its nearest exemplar with higher density and density connection;
for $i = 1 : n$ **do**
  Compute $\mathbf{v}_l = \arg\min_{\mathbf{v}_l\in\mathbf{V}} d(\mathbf{x}_i,\mathbf{v}_l)$;
  if $d(\mathbf{x}_i,\mathbf{v}_l) < 2d_c$ **then**
    Assign $\mathbf{x}_i$ to the cluster that $\mathbf{v}_l$ represents;

The space complexity of the algorithm is the same as that of the CFSFDP+A algorithm. The time complexity is $O(nk_mt + k_m^2)$. Since $k_m \ll n$, the CFSFDP+DE algorithm offers greater scalability in clustering large-scale data sets than the CFSFDP and CFSFDP+A algorithms.

5. Experimental analysis

5.1. Experimental environment

In this section, we verify the effectiveness and efficiency of the proposed algorithms on several synthetic and real data sets (Table 1) that were downloaded from [37,38]. Figs. 4 and 5 show the cluster distribution of these synthetic two-dimensional data sets.

The experiments were conducted on an Intel i7-4710MQ computer with 16G RAM and MATLAB 2012b. In the experiments, we used the maximum possible number of clusters to set k_m for CFSFDP+A and CFSFDP+DE. In the literature [39,40], many scholars have indicated that there are few guidelines for setting k_m; a rule of thumb used in many studies is $k_m \le \sqrt{n}$. In [41], the authors have provided a theoretical explanation for this rule. Thus, we set $k_m = \lfloor\sqrt{n}\rfloor$. For the CFSFDP, CFSFDP+A, and CFSFDP+DE algorithms, we selected the first k points or exemplars with higher density and separation as the cluster centers, where k was set by considering the number of classes in the data set.

5.2. Clustering effectiveness

In the effectiveness analysis, we compared the clustering performance of the DBSCAN, OPTICS, CFSFDP, DGB, CFSFDP+A, and CFSFDP+DE algorithms on several data sets. We considered four validity measures [42,43], namely, accuracy (AC), precision (PE), adjusted rand index (ARI) and normalized mutual information (NMI), to evaluate the clustering effectiveness of the algorithms. Let $\mathbf{X}$ be a data set, let $C = \{C_1, C_2, \cdots, C_k\}$ be a clustering result of $\mathbf{X}$, let $P = \{P_1, P_2, \cdots, P_k\}$ be a partition of the original classes in $\mathbf{X}$, let n_{ij} be the number of common points of groups C_i and P_j: $n_{ij} = |C_i \cap P_j|$, let b_i be the number of points in C_i, and let d_j be the number of points in P_j. The validity measures are defined as

$$AC = \frac{1}{n}\sum_{i=1}^{k}\max_{j=1}^{k} n_{ij},$$

$$PE = \frac{1}{k}\sum_{i=1}^{k}\frac{\max_{j=1}^{k} n_{ij}}{b_j},$$

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{b_i}{2}\sum_j \binom{d_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{b_i}{2} + \sum_j \binom{d_j}{2}] - [\sum_i \binom{b_i}{2}\sum_j \binom{d_j}{2}]/\binom{n}{2}},$$

$$NMI = \frac{2\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{n_{ij}}{n}\log\frac{n_{ij}n}{b_i d_j}}{\sum_{i=1}^{k} -\frac{b_i}{n}\log\frac{b_i}{n} + \sum_{j=1}^{k} -\frac{d_j}{n}\log\frac{d_j}{n}}.$$

The DBSCAN, OPTICS, CFSFDP, CFSFDP+A, and CFSFDP+DE algorithms are required to input the parameter d_c. We estimated the d_c value using $\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d(\mathbf{x}_i, = \mathbf{x})$, where $= \mathbf{x} = \sum_{j=1}^{n}\frac{\mathbf{x}_j}{n}$, and we used the Euclidean distance as d. However, these algorithms may need different d_c values. Thus, we tested each of these algorithms with 10 different d_c values, i.e., $d_c = \bar{d}, \bar{d}/2, \bar{d}/3, \bar{d}/4, \bar{d}/5, \bar{d}/6, \bar{d}/7, \bar{d}/8, \bar{d}/9, \bar{d}/10$, and we selected the best clustering result for comparison. The DGB algorithm requires

Table 1
Description of data sets.

Type	Data set	Points	Clusters	Dimensions
Synthetic two-dimensional data	Curve	2,000	2	2
	Aggregation	788	7	2
	Flame	240	2	2
	S1	5,000	15	2
Synthetic high-dimensional data	DIM1	1024	16	32/64/128/256/512/1024
	DIM2	20,000	4	32/64/128/256/512/1024
UCI data	Statlog	6435	7	36
	Handwritten Digits	5,620	10	64
	Shuttle	58,000	7	9
	KDD-CUP'99	1,048,576	3	40

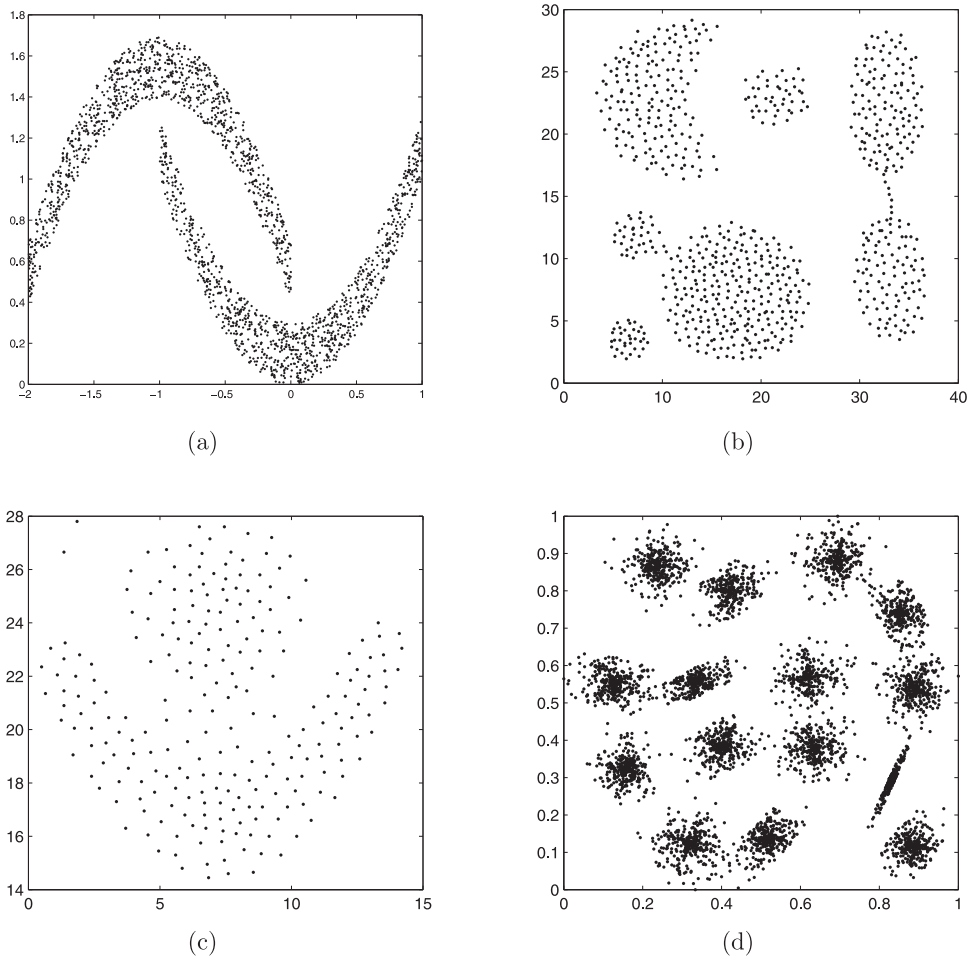

Fig. 4. Distributions of four synthetic data: (a) Curve. (b) Aggregation. (c) Flame. (d) S1.

several parameters. According to the suggestion of the authors, we set the number of grids to 22, the grid length to 1.1, and the best cutoff factor from the interval [0.1, 0.5]. Because the DGB algorithm can only cluster two-dimensional data, we compared the proposed algorithms with it on such data sets.

Tables 2 and 3 summarize the clustering accuracies of these algorithms on the synthetic two-dimensional and UCI data sets in terms of the indices AC, PE, ARI, and NMI. According to these results, we can see that (1) CFSFDP is superior to DBSCAN and OPTICS on most of the tested data sets; (2) the clustering results of CFSFDP+A are the same as those of CFSFDP; (3) although the clustering results of CFSFDP+DE are affected by different initial cluster centers, its average clustering results are close to those of CFSFDP; (4) DGB loses its clustering accuracy to some extent compared to CFSFDP.

5.3. Clustering efficiency

Firstly, we tested the clustering speeds of DBSCAN, OPTICS, and CFSFDP with the KD-Tree accelerating index structure. Furthermore, we compared their clustering efficiency with that of the proposed algorithms CFSFDP+A and CFSFDP+DE. We considered two indices, i.e., the running time (s) and number of distance calculations, to evaluate the clustering efficiency of different algorithms. Note that for a large-scale data set, some algorithms cannot obtain the clustering results within an acceptable time. Therefore, in our experiments, if the running time of an algorithm was longer

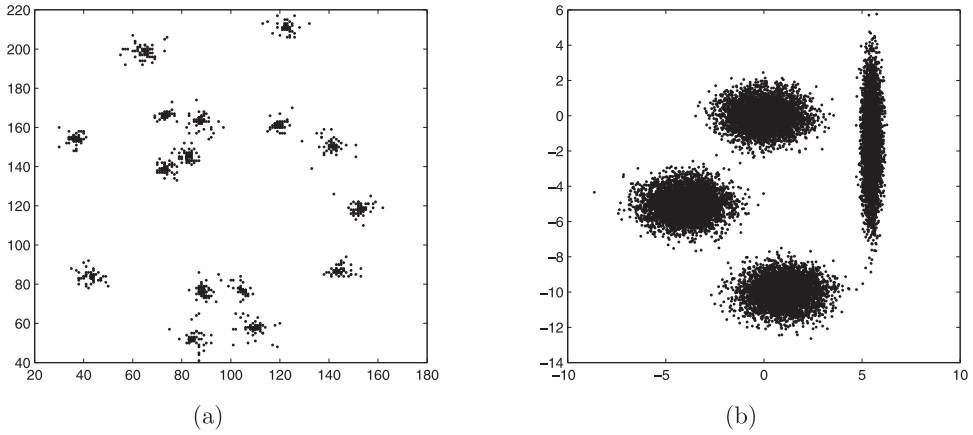

Fig. 5. (a) Two-dimensional distribution of the DIM1 data. (b) Two-dimensional distribution of the DIM2 data.

Table 2
Clustering effectiveness of algorithms on synthetic two-dimensional data sets.

Data set	Algorithm	AC	PE	ARI	NMI
Curve	DBSCAN	1.0000	1.0000	1.0000	1.0000
	OPTICS	1.0000	1.0000	1.0000	1.0000
	CFSFDP	1.0000	1.0000	1.0000	1.0000
	CGB	1.0000	1.0000	1.0000	1.0000
	CFSFDP+A	1.0000	1.0000	1.0000	1.0000
	CFSFDP+DE	0.9985	0.9985	0.9940	0.9911
Aggregation	DBSCAN	0.8274	0.8899	0.8089	0.8894
	OPTICS	0.8274	0.8899	0.8089	0.8894
	CFSFDP	0.9937	0.9880	0.9876	0.9823
	CGB	0.9898	0.9571	0.9706	0.9643
	CFSFDP+A	0.9937	0.9880	0.9876	0.9823
	CFSFDP+DE	0.9894	0.9863	0.9664	0.9767
Flame	DBSCAN	0.6458	0.8214	0.0128	0.0242
	OPTICS	0.6458	0.8214	0.0128	0.0242
	CFSFDP	0.9917	0.9935	0.9666	0.9355
	CGB	0.9833	0.9342	0.9120	0.8241
	CFSFDP+A	0.9917	0.9935	0.9666	0.9355
	CFSFDP+DE	0.9204	0.9403	0.7514	0.7369
S1	DBSCAN	0.9350	0.9429	0.8846	0.9167
	OPTICS	0.9350	0.9429	0.8846	0.9167
	CFSFDP	0.9928	0.9930	0.9846	0.9851
	CGB	0.9524	0.9439	0.9186	0.9336
	CFSFDP+A	0.9928	0.9930	0.9846	0.9851
	CFSFDP+DE	0.9610	0.9760	0.9486	0.9751

Table 3
Clustering effectiveness of algorithms on UCI data sets.

Data set	Algorithm	AC	PE	ARI	NMI
Statlog	DBSCAN	0.4437	0.8769	0.1463	0.3327
	OPTICS	0.4437	0.8769	0.1463	0.3327
	CFSFDP	0.7616	0.8233	0.5749	0.6202
	CFSFDP+A	0.7616	0.8233	0.5749	0.6202
	CFSFDP+DE	0.7476	0.7732	0.5736	0.6033
Digits	DBSCAN	0.7141	0.8934	0.5052	0.7163
	OPTICS	0.7141	0.8934	0.5052	0.7163
	CFSFDP	0.8041	0.8688	0.7584	0.8645
	CFSFDP+A	0.8041	0.8688	0.7584	0.8645
	CFSFDP+DE	0.7946	0.8505	0.7429	0.8569
Shuttle	DBSCAN	0.9992	0.9506	0.9677	0.8917
	OPTICS	0.9992	0.9506	0.9677	0.8917
	CFSFDP	0.9999	0.9995	0.9998	0.9985
	CFSFDP+A	0.9999	0.9995	0.9998	0.9985
	CFSFDP+DE	0.9692	0.9702	0.9692	0.9599
KDD-CUP'99	DBSCAN	0.9918	0.9869	0.5487	0.5662
	OPTICS	0.9921	0.9882	0.5739	0.5822
	CFSFDP	0.9885	0.9925	0.7140	0.7365
	CFSFDP+A	0.9885	0.9925	0.7140	0.7365
	CFSFDP+DE	0.9868	0.9913	0.7112	0.7327

than 24 h, we terminated it and set the values of the two indices as 'NA'. Further, we used the built-in function KD-Tree in MATLAB to accelerate the DBSCAN, OPTICS, and CFSFDP algorithms. Thus, we could not count their numbers of distance calculations. We set the index values of these algorithms as 'NA'. The selection of k_m initial cluster centers affects the efficiency of the CFSFDP+A algorithm and the effectiveness of the CFSFDP+DE algorithm. Therefore, we randomly produced 20 sets of k_m initial cluster centers for each data set and computed their average values for each of the above-mentioned indices.

Tables 4 and 5 list the clustering times and the numbers of computed distances for different algorithms on the given data sets. According to these results, we can see that (1) the improvement in the efficiency of some algorithms owing to the KD-Tree index structure is not obvious, because the index structure involves additional computational costs and is not suitable for high-dimensional data; (2) the clustering speed of DGB is lower than that of CFSFDP+A on the tested data sets, except S1; (3) CFSFDP+A can reduce a number of distance calculations while retaining the same clustering results as CFSFDP; and (4) CFSFDP+DE is the most efficient algorithm.

Furthermore, we tested CFSFDP, CFSFDP+KD, DBSCAN+KD, OPTICS+KD, CFSFDP+A, and CFSFDP+DE on the KDD-CUP'99 data set. Fig. 6 shows the clustering speeds of different algorithms on the KDD-CUP'99 data set with different numbers of points (no more than 10,000). According to the figure, we can see that the efficiency of the proposed algorithms increases with the number of points. Tables 6 and 7 summarize the clustering efficiency of different algorithms on the KDD-CUP'99 data set with different numbers of points (no less than 100,000). We can see that only the proposed algorithms can deal with the data set including more than 100,000 points within an acceptable time. Moreover, when the number of points reaches 800,000, only the CFSFDP+A algorithm is applicable.

Finally, to show the effect of the number of dimensions on the efficiency of the proposed algorithms, we tested the scalability with different numbers of dimensions (32, 64, 128, 256, 512, and 1024) on the two synthetic data sets DIM1 and DIM2. The DIM1 data set has been used in [44]. The two-dimensional distributions of these data are shown in Fig. 5. DIM1 with different dimensions includes the same points (n = 1024) and Gaussian clusters (k = 16). Similarly, DIM2 with different dimensions includes the same

Table 4
Clustering efficiency of algorithms on synthetic two-dimensional data sets.

Data set	Algorithm	# of Distances	Seconds
Curve	CFSFDP	3,998,000	0.3510
	CFSFDP+KD	NA	0.5170
	DBSCAN+KD	NA	1.0150
	OPTICS+KD	NA	2.3450
	DGB	NA	0.3320
	CFSFDP+A	887,664	0.2090
	CFSFDP+DE	715,916	0.0240
Aggregation	CFSFDP	620,156	0.0650
	CFSFDP+KD	NA	0.1590
	DBSCAN+KD	NA	0.2440
	OPTICS+KD	NA	0.6790
	DGB	NA	0.0630
	CFSFDP+A	150,481	0.0480
	CFSFDP+DE	88,634	0.0200
Flame	CFSFDP	57,360	0.0040
	CFSFDP+KD	NA	0.0400
	DBSCAN+KD	NA	0.0810
	OPTICS+KD	NA	0.2490
	DGB	NA	0.0150
	CFSFDP+A	22,344	0.0080
	CFSFDP+DE	14,505	0.0040
S1	CFSFDP	24,995,000	2.1440
	CFSFDP+KD	NA	2.2460
	DBSCAN+KD	NA	3.3070
	OPTICS+KD	NA	5.3550
	CFSFDP+KD	NA	2.2460
	DGB	NA	0.2300
	CFSFDP+A	2,624,762	1.1650
	CFSFDP+DE	1,402,415	0.0470

Table 5
Clustering efficiency of algorithms on UCI data sets.

Data set	Algorithm	# of Distances	Seconds
Statlog	CFSFDP	41,402,790	27.9840
	CFSFDP+KD	NA	24.9990
	DBSCAN+KD	NA	7.8180
	OPTICS+KD	NA	15.5670
	CFSFDP+A	8,517,991	4.9670
	CFSFDP+DE	2,062,360	0.2890
Digits	CFSFDP	31,578,780	64.5790
	CFSFDP+KD	NA	64.4760
	DBSCAN+KD	NA	18.8570
	OPTICS+KD	NA	45.6120
	CFSFDP+A	14,601,750	33.9130
	CFSFDP+DE	1,666,221	0.3800
Shuttle	CFSFDP	3,363,900,000	1,045.5000
	CFSFDP+KD	NA	1,000.5000
	DBSCAN+KD	NA	451.6100
	OPTICS+KD	NA	1,181.6000
	CFSFDP+A	205,107,293	371.3650
	CFSFDP+DE	55,708,680	2.1050

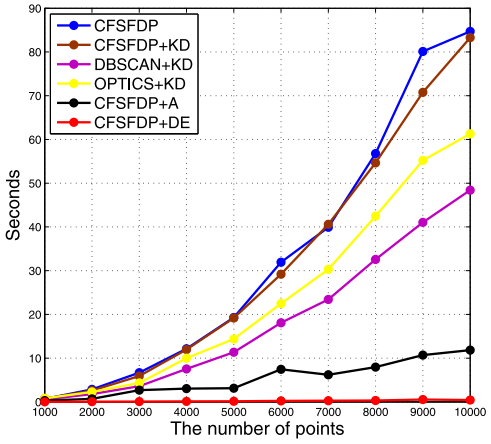

Fig. 6. Computational times for different numbers of points on the KDD-CUP'99 data set.

points (n = 20,000) and Gaussian clusters (k = 4). Fig. 7(a) shows that the CFSFDP+A and CFSFDP+DE algorithms exhibit better scalability with increasing dimensions, compared with the original algorithm. However, only a slight difference is observed in the efficiencies of the two proposed algorithms on the DIM1 data set, because it only includes 1024 points. Fig. 7(b) shows that the CFSFDP+DE algorithm is obviously faster than the CFSFDP+A algorithm on the data set including 20,000 points. Thus, the experimental results indicate that the efficiency of the CFSFDP+DE algorithm increases with the size of the data set.

6. Conclusions

In this paper, we proposed two new algorithms based on the *k*-means algorithm in order to enhance the scalability of the CFSFDP algorithm. The first algorithm is the CFSFDP+A algorithm, which uses an acceleration mechanism based on concept approximation. The CFSFDP+A algorithm can rapidly obtain the same clustering results as the original algorithm. The second algorithm is the density-exemplar clustering algorithm (CFSFDP+DE), which uses exemplar clustering to obtain approximate clustering results of the original algorithm. The CFSFDP+DE algorithm can rapidly cluster a large-scale data set while ensuring only a slight loss in precision. We tested the two proposed algorithms on several synthetic and real data sets. The experimental results showed that the CFSFDP+A algorithm requires a shorter computational time and fewer distance calculations while retaining the same clustering results as the original algorithm, and the CFSFDP+DE algorithm can obtain the approximate clustering results in the shortest time and with the fewest distance calculations, compared to the CFSFDP and CFSFDP+A algorithms.

Table 6
Clustering time (s) of algorithms on the KDD-CUP'99 data set.

Algorithm	n = 100,000	n = 300,000	n = 500,000	n = 700,000	n = 900,000	n = 1,040,000
CFSFDP	11,342.02	NA	NA	NA	NA	NA
CFSFDP+KD	13,791.22	NA	NA	NA	NA	NA
DBSCAN+KD	5,962.23	NA	NA	NA	NA	NA
OPTICS+KD	6,482.13	NA	NA	NA	NA	NA
CFSFDP+A	1,425.31	12,946.14	38,373.85	70,190.53	NA	NA
CFSFDP+DE	9.09	37.66	99.44	217.13	337.72	495.26

Table 7
Number of distance calculation of algorithms on the KDD-CUP'99 data set.

Algorithm	n = 100,000	n = 300,000	n = 500,000	n = 700,000	n = 900,000	n = 1,040,000
CFSFDP	9,999,900,000	NA	NA	NA	NA	NA
CFSFDP+KD	NA	NA	NA	NA	NA	NA
DBSCAN+KD	NA	NA	NA	NA	NA	NA
OPTICS+KD	NA	NA	NA	NA	NA	NA
CFSFDP+A	631,269,045	4,819,950,480	12,450,461,789	23,890,000,051	NA	NA
CFSFDP+DE	158,049,770	820,649,331	1,767,749,571	2,926,349,030	4,266,448,878	5,299,318,671

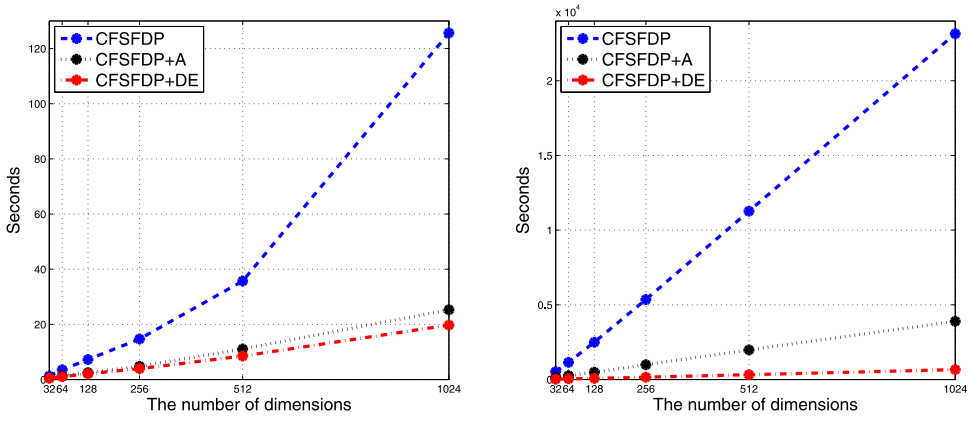

Fig. 7. (a) Computational times for different numbers of dimensions on the DIM1 data. (b) Computational times for different numbers of dimensions on the DIM2 data.

Acknowledgment

The authors are very grateful to the editors and reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (Nos. 61305073, 61432011, 61573229, U1435212, 61403238), the National Key Basic Research and Development Program of China (973)(Nos. 2013CB329404, 2014CB340400), the Foundation of Doctoral Program Research of Ministry of Education of China(No. 20131401120001), the Technology Research Development Projects of Shanxi (Nos. 2015021100, 201601D202036), and Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi (No. 2015107).

References

[1] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2001.
[2] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice Hall, 1988.
[3] A.K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[4] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, vol. 1, University of California Press, 1967, pp. 281–297.
[5] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.
[6] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. R. Stat. Soc. 39 (1) (1977) 1–38.
[7] L. Kaufman, R.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley&Sons, 1990.
[8] F. Camastra, A. Verri, A novel kernel method for clustering, IEEE Trans. Pattern Anal. Mach.Intell. 27 (5) (2005) 801–805.
[9] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, in: SIGMOD Conference, 1996, pp. 103–114.
[10] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for clustering large databases, in: Proceedings of the Symposium on Management of Data (SIGMOD), 1998, pp. 73–84.
[11] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Comput. 32 (8) (1999) 68–75.
[12] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: E. Simoudis, J. Han, U. Fayyad (Eds.), Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press, 1996, pp. 226–231.
[13] M. Ankerst, M. Breunig, H.P. Kriegel, OPTICS: ordering points to identify the clustering structure, in: Proceedings of International Conference on Management of Data(SIGMOD99), Philadelphia, PA, 1999, pp. 49–60.
[14] A. Hinneburg, D. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York, 1998, pp. 58–65.
[15] W. Wang, J. Yang, R. Muntz, STING: a statistical information grid approach to spatial data mining, in: Proceedings of the 23rd International Conference on Very Large Data Bases, Athens: Morgan Kaufmann, 1997, pp. 18–195.
[16] G. Sheikholeslami, S. Chatterjee, A.D. Zhang, Wavecluster: a multi-resolution clustering approach for very large spatial databases, in: A. Gupta, O. Shmueli, J. Widom (Eds.), Proceedings of the 24th International Conference on Very Large Data Bases, Morgan Kaufmann, New York, 1998, pp. 428–439.
[17] Y. Andrew, M.I. Ng, Y. Jordan, On spectral clustering: analysis and an algorithm, Adv. Neural Inf. Process. Syst. 14 (2001) 849–856.
[18] K. Fukunaga, L. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Trans. Inf. Theory 21 (1) (1975) 32–40.
[19] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Anal. Mach. Intell. 17 (8) (1995) 790–799.
[20] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[21] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[22] N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger, The r*-tree: an efficient and robust access method for points and rectangles, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1990, pp. 322–331.
[23] S. Berchtold, D. Keim, H.P. Kriegel, The x-tree: an efficient and robust access method for points and rectangles, in: Proceedings of International Conference on Very Large Data Bases, 1996, pp. 28–39.
[24] M. Dash, H. Liu, X. Xu, '1+1>2 merging distance and density based clustering, in: Proceedings of the 7th International Conference on Database Systems for Advanced Applications, IEEE, Hong Kong, 2001, pp. 32–39.
[25] P. Viswanath, R. Pinkesh, L-DBSCAN : a fast hybrid density based clustering method, in: Proceedings of 18th International Conference on Pattern Recognition, 2006, pp. 912–915.
[26] P. Viswanath, V.S. Babu, Rough-DBSCAN: a fast hybrid density based clustering method for large data sets, Pattern Recognit. Lett. 30 (2009) 1477–1488.
[27] S.J. Nanda, G. Panda, Design of computationally efficient density-based clustering algorithms, Data Knowl. Eng. 95 (2015) 23–38.
[28] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, in: OSDI, 2004, pp. 137–150.
[29] Y. He, H. Tan, W. Luo, et al., MR-DBSCAN: an efficient parallel density-based clustering algorithm using mapreduce, in: IEEE 17th International Conference on Parallel and Distributed Systems, 2011.
[30] Y. Kim, K. Shim, M. Kim, J. Lee, DBCURE-MR: an efficient density-based clustering algorithm for large data using mapreduce, Inf. Syst. 42 (2014) 14–35.

[31] W. Chen, Y. Song, H. Bai, C. Lin, E. Chang, Parallel spectral clustering in distributed systems, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 568–586.
[32] B. Wu, B.M. Wilamowski, A fast density and grid based clustering method for data with arbitrary shapes and noise, IEEE Trans. Ind.Inf. (2016), doi:10.1109/TII.2016.2628747.
[33] Y. Zhang, S. Chen, G. Yu, Efficient distributed density peaks for clustering large data sets in mapreduce, IEEE Trans. Knowl. Data Eng. 28 (12) (2016) 3218–3230.
[34] J. Gao, L. Zhao, Z. Chen, P. Li, H. Xu, Y. Hu., ICFS: an improved fast search and find of density peaks clustering algorithm, in: Dependable, Autonomic and Secure Computing, 14th International Conference on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2016, pp. 537–543.
[35] H. Xiong, J.J. Wu, J. Chen, K-means clustering versus validation measures: adata-distribution perspective, IEEE Trans. Syst. Man, Cybern.-Part B 39 (2) (2009) 318–331.
[36] Z. Pawlak, Rough Sets-Theoretical Aspects of Reasoning about Data, Dordrecht, Boston, Kluwer Academic Publishers, London, 1991.
[37] Clustering datasets, 2015. http://cs.joensuu.fi/sipu/datasets.
[38] UCI machine learning repository, 2015. http://www.ics.uci.edu/mlearn/MLRepository.html.
[39] J.C. Bezdek, N.R. Pal, Some new indexes of cluster validity, IEEE Trans. Syst. Man Cybern.Part B 28 (3) (1998) 301–315.
[40] N.R. Pal, J.C. Bezdek, On cluster validity for the fuzzy c-means model, IEEE Trans. Fuzzy Syst. 3 (3) (1995) 370–379.
[41] J. Yu, Q. Cheng, The upper bound of the optimal number of clusters in fuzzy clustering, Sci. China Ser. F 44 (2) (2001) 119–125.
[42] Y.M. Yang, An evaluation of statistical approaches to text categorization, J. Inf. Retrieval 1 (1–2) (1999) 67–88.
[43] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1) (1985) 193–218.
[44] 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.

Liang Bai received his Ph.D. degree in Computer Science from Shanxi University in 2012. He is currently an Associate Professor with the School of Computer and Information Technology, Shanxi University. His research interest is in the areas of cluster analysis. He has published several journal papers in his research fields, including IEEE TPAMI, IEEE TKDE, DMKD, IEEE TFS, PR and INS and so on.

Xueqi Cheng is a professor at the Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS), and the director of the Research Center of Web Data Science & Engineering (WDSE) in ICT-CAS. His main research interests include network science, web search and data mining, big data processing and distributed computing architecture, and so on. He has published more than 100 publications in prestigious journals and conferences.

Jiye Liang received the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1990 and 2001, respectively. He is currently a Professor with the School of Computer and Information Technology and the Key Laboratory of Computational Intelligence and Chinese Information Processing of the Ministry of Education, Shanxi University, Taiyuan, China. His current research interests include computational intelligence, rough set theory, granular computing and so on. He has published more than 70 journal papers in his research fields, including AI, IEEE TPAMI, IEEE TKDE, IEEE TFS, DMKD, IEEE TSMC and so on.

Huawei Shen received the B.E. degree in electronic information from Xi'an Institute of Posts and Telecommunications, China, in 2003, and the Ph.D. degree in information security from the Institute of Computing Technology, the Chinese Academy of Sciences (CAS), China in 2010. Dr. Shen is currently an Associate Professor in Institute of Computing Technology, CAS. His major research interests include network science, information recommendation, user behaviour analysis, machine learning, and social network. He has published more than 20 papers in prestigious journals and top international conferences.

Yike Guo received the Ph.D. degree in logic and declarative programming from Imperial College, University of London. He is now a professor in computing science in the Department of Computing, Imperial College London. His research is in large scale scientific data analysis, data mining algorithms and applications, parallel algorithms, and cloud computing.