Soft Computing (2020) 24:17797–17819
https://doi.org/10.1007/s00500-020-05028-x

METHODOLOGIES AND APPLICATION

A two-stage density clustering algorithm

Min Wang[1] · Ying-Yi Zhang[1] · Fan Min[2,3] · Li-Ping Deng[4] · Lei Gao[2]

Published online: 26 May 2020

Abstract
Clustering by fast search and find of density peaks (CFDP) is a popular density-based algorithm. However, it is criticized because it is inefficient and applicable only to some types of data, and requires the manual setting of the key parameter. In this paper, we propose the two-stage density clustering algorithm, which takes advantage of granular computing to address the aforementioned issues. The new algorithm is highly efficient, adaptive to various types of data, and requires minimal parameter setting. The first stage uses the two-round-means algorithm to obtain $\sqrt{n}$ small blocks, where n is the number of instances. This stage decreases the data size directly from n to $\sqrt{n}$. The second stage constructs the master tree and obtains the final blocks. This stage borrows the structure of CFDP, while the cutoff distance parameter is not required. The time complexity of the algorithm is $O(mn^{\frac{3}{2}})$, which is lower than $O(mn^2)$ for CFDP. We report the results of some experiments performed on 21 datasets from various domains to compare a new clustering algorithm with some state-of-the-art clustering algorithms. The results demonstrated that the new algorithm is adaptive to different types of datasets. It is two or more orders of magnitude faster than CFDP.

Keywords Clustering · Density peak · Efficiency

1 Introduction

Clustering is a fundamental approach used to organize data into distinct groups to find intrinsic hidden patterns in the data. Its applications include image processing (Pappas 1992; Leong and Ong 2017), bioinformatics (Shuji et al. 2015), social networks (Chang et al. 2014) and pattern recognition (Guo et al. 2009). Popular clustering algorithms include partition-based (MacQueen et al. 1967; Kaufman 2008), density-based (Kriegel et al. 2011; Rodriguez and Laio 2014) and hierarchical (Johnson 1967; Dasgupta 2002). Partition-based clustering algorithms typically include the k-means (MacQueen et al. 1967) and k-medoid algorithms (Kaufman 2008). They construct a single partition of the dataset based on the distance between instances. Among them, the k-medoid algorithm always requires that each cluster center is an existing instance. By contrast, the k-means algorithm uses the average value of the cluster to build the virtual center. Because an instance is always assigned to the nearest center, these approaches are unable to detect non-spherical clusters.

Density clustering (Ester et al. 1996; Rodriguez and Laio 2014) performs well on data with circular, arc and some other irregular shapes. CFDP (Rodriguez and Laio 2014) has attracted much attention because of its simplicity and good results, and it can automatically find the clustering centers. However, it is criticized because it is inefficient and applicable only to some types of data, and requires the manual setting of the key parameter. First, the time complexity of the algorithm is $O(mn^2)$, where m and n are the number of attributes and instances, respectively. Hence, the algorithm is inapplicable to data with millions of instances. Second, for many datasets, the clustering results are often unsatisfactory in terms of purity, the Jaccard coefficient (JC), Fowlkes and Mallows index (FMI) and Rand index (RI). Third, the quality of the results depends substantially on cutoff distance thresh-

Communicated by V. Loia.

✉ Fan Min
minfan@swpu.edu.cn

[1] School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China

[2] School of Computer Science, Southwest Petroleum University, Chengdu 610500, China

[3] Institute for Artificial Intelligence, Southwest Petroleum University, Chengdu 610500, China

[4] School of Computer Science and Technology, China West Normal University, Nanchong 637002, China

old d_c. It is difficult, if not impossible, for the user to make an optimal setting in practice.

One solution to these issues is to introduce the granular computing (Hu et al. 2016; Li et al. 2016; Qian et al. 2015; Yao and Yao 2002; Chen et al. 2015) methodology. This methodology is widely used to manage many machine learning tasks, such as classification (Wang and Musa 2014), clustering (Wilderjans and Cariou 2016; Sarma et al. 2013), recommendation (Zhang et al. 2019, 2017), active learning (Wang et al. 2017), attribute reduction (Min et al. 2011; Li et al. 2011) and three-way decisions (Huang et al. 2017; Zhao et al. 2016a, b; Li et al. 2017; Yu et al. 2016; Liu and Liang 2017). For the clustering task, each instance can be considered as the finest granule, the entire dataset can be considered as the coarsest granule, and the clustering result has the granule level between the finest and coarsest. We may construct other granule levels to address the aforementioned issues.

In this paper, we propose the two-stage density clustering (TSD) algorithm, which is highly efficient, adaptive to various types of data, and requires minimal parameter setting. Figure 1 illustrates our new algorithm through a running example. Figure 1a describes a dataset of 100 instances. The first stage is pre-clustering, as shown in Fig. 1b. We divide the dataset into ten blocks and obtain ten virtual centers $[c_1, c_2, \ldots, c_{10}]$. Block size array $\rho = [17, 11, 9, 6, 12, 11, 9, 9, 6, 10]$ is obtained for the next stage. Pre-clustering ensures the local distribution of data and decreases the data size. The second stage is the density clustering of virtual centers, as shown in Fig. 1c. First, we obtain density ρ_i of each instance and calculate its minimum distance δ_i. Second, we construct the master tree according to (ρ_i, δ_i). Finally, we cluster ten virtual centers into three blocks and obtain their cluster indices $cl = [1, 1, 1, 3, 3, 2, 2, 2, 3, 3]$. Figure 1d shows the cluster results. All instances in each block have the same cluster indices as the virtual centers.

The granular computing methodology is used to design the TSD algorithm. In the pre-clustering stage, approximately $\sqrt{n}$ small local granules are obtained using a two-round-means subroutine, where n is the number of instances. This stage does not change the distribution of the data. In the density clustering stage, both the inner-granule size and inter-granule distance are used to construct the master tree. Then, local granules are accumulated to form the final clusters.

The two-stage density clustering (TSD) algorithm has the following advantages. First, the number of instances required for density clustering is reduced to $\sqrt{n}$. Thus, the time complexity is $O(mn^{\frac{3}{2}})$, which is much lower than $O(mn^2)$ for clustering by fast search and find of density peaks (CFDP). Second, the TSD algorithm has the advantages of the k-means algorithm and CFDP and has good adaptability. It has good clustering performance for datasets containing data of various types. Third, the density ρ is set to the number of instances in each block. The method is simple and effective and avoids manually setting the cutoff distance d_c. Therefore, the main problem solved is the efficiency and parameter setting in the CFDP algorithm. The new algorithm is $\sqrt{n}$ times faster than CFDP. It does not require the value of ρ to be set.

Experiments were performed on 21 datasets to quantify the performance of the TSD algorithm. These datasets were chosen from different applications, such as botany, materials science and games, with different data distributions. The largest dataset, Poker (Cattral and Oppacher 2007), contains 1,025,009 instances. We compared the TSD algorithm with five types of clustering algorithms: partition clustering (MacQueen et al. 1967), peak density clustering (Rodriguez and Laio 2014; Xie et al. 2016; Liu et al. 2018; Xu et al. 2016), maximum margin clustering (MMC) (Li et al. 2009), spectral clustering (Wang et al. 2011) and balanced clustering (Liu et al. 2017a). Four external evaluation functions were used to evaluate the clustering results. The time complexity was verified on seven large datasets. The experimental results demonstrated that the TSD algorithm had good clustering performance on various types of datasets. It was two or more orders of magnitude faster than CFDP.

The remainder of this paper is organized as follows: In Sect. 2, we review five types of clustering algorithms. In Sect. 3, we present the TSD clustering algorithm. We describe experiments on 21 datasets in Sect. 4. Finally, we draw a conclusion in Sect. 5.

2 Related work

In this section, we review five types of clustering algorithms: partition-based clustering (MacQueen et al. 1967; Kaufman 2008), density-based clustering (Kriegel et al. 2011; Rodriguez and Laio 2014; Xie et al. 2016; Liu et al. 2018; Xu et al. 2016), maximum margin clustering (MMC) (Li et al. 2009), spectral clustering (Chen and Cai 2011; Wang et al. 2011) and balanced clustering (Liu et al. 2017a).

2.1 Partition-based clustering

Partition-based clustering algorithms, such as the k-means (MacQueen et al. 1967) and k-medoid (Kaufman 2008) algorithms, are classic and efficient. The k-means algorithm calculates cluster centers iteratively as follows:

Step 1. Initialize k centers $c_1 \ldots c_k$ using random sampling.
Step 2. Each instance belongs to the block of the nearest center.

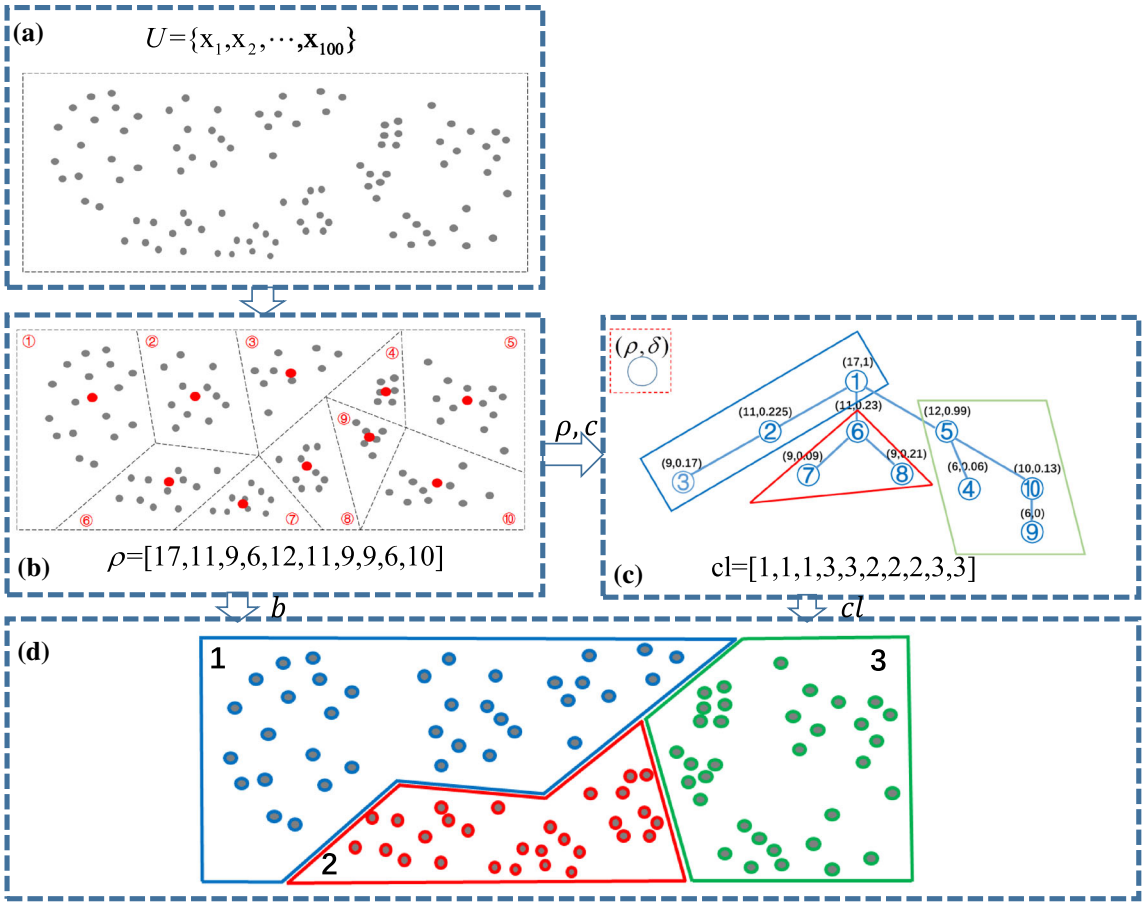


Fig. 1 Running example of the TSD algorithm. **a** describes the input, and **b** shows the first stage of TSD, namely pre-clustering. The pre-clustering stage obtains the virtual centers c and block density ρ, and **c** shows the second stage of TSD, namely the density clustering of virtual centers. Finally, **d** shows the cluster results

Step 3. Each new center takes the mean values of all instances of its block.
Step 4. Repeat Steps 2 and 3 until the cluster centers do not change.

Because the Euclidean distance is typically used as a similarity measure, partition-based clustering algorithms cannot detect non-spherical clusters.

2.2 Peak density clustering

Density clustering (Ester et al. 1996; Rodriguez and Laio 2014) explores clusters with different shapes based on the data density. DBSCAN (Ester et al. 1996) can find clusters with various shapes and manage noise. It controls class growth based on a density threshold. However, it does not perform well for overlapping densities.

Similar to DBSCAN, CFDP (Rodriguez and Laio 2014) aims to detect non-spherical clusters. Cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from instances with higher densities. For each instance i, CFDP computes two quantities: its density ρ_i and its minimum distance δ_i from instances of a higher density. Density ρ_i of instance i is defined as

$$\rho_i = \sum_j \chi(d_{ij} - d_c), \tag{1}$$

where $\chi(x) = 1$ if $x < 0$ and $\chi(x) = 0$ otherwise, and d_c is a cutoff distance. δ_i is measured by computing the minimum distance between instance i and any other instance with a higher density:

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}). \tag{2}$$

For the instance with the highest density, we conventionally take $\delta_i = \max(d_{ij})$.

CFDP detects non-spherical clusters and automatically finds the correct number of clusters. However, it encounters the following drawbacks in practice:

(1) High time complexity: The time complexity of the algorithm is $O(mn^2)$, where m and n are the number of attributes and instances, respectively. The high time complexity makes it impossible to use the algorithm for large-scale data clustering.
(2) Difficult to set d_c and accurately select the cluster centers. The performance of the algorithm depends on user-specified cutoff distance d_c. No specific method was presented in (Rodriguez and Laio 2014).

Liu et al. (2017b) also discovered this problem. He has the following description in the abstract of the paper. "However, the improper selection of its parameter cutoff distance d_c will lead to the wrong selection of initial cluster centers, but the CFDP cannot correct it in the subsequent assignment process. Furthermore, in some cases, even the proper value of d_c was set, initial cluster centers are still difficult to be selected from the decision graph." Chen et al. (2016) described this problem in his paper. "But is has two big challenges when selecting cluster centers. The first challenge is it needs to manually select cluster centers. Even so, on some datasets, the number of cluster centers it generates will be either more or less than the right number. The second one is it is unable to group data points correctly when a cluster has more than one centers."

Various methods have been proposed to further improve the CFDP algorithm. Liu et al. (2017b, 2018), Chen et al. (2016), Xie et al. (2016), Du et al. (2016) introduced k-nearest neighbors (KNN) ideas into the CFDP algorithm to improve its adaptive ability and performance. Xu et al. (2016), Liang and Chen (2016), Lu and Zhu (2017) combined the idea of hierarchy with the CFDP algorithm to improve efficiency.

Xie et al. (2016) proposed a new robust fuzzy KNN density peak clustering (FKNN-DPC) algorithm. The proposed algorithm introduced a uniform metric to calculate the local density and developed two assignment strategies to detect the true distribution of a dataset. Liu et al. (2018) proposed a shared-nearest-neighbor-based clustering by fast search and find of density peaks (SNN-DPC) algorithm. They presented three new definitions: SNN similarity, the local density ρ and the distance from the nearest larger density point δ. These definitions take the information about nearest neighbors and shared neighbors into account. Xu et al. (2016) proposed a density peak-based hierarchical clustering method (DenPEHC) algorithm that directly generates clusters on each possible clustering layer and introduced a grid granulation framework to enable DenPEHC to cluster large-scale and high-dimensional datasets.

As opposed to the above methods, TSD uses two-stage clustering. The first stage uses a two-round-means algorithm to better handle spherical datasets. The second stage uses the improved CFDP algorithm to efficiently process non-spherical datasets. Thus, TSD effectively integrates the idea of hierarchical clustering, which can improve the adaptability of the algorithm.

2.3 Maximum margin clustering

MMC algorithms (Li et al. 2009) aim to find an optimal (maximum) hyperplane in high-dimensional feature space. Specifically, the hyperplane and labeled sample can be obtained by optimizing the following objective function:

$$\begin{aligned} &\min_{y\in\{\pm 1\}^n} \min_{\omega,b,\xi} \tfrac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\xi_i, \\ &s.t.\ \ y_i(\omega^T\phi(x_i)+b) \geq 1-\xi_i, \forall i = 1,\ldots,n, \\ &\xi_i \geq 0, \forall i = 1,\ldots,n, \\ &-l \leq \sum_{i=1}^{n} y_i \leq l, \end{aligned} \tag{3}$$

where $\phi(\cdot)$ denotes a nonlinear mapping from the original space to high-dimensional space. $\xi_i \geq 0$ is the relaxation variable that corresponds to x_i. l is the constant controlling the balance between classes. ω and b uniquely determine the hyperplane. Naturally, the clustering label is optimized by the objective function.

MMC is limited to small to medium-sized datasets because of the semidefinite program. The LGMMC algorithm (Li et al. 2009) improves efficiency and scalability by maximizing the margin of opposite clusters using label generation.

2.4 Spectral clustering

Spectral clustering (Chen and Cai 2011; Wang et al. 2011) is evolved from graph theory. It mainly includes three steps:

Step 1. Construct a new matrix to represent the original dataset.
Step 2. Compute the eigenvalues and eigenvectors of the matrix. Map each instance to a low-dimensional representation based on the eigenvectors.
Step 3. Assign cluster indices according to the new representation.

Spectral clustering has advantages in managing sparse and high-dimensional datasets. The disadvantage is that the time complexity is too high and cannot manage intersections. Chen and Cai (2011) proposed the landmark-based spectral clustering algorithm to improve efficiency. Wang et al. (2011) proposed the spectral multi-manifold clustering (SMMC) algorithm to manage intersections.

2.5 Balanced clustering

Balanced clustering (Liu et al. 2017a) is required in a variety of applications, such as photo query systems (Dengel et al. 2011) and wireless sensor networks (Chuang et al. 2009). These balanced algorithms can be categorized into two types: hard-balanced and soft-balanced. Liu et al. (2017a) proposed a soft-balanced BCLS algorithm based on least square linear regression. It considers a balance constraint to regularize the clustering model. The purpose is to minimize

$$\sum_{k=1}^{c} s_k^2 = \|s\|_2^2 = \left\|1^T Y\right\|_2^2 = tr(Y^T 1 1^T Y). \tag{4}$$

The algorithm achieves balanced clustering by minimizing the square sum of instances in each cluster.

Algorithm 1 two-round-means

Input: Dataset with n instances $U = \{x_1, \ldots, x_n\}$.
Output: $b = [b_1, b_2, \ldots, b_e]$, where b_i denotes the set of all instances in block i and $c = [c_1, c_2, \ldots, c_e]$ denotes the virtual centers array.

1: $b_1 = b_2 = \cdots = b_e = \emptyset$;
2: Randomly select e virtual centers $c = [c_1, c_2, \ldots, c_e]$.
3: **for** $r = 1$ to 2 **do**
4: **for** $i = 1$ to n **do**
5: $m = 0, dist = \infty$;
6: **for** $j = 1$ to e **do**
7: **if** $(d(x_i, c_j) < dist)$ **then**
8: $dist = d(x_i, c_j)$;
9: $m = j$;
10: **end if**
11: **end for**
12: $b_m = b_m \cup \{x_i\}$;
13: **end for**
14: Recompute the virtual centers according to Equation (5).
15: **end for**
16: **return** b, c;

3 Proposed algorithm

In this section, we present the TSD algorithm with time complexity analysis.

3.1 Algorithm description

Figure 2 shows our TSD clustering framework. Table 1 is the symbols and variables used in Fig. 2. Stage I is pre-clustering. The dataset is divided into e clusters using the two-round-means algorithm (Algorithm 1), $e = \sqrt{n}$. $\sqrt{n}$ is an empirical value used in many studies. In Yu and Cheng (2001), the authors have provided a theoretical explanation for this rule. Therefore, we set $\sqrt{n}$ as the block number in Algorithm 1. This stage computes block information $b_{1\times e}$ and determines virtual centers $c_{1\times e}$. Simultaneously, we obtain block size array $\rho = [|b_1|, |b_2|, \ldots, |b_e|]$ for the next stage. We do not use the k-means algorithm directly. Instead, we control the iteration to exactly two to save runtime because more iterations require more time, but the performance of the clustering cannot be substantially improved. We discuss this issue in Sect. 4.

Stage II is the density clustering of all virtual centers (Algorithm 2). This stage acquires cluster indices $cl_{1\times e}$. First, density ρ_i of each virtual center is obtained directly from the first stage. Second, we construct master tree $ms_{1\times e}$ based on density ρ and minimal distance δ. Finally, we cluster the virtual centers according to the master tree. The algorithm has the following advantages compared with CFDP. First, we only need to cluster e virtual centers. This is the major technique that reduces the time complexity. Second, density ρ_i is redefined as the size of b_i without setting cutoff distance d_c.

Finally, all remaining instances in each block have the same cluster indices as the virtual centers. They are assigned cluster indices $l = [l_1, l_2, \ldots, l_n]$, that is, $\forall x \in b_i, l_x = cl_i$.

3.1.1 Stage I: Two-round-means

Algorithm 1 lists the two-round-means algorithm. It mainly includes three steps:

Step 1. Initialization. Line 1 initializes $b_1 = b_2 = \cdots = b_e = \emptyset$. Line 2 initializes virtual centers $cl_{1\times e}$ using random sampling.
Step 2. Compute block information. Lines 6–11 find the nearest center of each instance and assign it to the corresponding block. Line 9 finds the nearest center c_j for x_i, denoted as m. Line 12 adds the instance x_i to block b_m.
Step 3. Recalculate the virtual centers. The virtual centers need to be updated based on the generated block information. Line 14 recalculates the virtual center c_k of block k according to Eq. (5).

$$c_k = \frac{\sum_{x_i \in b_k} x_i}{|b_k|}. \tag{5}$$

The loop terminates after two iterations for the following reason: For the k-means algorithm, k is often a small integer that corresponds to the number of clusters. Hence, the algorithm may run many iterations to converge to k clusters. By contrast, in Algorithm 1, the number of blocks is e, which is quite large. These e local blocks influence, but do not determine, the final clusters.

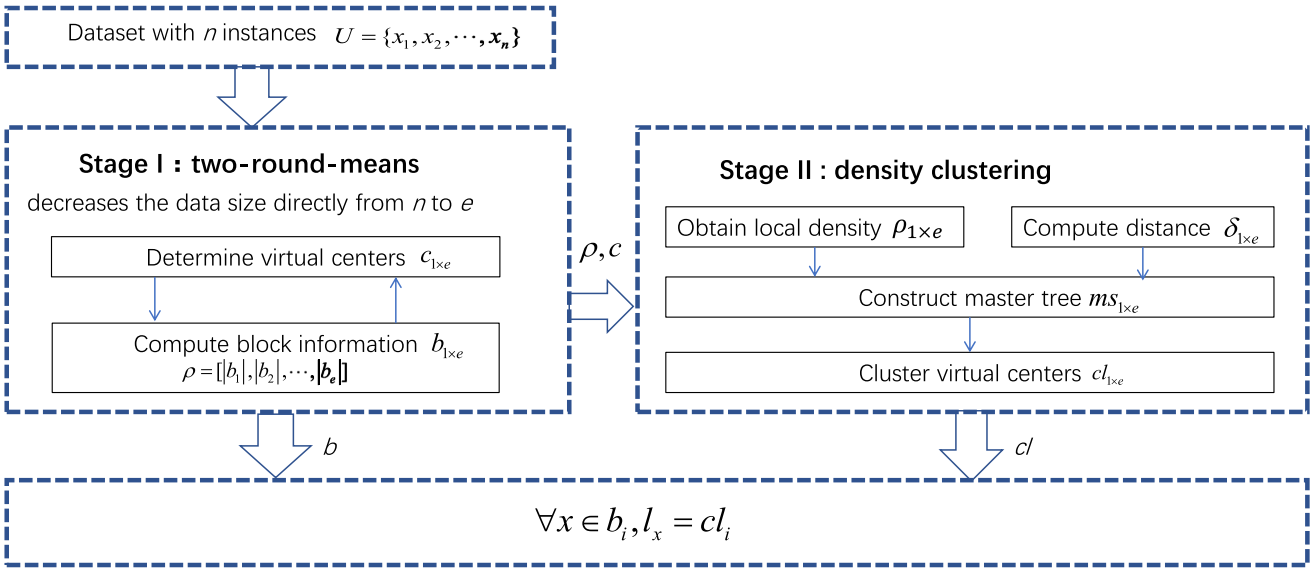


Fig. 2 TSD algorithm framework

Table 1 Notation and variables used in Fig. 2

Notation	Meaning	Comments
U	All instances	$U = \{x_1, \ldots, x_n\}$
e	$e = \lfloor \sqrt{n} \rfloor$	
ρ_i	The density of x_i	See Eq. (6)
δ_i	The minimum distance of x_i	See Eq. (2)
b	Block information	e clusters
$\|b_i\|$	The size of the ith block	$\|b_i\| = \rho_i$
c_i	The ith virtual center	e virtual centers
ms_i	The master index of x_i	
cl_i	The cluster indices of c_i	$cl = [cl_1, cl_2, \ldots, cl_e]$
l_i	The cluster indices of x_i	$l = [l_1, l_2, \ldots, l_n]$

3.1.2 Stage II: Density clustering

Algorithm 2 lists the density clustering algorithm. It mainly includes the following two steps:

Step 1. Construct the master tree.

Lines 2–3 obtain density ρ and sort. In CFDP, density ρ is computed based on Eq. (1). However, it requires the manual setting of cutoff parameter d_c. Considering the local distribution characteristics and nonparametric properties, the density ρ_i is redefined as

$$\rho_i = |b_i|, \tag{6}$$

where $|b_i|$ is the size of block i. It can be obtained directly from the first stage of pre-clustering.

Lines 4–12 calculate minimum distance δ and construct the master tree. From lines 5–12, δ is computed based on Eq. (2). δ_i is the minimum distance between instance i and any other instance with a higher density. Line 6 indicates that the search ranges from c_{q_1} to $c_{q_{i-1}}$, where $q = [q_1, \ldots, q_e]$ is the index array according to ρ in descending order, $\rho_{q_1} \geq \rho_{q_2} \geq \cdots \geq \rho_{q_e}$. In line 8, closest distance $d(c_{q_i}, c_{q_j})$ is determined, and termed δ_{q_i}. According to Definition 1, a master is the nearest neighbor with a higher density. Line 9 updates the master of q_i as q_j. Thus, the master tree is built. When there are multiple masters, we choose that with the smallest index.

Definition 1 (Wang et al. 2017) Let $x_i, x_j \in U$ and $d(x_i, x_j)$ be the distance between x_i and x_j. $x_j \in U$ is called a master of x_i iff

1. $\rho(x_j) > \rho(x_i)$; and
2. $\forall x_l \in U, \rho(x_l) > \rho(x_i) \Rightarrow d(x_i, x_j) \leq d(x_i, x_l)$.

Step 2. Cluster and assign cluster indices to all virtual centers.

The cluster centers are characterized by a higher density than their neighbors and by a relatively large distance

Algorithm 2 density clustering

Input: $b = [b_1, b_2, \ldots, b_e]$, where b_i denotes the set of all instances in block i and $c = [c_1, c_2, \ldots, c_e]$ denotes the virtual centers.
Output: $cl = [cl_1, \ldots cl_e]$ denotes the cluster indices for the virtual centers.

1: $cl = ms = [-1, \ldots, -1]_{1\times e}$; //$ms$ is the master array
   //Step 1 Construct the master tree
   //Step 1.1 Obtain ρ and sort
2: $\rho = [\rho_1, \ldots, \rho_e] = [|b_1|, \ldots, |b_e|]$;
3: $q = [q_1, \ldots, q_e]$ = sort(ρ); //$\rho_{q_1} \geq \rho_{q_2} \geq \cdots \geq \rho_{q_e}$
   //Step 1.2 Compute δ and construct the master tree
4: $\delta = [\delta_1, \ldots, \delta_e]$ = [MAX, …, MAX];//Initialized as the maximal value
5: **for** ($i = 2$ to e) **do**
6: **for** ($j = 1$ to $i - 1$) **do**
7: **if** ($d(c_{q_i}, c_{q_j}) < \delta_{q_i}$) **then**
8: $\delta_{q_i} = d(c_{q_i}, c_{q_j})$;
9: $ms_{q_i} = q_j$; //Update the master
10: **end if**
11: **end for**
12: **end for**
   //Step 2. Clustering
   //Step 2.1 Compute γ and sort
13: $\gamma = [r_1, \ldots, r_e] = [\rho_1 \ldots \delta_1, \ldots, \rho_e \ldots \delta_e]$;
14: $p = [p_1, \ldots, p_e]$ = sort(γ);//$\gamma_{p_1} \geq \gamma_{p_2} \geq \cdots \geq \gamma_{p_e}$
   //Step 2.2 Assign cluster indices to k cluster centers
15: **for** ($i = 1$ to k) **do**
16: $cl_{p_i} = i$;
17: **end for**
   //Step 2.3 Assign cluster indices to other virtual centers
18: **for** ($i = 1$ to e) **do**
19: **if** ($cl_{q_i} == -1$) **then**
20: $cl_{q_i} = cl_{ms_{q_i}}$;
21: **end if**
22: **end for**
23: **return** cl;

from instances with higher densities (Rodriguez and Laio 2014). To consider both density and distance, we introduce the importance measure:

$$\gamma = \rho \cdot \delta. \tag{7}$$

Lines 13–14 compute γ and sort. $p = [p_1, \ldots, p_e]$ is the index array according to γ in descending order, and $\gamma_{p_1} \geq \gamma_{p_2} \geq \cdots \geq \gamma_{p_e}$. Lines 15–17 select k centers in turn according to $p = [p_1, \ldots, p_e]$. k is given by the user, and it is usually set to the actual number of clusters. Line 16 assigns the cluster index i to the ith center. Lines 18–22 assign cluster indices to non-center instances. The cluster assignment is performed in a single step (line 20), in contrast to other clustering algorithms, where an objective function is optimized iteratively (MacQueen et al. 1967; Kaufman 2008). We prove that each block is assigned a real label.

Property 1 On the completion of Algorithm 2, $\forall 1 \leq i \leq e$, $cl_i \neq -1$.

Proof We prove the property using mathematical induction.

Table 2 Space complexity of the TSD algorithm

Algorithm	Complexity	Description
Algorithm 1	$O(n)$	Block information b
Algorithm 1	$O(n^{\frac{1}{2}})$	Virtual centers c
Algorithm 2	$O(n^{\frac{1}{2}})$	ρ, δ, γ, master tree ms
Algorithm 2	$O(n^{\frac{1}{2}})$	Clusters indices cl
Algorithm 2	$O(n)$	Clusters indices l
Total	$2O(n) + 5O(n^{\frac{1}{2}}) = O(n)$	

(**Basis**) From Eq. (6) and line 3, we know that q_1 corresponds to the instance with the maximal density. Because it also has the maximal distance, according to Eq. (7) and Line 14, $q_1 = p_1$.

Line 16 assigns $cl_{p_1} = 1$, which also indicates that $cl_{q_1} = 1 \neq -1$.

(**Induction**) Suppose that $cl_{q_k} \neq -1$ for $1 \leq k \leq i - 1$ while executing line 20.

Because ms_{q_i} is the master of q_i, we have $\rho_{ms_{q_i}} > \rho_{q_i}$. Moreover, blocks are sorted in line 3. Let $q_j = ms_{q_i}$, then naturally $j < i$. According to the hypothesis, $cl_{q_j} \neq -1$, and $cl_{q_i} \neq -1$ after executing line 20.

Combining the basis and induction, the property holds. This completes the proof. □

3.2 Complexity analysis

We analyze the space and time complexities of the algorithm.

Proposition 1 *Let m and n be the number of attributes and instances, respectively. For Algorithm TSD, the space and time complexity are $O(n)$ and $O(mn^{\frac{3}{2}})$, respectively.*

Proof Table 2 lists the space complexity. ρ, δ and γ require $O(n^{\frac{1}{2}})$ of space. Virtual center c and its cluster index cl also require $O(n^{\frac{1}{2}})$ of space. Block information b and cluster index l require $O(n)$ of space. Thus, the total space complexity is

$$3O(n^{\frac{1}{2}}) + 2O(n^{\frac{1}{2}}) + 2O(n) = O(n). \tag{8}$$

Table 3 lists the time complexity. Algorithm 1 takes $O(mn^{\frac{3}{2}})$ of time. Algorithm 2 takes $O(mn)$ of time. The final step for cluster index sharing takes $O(n)$ of time. Therefore, the time complexity of Algorithm TSD is

$$O(mn^{\frac{3}{2}}) + O(mn) + O(n) = O(mn^{\frac{3}{2}}). \tag{9}$$

This completes the proof. □

Table 3 Time complexity of the TSD algorithm

Algorithm	Lines	Complexity	Description
Algorithm 1	Line 2	$O(mn^{\frac{3}{2}})$	Stage I two-round-means
Algorithm 2	Lines 2–3	$O(me^2) = O(mn)$	Stage II compute ρ and sort
Algorithm 2	Lines 5–12	$O(me^2) = O(mn)$	Stage II compute δ and updata master
Algorithm 2	Lines 13–14	$O(me^2) = O(mn)$	Stage II compute γ and sort
Algorithm 2	Lines 15–22	$O(n^{\frac{1}{2}})$	Stage II clustering
		$O(n^{\frac{1}{2}})$	Cluster indices sharing
Total	$O(mn^{\frac{3}{2}}) + O(n^{\frac{1}{2}}) + 3O(mn) + O(n) = O(mn^{\frac{3}{2}})$		

4 Experiments

We conduct experiments to analyze the adaptability, clustering performance and efficiency of the TSD algorithm and to answer the following question:

(1) Does the TSD algorithm have better clustering performance than classical and state-of-the-art clustering algorithms, such as k-means, CFDP, CFSFDP+A and SNN-DPC?
(2) Is the TSD algorithm efficient?

The computations are performed on a Windows 10 64-bit operating system with 8 GB RAM and Intel (R) Core(TM) i5-8300H CPU @2.30 GHz processors, using Java and MATLAB software. The TSD source code is available at www.fansmale.com/software.html and https://github.com/FanSmale/TSD.

4.1 Datasets

We chose different types, shapes and sizes of datasets for the experiments. These included six synthetic datasets obtained from the literature (Rodriguez and Laio 2014), 12 from the University of California at Irvine (UCI) ML repository (Blake and Merz 1998) and one from the literature (Stenger 2011). The six synthetic datasets had typical shape distributions. The number of instances ranged from 150 to 1,025,009, the number of attributes ranged from 2 to 40, and the number of classes ranged from 2 to 17. These datasets are listed in Table 4.

Note that the class attributes of the datasets were not used in the clustering processing. We first removed the labels for all instances, then predicted the cluster indices and finally measured the clustering performance according to the true label.

Figure 3 illustrates the six synthetic datasets with two-dimensional visualization graphs. The six synthetic datasets had different data distributions, shapes, cluster sizes and numbers of clusters. Figure 3a shows the typical non-spherical dataset, which was divided into three classes, thereby forming three semicircular rings. Table 4 (lines 7–11) lists the selected five typical small UCI datasets. It is perhaps the best-known database in the clustering, classification literature (Chiroma et al. 2014). Table 4 (lines 12–21) lists ten large datasets of different types and different domains. The Poker (Cattral and Oppacher 2007) dataset contained 1,025,009 instances and 10 attributes, and the DLA (Ugulino et al. 2012) dataset contained 165,633 instances and 17 attributes.

4.2 Evaluation measure

We use four external evaluation functions to evaluate the clustering algorithm, including purity, JC, FMI and RI.

We assume that the clusters given by the clustering algorithm were divided into $C = (C_1, C_2, \ldots, C_k)$ and the clusters given by the reference model were divided into $C^* = (C_1^*, C_2^*, \ldots, C_s^*)$. λ and λ^* denote the cluster index vectors that correspond to C and C^*, respectively. Accordingly, we compute the following four parameters:

$$a = |SS|, SS = \{(x_i, x_j)|\lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \quad (10)$$

$$b = |SD|, SD = \{(x_i, x_j)|\lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}, \quad (11)$$

$$c = |DS|, DS = \{(x_i, x_j)|\lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}, \quad (12)$$

$$d = |DD|, DD = \{(x_i, x_j)|\lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}. \quad (13)$$

According to Eqs. (10)–(13), we obtain the external evaluation functions as follows:

$$\text{Purity} = \frac{1}{n}\sum_k \max_s \left|C_k \cap C_s^*\right|, \quad (14)$$

$$\text{JC} = \frac{a}{a+b+c}, \quad (15)$$

$$\text{FMI} = \sqrt{\frac{a}{a+b} * \frac{a}{a+c}}, \quad (16)$$

$$\text{RI} = \frac{2(a+d)}{n(n-1)}, \quad (17)$$

Table 4 Dataset information

ID	Name	Source	Domain	n	m	k
1	Iris	UCI	Botany	150	4	3
2	Seeds	UCI	Biological	210	7	3
3	Glass	UCI	Material	214	10	6
4	Flame	Synthetic	N/A	240	2	2
5	Spiral	Synthetic	N/A	312	2	3
6	Ecoli	UCI	Life	336	8	8
7	Jain	Synthetic	N/A	373	2	2
8	Compound	Synthetic	N/A	399	2	6
9	Led7digit	UCI	Computer	500	10	10
10	R15	Synthetic	N/A	600	2	15
11	Aggregation	Synthetic	N/A	788	2	7
12	Texture	UCI	Material	5500	40	11
13	Ring	KEEL	Historical	7400	20	2
14	Twonorm	KEEL	Historical	7400	20	2
15	Penbased	UCI	Computer	10,992	16	10
16	Magic	UCI	Physical	19,020	10	2
17	Kr-vs-k	UCI	Game	28,056	6	17
18	ConfLongDemo	UCI	Life	164,860	8	11
19	DLA	DWP	Society	165,633	17	5
20	Skin	UCI	Computer	245,057	3	2
21	Poker	UCI	Game	1,025,009	10	10

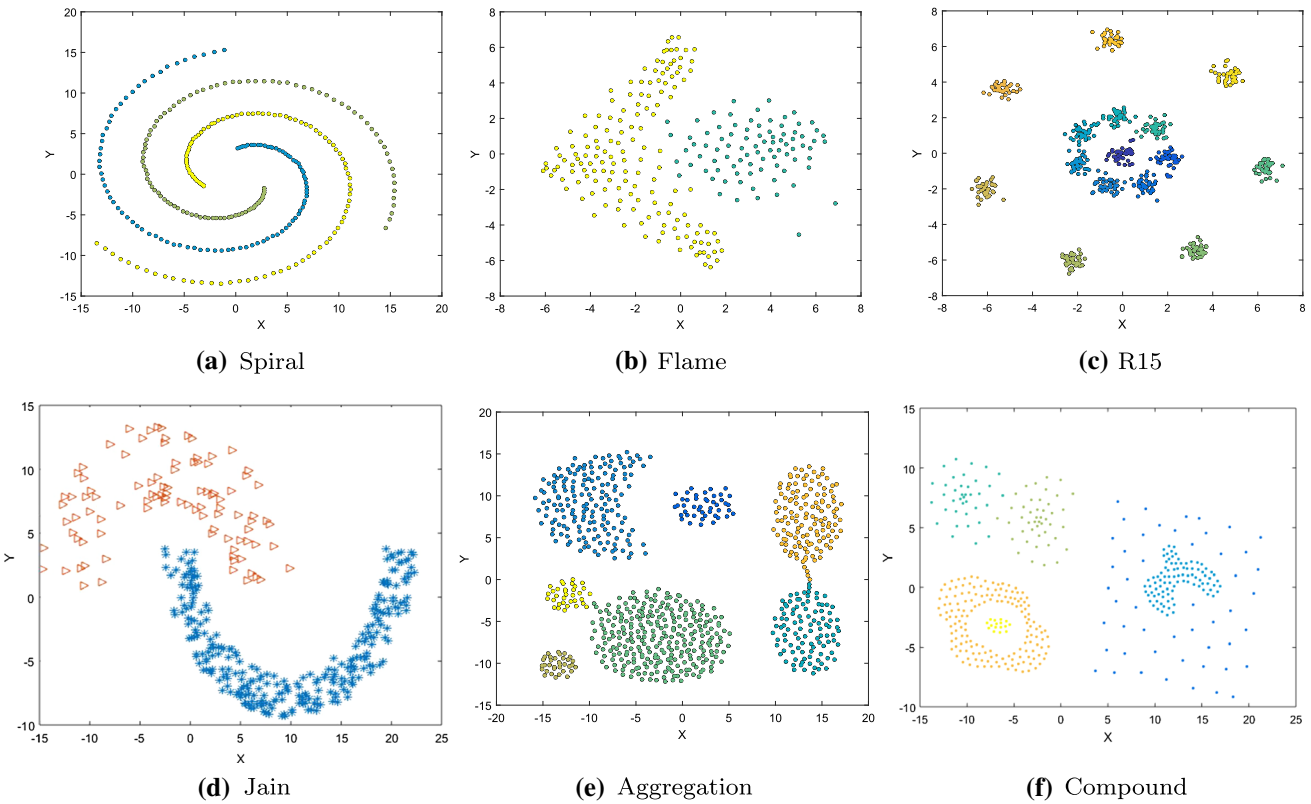


Fig. 3 Two-dimensional visualization of six of the datasets

where n is the number of instances. Obviously, the results of the above performance metrics were all between zero and one, the bigger the better.

4.3 Algorithm adaptability

In this section, we design our experiments to observe the effect of the preporcessing technique for different shapes of datasets and elaborated the parameter settings.

4.3.1 Impact of parameter settings

In this subsection, we mainly answer the following two questions through experiments:

(1) Is it appropriate to set the number of iterations to two?
(2) Is it appropriate to divide the dataset into e clusters?

We chose seven different sizes and different types of datasets, including Iris and Seeds. Figure 4a shows the change of purity when the number of iterations r varied from 2 to 19. For Twonorm, Magic, Glass and Ring, the purity variation was small. Increasing the number of iterations cannot improve its performance. For Seeds, the highest purity was achieved by iterating twice. For Jain and Iris, the difference in optimal performance was less than 10%. Therefore, it was appropriate to set number of iterations r to two. The efficiency of clustering is greatly improved while the clustering performance is guaranteed.

Figure 4b shows the change of purity when number of clusters k varied from $0.5e$ to $1.5e$. The performance is the best when $k = e$. If the number of clusters continued to increase, purity no longer increased. Therefore, it was appropriate to divide the dataset into e clusters.

4.3.2 Effect of the preprocessing technique for different shapes of datasets

Figure 5 shows the pre-clustering and sampling process of the original dataset. We choose synthetic datasets with typical shape distributions, such as R15, Jain, Aggregation and Compound. Figure 5a, d, g, j shows the original distribution of datasets. Jain is two semicircles; Aggregation is seven clusters with different shapes and sizes; Compound is petals and scatter patterns; R15 was three rings formed by 15 evenly distributed clusters.

Figure 5b, e, h, k shows the e clusters formed by two-round-means. Figure 5c, f, i, l shows the e virtual centers. Figure 5 shows that local blocks b and virtual centers c maintained good data distribution characteristics. The pre-clustering process did not destroy the data structure.

4.4 Comparison with state-of-the-art clustering algorithms

Because a considerable amount of work has been conducted on clustering, it is interesting and meaningful to compare our proposed TSD algorithm with these state-of-the-art methods. We compare the TSD algorithm with state-of-the-art eight clustering approaches, including partition-based clustering (k-means algorithm) (MacQueen et al. 1967) and density-based clustering (CFDP (Rodriguez and Laio 2014)[1]), balanced clustering (BCLS) (Liu et al. 2017a)[2], SNN-DPC (Liu et al. 2018)[3], FKNN-DPC (Xie et al. 2016), DenPEHC (Xu et al. 2016)[4], CFSFDP+A and CFSFDP+DE (Bai et al. 2017).[5]

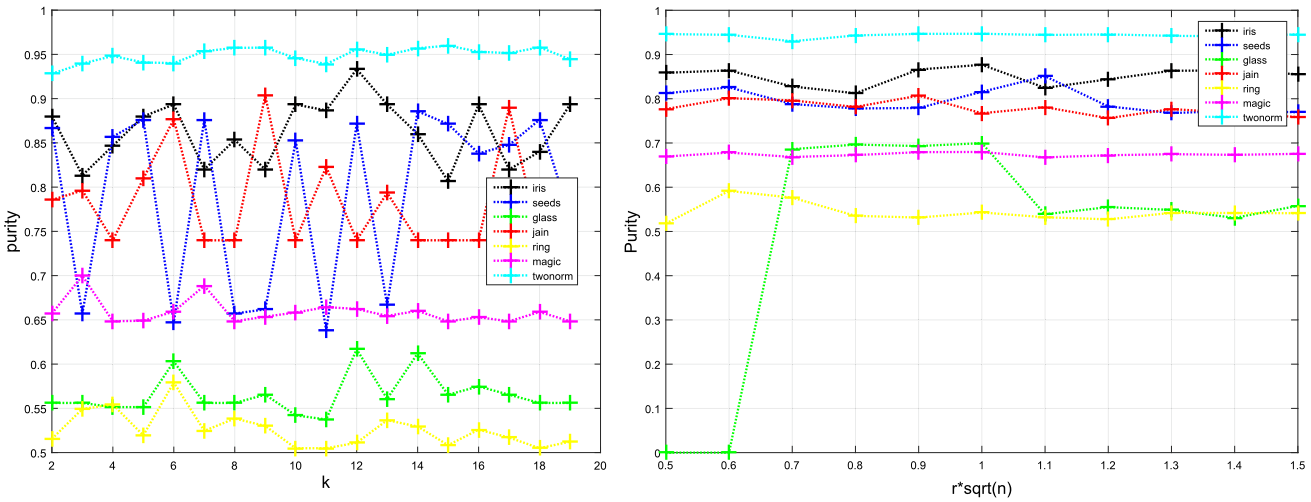
k-means is tested using Weka's built-in codes. CFDP, SNN-DPC, FKNN-DPC, DenPEHC, BCLS, CFSFDP+A and CFSFDP+DE use the source code provided by the algorithm authors. We list the URLs of the source code provided by the author. For SNN-DPC, we adopt the original results in the paper to avoid parameter adjustment. For datasets not available in the original paper, we choose $k = 11$ for testing. (For k values, the recommended range in the original paper is 5-30.) The CFDP algorithm uses the optimal parameter setting and the Gaussian distance described in the original paper. The TSD algorithm is run ten times, and the average purity is calculated. At the same time, we list the optimal results of the TSD algorithm. This result is for reference only and is referred to as TSD-H for short.

Comprehensive evaluations of clustering performance are performed on 21 datasets, which cover a wide range of properties. For CFDP, SNN-DPC, FKNN-DPC, DenPEHC and BCLS, the code runs on the MATLAB platform. Memory overflows when testing large datasets. Therefore, for the Pen-based, Magic and Kr-vs-k datasets, we sample 10% of the instances for each class and formed new datasets. For the ConfLongDemo, DLA, Poker and Skin datasets, we sample 1% of the instances for each class and formed new datasets.

4.4.1 Comparison on synthetic datasets

Tables 5, 6, 7 and 8 compare the purity, JC, FMI and RI of TSD with that of the eight state-of-the-art clustering algorithms on synthetic datasets. The mean ranks of algorithms are listed from a statistical point of view. The best and second-best results are highlighted in boldface and italic,

[1] http://people.sissa.it/~laio/Research/Res_clustering.php.
[2] https://github.com/ericstark/BCLS.
[3] https://github.com/liurui39660/SnnDpc.
[4] https://github.com/alanxuji/DenPEHC.
[5] https://www.researchgate.net/publication/317617974_Fast_density_clustering_strategies_based_on_the_k-means_algorithm.



(a) Parameter test of r iterations. **(b)** Parameter test of k clusters.

Fig. 4 Impact of parameter settings

respectively. For TSD, the mean ranks are 5.83, 5.33, 6.42 and 4.67, respectively.

4.4.2 Comparison on benchmark datasets

Tables 9, 10, 11 and 12 compare the purity, JC, FMI and RI of TSD with that of the eight state-of-the-art clustering algorithms on benchmark datasets. We perform statistical analysis of the experiments using the methods described in Reyes et al. (2018) and Zhou (2016). The average ranks are obtained by applying a Friedman test Reyes et al. (2018), which is the most well-known nonparametric test. The Friedman test analyzes whether there are significant differences among the algorithms. TSD is generally superior to the eight clustering algorithms. For purity, the average rank is 3.30. The average rankings of other eight algorithms are 4.13, 4.37, 7.70, 3.33, 5.17, 8.17, 4.43 and 4.40, respectively. In particular, the TSD algorithm has the highest purity on three datasets.

4.4.3 Comparison on domain datasets

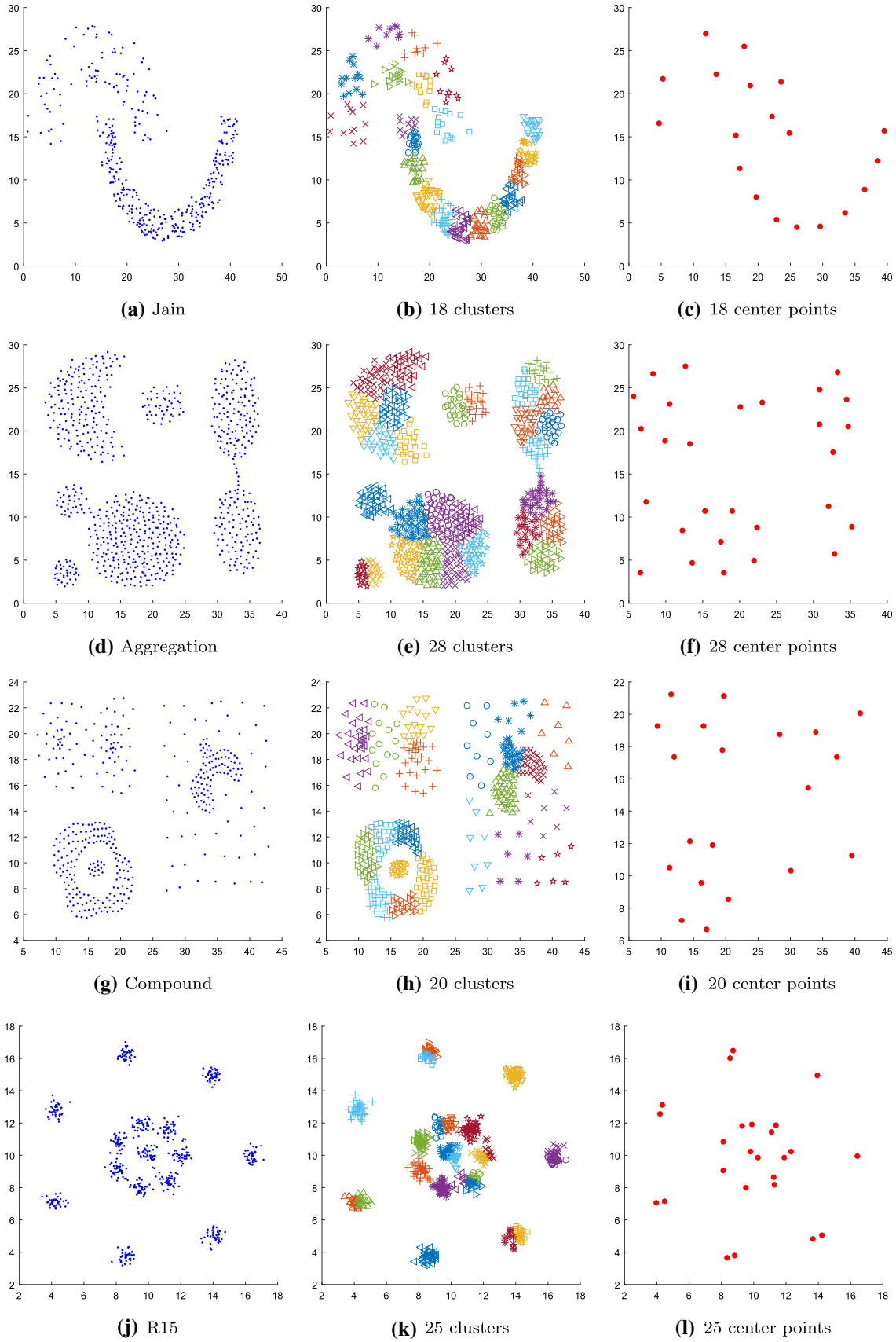
We adopt actual line loss data to further verify the performance of TSD. Line loss data contain 2585 instances, i.e., 2585 electrical transformer districts. The five attributes are wire size, wire length, active power, reactive power and energy indication. Decision attributes are five types of districts.

Figure 6a illustrates the purity of TSD and the eight clustering algorithms. Figure 6b–d illustrates the comparison of JC, FMI and RI, respectively. For TSD, purity, JC, FMI and RI are 0.9280, 0.8109, 0.8955 and 0.9058, respectively. TSD ranks first among all four indicators. The purity of the other eight clustering algorithms is 0.6572, 0.8300, 0.6112, 0.6274, 0.6112, 0.6112, 0.6758 and 0.6271. The JC of the other eight clustering algorithms is 0.5533, 0.3325, 0.4509, 0.1951, 0.2252, 0.4509, 0.4786 and 0.4478. The FMI of the other eight clustering algorithms is 0.7417, 0.5240, 0.6715, 0.3448, 0.3738, 0.6715, 0.6879 and 0.6516. The RI of the other eight clustering algorithms is 0.7973, 0.6538, 0.4509, 0.5404, 0.5175, 0.4509, 0.7623 and 0.7389. TSD has good performance in the actual line loss prediction.

4.4.4 Comparison with CFDP optimization algorithm

Finally, we compare TSD with three CFDP optimization algorithms: DPCG (Xu et al. 2018a), GDPC (Xu et al. 2018b) and CDPC (Xu et al. 2018b). DPCG (Xu et al. 2018a) is an improved grid-based density peak clustering algorithm. GDPC (Xu et al. 2018b) and CDPC (Xu et al. 2018b) are two improved density peak clustering algorithms that quickly find cluster centers. For these three algorithms, the author did not provide source code. To ensure optimal performance, we adopt the datasets and results provided in these references for comparison. There is no need to run the program and adjust parameters.

Table 13 compares the purity of TSD and DPCG. For six datasets, TSD performed better than DPCG on four datasets (e.g., Soybean, Ionosphere). The mean rank of TSD and DPCG are 1.33 and 1.67, respectively. Table 14 compares the purity of TSD, GDPC and CDPC. For six datasets, TSD performed better than DPCG on four datasets. The average rankings of GDPC, CDPC and TSD are 2.33, 2.00 and 1.67, respectively.



(a) Jain **(b)** 18 clusters **(c)** 18 center points

(d) Aggregation **(e)** 28 clusters **(f)** 28 center points

(g) Compound **(h)** 20 clusters **(i)** 20 center points

(j) R15 **(k)** 25 clusters **(l)** 25 center points

Fig. 5 Effect of the preprocessing technique for different shapes of datasets

Table 5 Purity of TSD and eight clustering algorithms on synthetic datasets

Purity	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Flame	0.7787	**1.0000**	0.6375	0.9875	0.9875	0.9792	*0.9917*	0.9204	0.8929	0.9917
Spiral	0.3520	**1.0000**	0.3397	*0.8878*	0.6635	0.6763	0.3590	0.4269	0.3907	0.5128
Jain	0.8456	0.8928	0.7399	**1.0000**	0.7400	*0.9464*	**1.0000**	0.8491	0.8515	1.0000
Compound	0.8406	0.8321	0.3960	**0.9148**	*0.8591*	0.6266	0.8471	0.8043	0.8191	0.9023
R15	0.8855	*0.9967*	0.0667	*0.9967*	0.9000	0.0667	**1.0000**	0.6830	0.8817	0.9967
Aggregation	0.8111	**0.9987**	0.3464	0.9911	0.7970	0.8541	*0.9937*	0.9894	0.9292	1.0000
Mean rank	6.50	*2.42*	8.92	**2.25**	4.92	5.58	2.75	5.83	5.83	–

Table 6 JC of TSD and eight clustering algorithms on synthetic datasets

JC	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Flame	0.5402	**1.0000**	0.5359	0.1296	*0.9551*	0.9265	0.5357	0.6658	0.7409	0.9694
Spiral	0.1962	**1.0000**	0.3313	0.3125	0.4393	*0.6002*	0.3251	0.2562	0.2338	0.2839
Jain	0.6240	0.7135	0.6141	0.1240	0.4664	*0.9639*	**1.0000**	0.6559	0.6441	1.0000
Compound	0.5764	0.4874	0.2472	0.2509	**0.7928**	0.4574	*0.7851*	0.5577	0.4721	0.7771
R15	0.7625	*0.9866*	0.0651	0.9560	0.6987	0.0651	**1.0000**	0.4368	0.7339	0.9868
Aggregation	0.5109	*0.9966*	0.2165	0.3512	0.6166	0.7055	**1.0000**	0.6111	0.6931	1.0000
Mean rank	5.83	**2.33**	7.42	7.17	4.17	4.25	*3.00*	5.50	5.33	–

Table 7 FMI of TSD and eight clustering algorithms on synthetic datasets

FMI	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Flame	0.6960	**1.0000**	0.7320	0.9768	*0.9771*	0.9619	0.7300	0.7824	0.8456	0.9845
Spiral	0.3280	**1.0000**	0.5756	**1.0000**	0.6127	*0.7748*	0.5595	0.4163	0.3832	0.4621
Jain	0.7673	0.8348	0.7837	**1.0000**	0.6363	*0.9818*	**1.0000**	0.7810	0.7810	1.0000
Compound	0.7296	0.6627	0.4972	0.8463	**0.8845**	0.6763	*0.8843*	0.7119	0.6424	0.8773
R15	0.8679	*0.9975*	0.2552	0.9933	0.8228	0.2552	**1.0000**	0.6381	0.8433	0.9933
Aggregation	0.6799	*0.9983*	0.4653	0.9681	0.9886	0.8389	**1.0000**	0.7530	0.8171	1.0000
Mean rank	7.00	*2.92*	7.25	**2.67**	4.17	4.92	3.25	6.42	6.42	–

Table 8 RI of TSD and eight clustering algorithms on synthetic datasets

RI	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Flame	0.6849	**1.0000**	0.5359	0.5324	*0.9752*	0.9590	0.5406	0.7586	0.8248	0.9834
Spiral	0.5544	**1.0000**	0.3313	0.7430	0.7181	*0.7793*	0.3508	0.5229	0.5438	0.6076
Jain	0.7382	0.8080	0.6141	0.4620	0.5643	*0.9778*	**1.0000**	0.7372	0.7392	1.0000
Compound	0.8741	0.8516	0.2472	0.8071	**0.9423**	0.7068	*0.9337*	0.8468	0.8336	0.9327
R15	0.9808	0.9991	0.0651	*0.9971*	0.9764	0.0651	**1.0000**	0.9109	0.9778	0.9991
Aggregation	0.8603	*0.9993*	0.2165	0.8590	0.8971	0.9103	**1.0000**	0.8951	0.9261	1.0000
Mean rank	5.00	**2.17**	8.42	6.50	4.33	4.58	*3.33*	6.00	4.67	–

Table 9 Purity of TSD and eight clustering algorithms on benchmark datasets

Purity	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Iris	0.7553	0.9067	0.7600	0.9600	*0.9667*	0.6667	**1.0000**	0.8073	0.8727	0.9800
Seeds	0.8376	**0.9048**	0.7714	*0.8952*	**0.9048**	0.6190	0.6524	0.7862	0.8443	0.9286
Glass	**0.7154**	0.4813	0.3551	0.5467	0.4158	0.3551	*0.6636*	0.5449	0.5537	0.5841
Ecoli	*0.6821*	0.6786	0.4256	0.6458	0.6071	0.4792	0.5833	0.6497	**0.6875**	0.7530
Led7digit	0.6852	*0.7520*	0.2940	0.7160	0.2120	0.1140	**0.9800**	0.7150	0.6754	0.7580
Texture	0.6019	0.1818	0.2727	*0.6335*	0.5273	0.2727	**0.6715**	0.4899	0.6326	0.6805
Ring	**0.5696**	0.5066	0.5049	*0.5278*	0.5049	0.4755	0.5050	0.5054	0.5066	0.5116
Twonorm	*0.9108*	0.6415	0.5004	0.5438	0.5028	0.5004	0.5008	0.5939	**0.9472**	0.9604
Penbased0.1*	0.4323	**0.7352**	0.1110	0.5305	0.6675	0.1041	0.3776	0.3198	*0.6695*	0.6697
Magic0.1*	0.6443	0.6430	0.6425	**0.7560**	0.6772	0.5284	0.6451	0.6513	*0.6791*	0.7035
Kr-vs-k0.1*	0.2221	0.2011	0.1597	0.2535	0.2471	0.0802	**0.3412**	*0.2727*	0.2332	0.2446
ConfLongDemo0.01↑	0.3261	0.3277	0.3161	0.3283	*0.3368*	0.0334	0.3258	**0.3732**	0.3259	0.3313
DLA0.01↑	*0.7374*	0.6872	0.3780	0.6135	0.0568	0.0628	0.4082	0.3931	**0.7613**	0.7627
Skin0.01↑	0.7925	0.7935	0.7935	**0.9873**	0.7935	0.5718	0.7935	0.8092	*0.8198*	0.8539
Poker0.01↑	*0.5142*	0.0914	0.0836	0.0892	0.0856	**0.6425**	0.1247	0.1364	0.0929	0.0956
Mean rank	4.13	4.37	7.70	*3.33*	5.17	8.17	4.43	4.40	**3.30**	–

The best and second best results are highlighted in boldface and italic, respectively
*Indicates 10% sampling of the original dataset
↑Indicates 1% sampling of the original dataset

Table 10 JC of TSD and eight clustering algorithms on benchmark datasets

JC	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Iris	0.5892	0.7248	0.5227	0.2840	*0.8787*	0.5951	**1.0000**	0.7243	0.6793	0.9238
Seeds	0.6231	*0.7022*	0.4798	0.2358	**0.7059**	0.4862	0.4881	0.6145	0.6108	0.7612
Glass	*0.4127*	0.3082	0.2598	0.1911	0.2021	0.2598	**0.4483**	0.3619	0.2906	0.3558
Ecoli	0.2634	**0.4383**	0.2170	0.2339	*0.3741*	0.2891	0.3573	0.3648	0.3512	0.5823
Led7digit	0.4977	0.4043	0.2045	0.3140	0.1118	0.0995	**0.9277**	*0.5595*	0.3519	0.4088
Texture	0.3382	0.1299	0.1581	*0.4285*	0.2554	0.1984	**0.4393**	0.3552	0.3716	0.4541
Ring	0.4153	0.4819	**0.5000**	0.3864	**0.5000**	0.4449	*0.4999*	0.4997	0.4991	0.4999
Twonorm	*0.7443*	0.4400	0.4999	0.4703	0.4987	0.4999	0.4995	0.5762	**0.8188**	0.8586
Penbased0.1*	0.1850	**0.4585**	0.0997	0.1964	0.3633	0.1001	0.1598	0.1497	*0.3842*	0.3902
Magic0.1*	0.4416	0.2428	*0.5404*	0.1642	0.5262	0.3835	0.5403	0.5316	**0.5410**	0.6439
Kr-vs-k0.1*	0.0530	0.0891	0.1060	0.0701	0.0692	0.1043	*0.1278*	**0.1343**	0.0662	0.0967
ConfLongDemo0.01↑	0.0852	0.0906	**0.1899**	0.0900	0.0907	0.1486	0.1109	*0.1659*	0.0930	0.1095
DLA0.01↑	**0.5437**	0.3621	0.2393	0.1765	0.0245	0.0289	0.2529	0.2584	*0.5349*	0.5422
Skin0.01↑	0.4235	0.4628	**0.6721**	0.2038	0.6083	0.4607	0.4605	*0.6448*	0.6033	0.7208
Poker0.01↑	*0.0896*	0.0618	0.0769	0.0745	0.0767	**0.5357**	0.0638	0.0681	0.0566	0.0768
Mean rank	5.13	5.00	5.17	7.20	4.83	6.00	*3.80*	**3.53**	4.33	–

The best and second best results are highlighted in boldface and italic, respectively
*Indicates 10% sampling of the original dataset
↑Indicates 1% sampling of the original dataset

Table 11 FMI of TSD and eight clustering algorithms on benchmark datasets

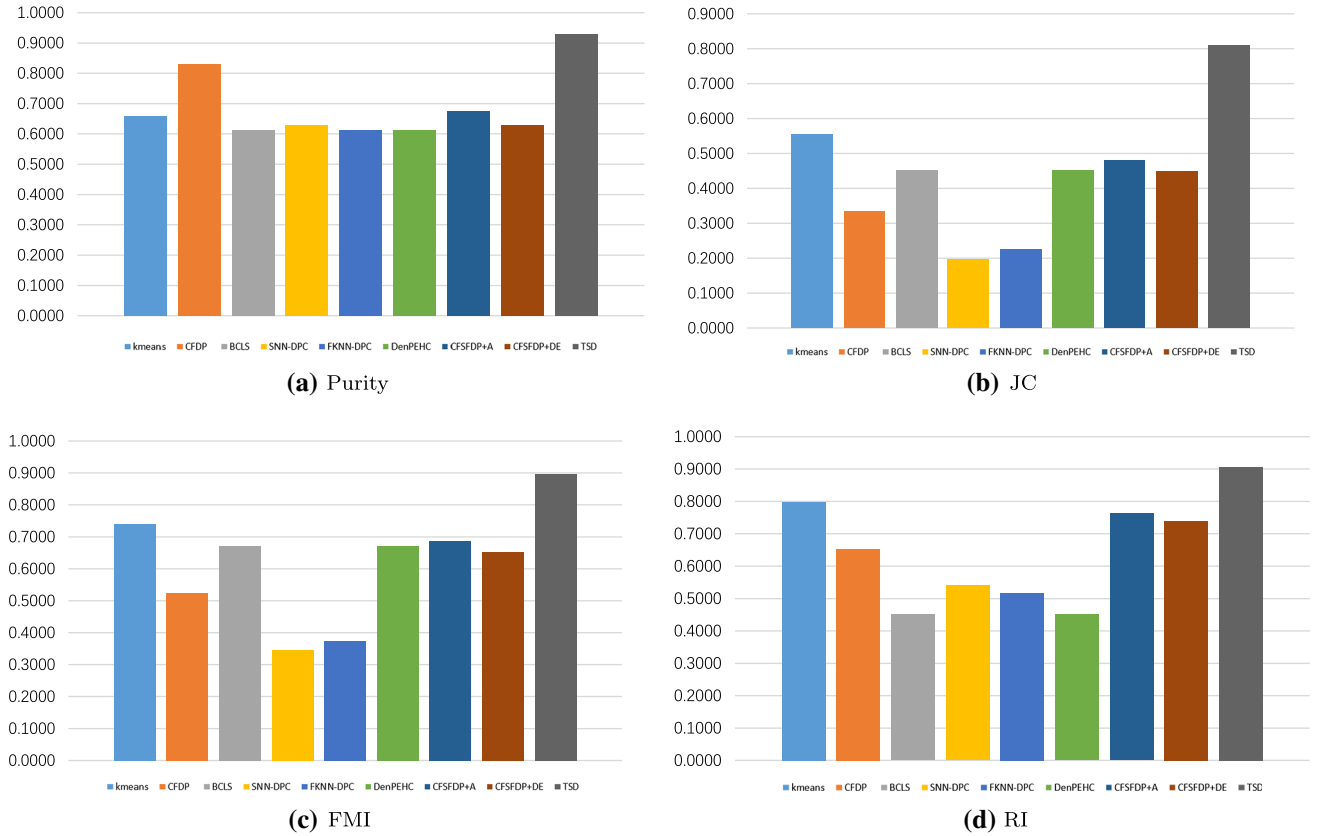
FMI	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Iris	0.7436	0.8407	0.6865	*0.9479*	0.9355	0.7715	**1.0000**	0.8368	0.8088	0.9604
Seeds	0.7627	0.8251	0.6484	**0.8589**	*0.8276*	0.6814	0.6770	0.7562	0.7566	0.8644
Glass	*0.5742*	0.5247	0.5097	0.3213	0.3375	0.5097	**0.6456**	0.5674	0.4565	0.5662
Ecoli	0.4323	*0.6115*	0.5197	**0.8243**	0.5862	0.5085	0.5492	0.5328	0.5256	0.7374
Led7digit	0.6779	0.5758	0.4031	0.4797	0.3142	0.3154	**0.9625**	*0.7334*	0.5211	0.5804
Texture	0.5245	0.3538	0.3950	**0.6413**	0.4100	0.4337	*0.6163*	0.5744	0.5570	0.5687
Ring	0.5950	0.6820	**0.7071**	0.5611	**0.7071**	0.6330	*0.7070*	0.7067	0.7059	0.7070
Twonorm	*0.8461*	0.6210	0.7071	0.6659	0.7054	0.7071	0.7064	0.7529	**0.9002**	0.9239
Penbased0.1*	0.3162	*0.6292*	0.3157	0.3288	**0.9015**	0.3164	0.3092	0.2986	0.5917	0.5999
Magic0.1*	0.6151	0.4044	**0.7351**	0.3471	0.7064	0.5548	*0.7345*	0.7223	0.7275	0.7350
Kr-vs-k0.1*	0.1056	0.1868	*0.3255*	0.1309	0.1297	0.3043	0.2274	**0.3374**	0.1242	0.2191
ConfLongDemo0.01↑	0.1607	0.1682	**0.4358**	0.1680	0.1684	0.2769	0.1996	*0.3157*	0.1720	0.1974
DLA0.01↑	**0.7154**	0.5344	0.4212	0.3083	0.0856	0.3619	0.4467	0.4682	*0.7061*	0.7071
Skin0.01↑	0.5976	0.6328	**0.8198**	0.4485	0.7624	0.6309	0.6307	*0.7901*	0.7577	0.8445
Poker0.01↑	0.2096	0.1357	*0.2774*	0.2323	0.2718	**0.7297**	0.1324	0.1346	0.1165	0.2743
Mean rank	5.67	5.13	4.77	5.67	4.97	5.67	*4.27*	**4.00**	4.87	–

The best and second best results are highlighted in boldface and italic, respectively
*Indicates 10% sampling of the original dataset
↑Indicates 1% sampling of the original dataset

Table 12 RI of TSD and eight clustering algorithms on benchmark datasets

RI	k-means	CFDP	BCLS	SNN-DPC	FKNN-DPC	DenPEHC	CFSFDP+A	CFSFDP+DE	TSD	TSD-H
Iris	0.7885	0.8923	0.7938	0.7618	*0.9575*	0.7763	**1.0000**	0.8543	0.8651	0.9740
Seeds	0.8347	*0.8843*	0.7679	0.7344	**0.8858**	0.6833	0.6984	0.8017	0.8317	0.9104
Glass	**0.7945**	0.5587	0.2598	0.6647	0.6214	0.2598	*0.7237*	0.5860	0.6713	0.6890
Ecoli	0.7561	*0.7708*	0.2701	0.7383	0.6092	0.4368	0.6425	0.7364	**0.7808**	0.8480
Led7digit	0.9049	0.9151	0.7150	0.9042	0.3140	0.0995	**0.9925**	*0.9244*	0.8980	0.9156
Texture	0.8705	0.4191	0.5238	0.8877	0.8772	0.6571	**0.9180**	0.8181	*0.8888*	0.9256
Ring	**0.5116**	0.5000	0.5000	*0.5006*	0.5000	0.4999	0.5000	0.5000	0.5000	0.5096
Twonorm	*0.8446*	0.5400	0.4999	0.5014	0.5000	0.4999	0.4999	0.5874	**0.9001**	0.9239
Penbased0.1	0.8374	**0.9230**	0.0997	0.8728	0.5337	0.1001	0.7356	0.7062	*0.9067*	0.9076
Magic0.1	0.5414	0.4767	0.5404	0.5098	0.5623	0.5013	0.5419	0.5441	**0.5641**	0.5826
Kr-vs-k0.1	**0.8557**	0.7816	0.1060	0.8167	*0.8260*	0.2034	0.8214	0.4225	0.8118	0.8397
ConfLongDemo0.01↑	*0.7272*	0.7271	0.1899	**0.7323**	0.7270	0.5627	0.6989	0.5321	0.7173	0.7494
DLA0.01↑	**0.7933**	0.7326	0.4754	0.7077	0.7113	0.3619	0.4646	0.4182	*0.7912*	0.7924
Skin0.01↑	0.5039	0.5112	**0.6721**	0.4630	0.6259	0.5101	0.5100	*0.6716*	0.6436	0.7504
Poker0.01↑	0.5549	*0.7867*	0.0769	0.3301	0.1104	0.5377	0.7654	**0.8027**	0.7743	0.8175
Mean rank	*3.47*	4.30	7.13	5.00	4.63	7.90	4.57	4.83	**3.17**	–

The best and second best results are highlighted in boldface and italic, respectively
*Indicates 10% sampling of the original dataset
↑Indicates 1% sampling of the original dataset



(a) Purity **(b)** JC **(c)** FMI **(d)** RI

Fig. 6 Comparison on domain dataset

Table 13 Purity comparison of TSD versus DPCG

Dataset	Algorithms	
	DPCG	TSD
Iris	**0.9600**	0.8727
Wine	**0.8146**	0.7146
Soybean	0.8085	**0.9213**
Ionosphere	0.6752	**0.7578**
Sonar	0.5448	**0.5582**
Seeds	0.7552	**0.8443**
Mean rank	1.67	**1.33**

The best results are highlighted in boldface

Table 14 Purity comparison of TSD versus GDPC, CDPC

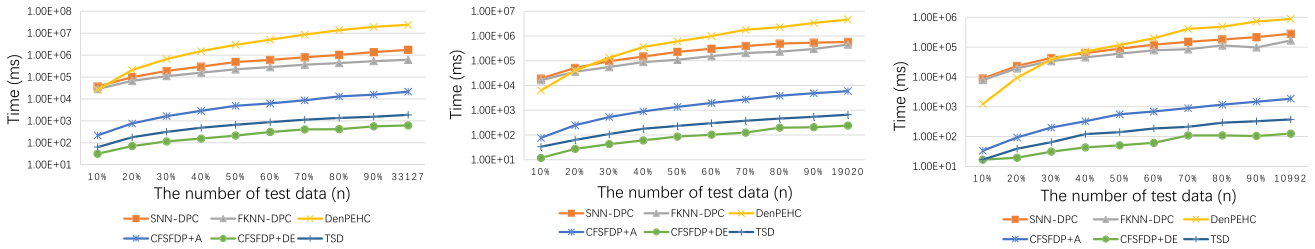
Dataset	Algorithms		
	GDPC	CDPC	TSD
Iris	0.9400	**0.9667**	0.8727
Seeds	**0.8762**	0.8671	0.8443
Waveform	0.6100	0.5864	**0.6149**
Twonorm	0.6977	0.8539	**0.9472**
Pendigits	0.3684	0.4491	**0.6890**
Gamma	0.5110	0.6073	**0.6813**
Mean rank	2.33	2.00	**1.67**

The best results are highlighted in boldface

4.5 Runtime comparison

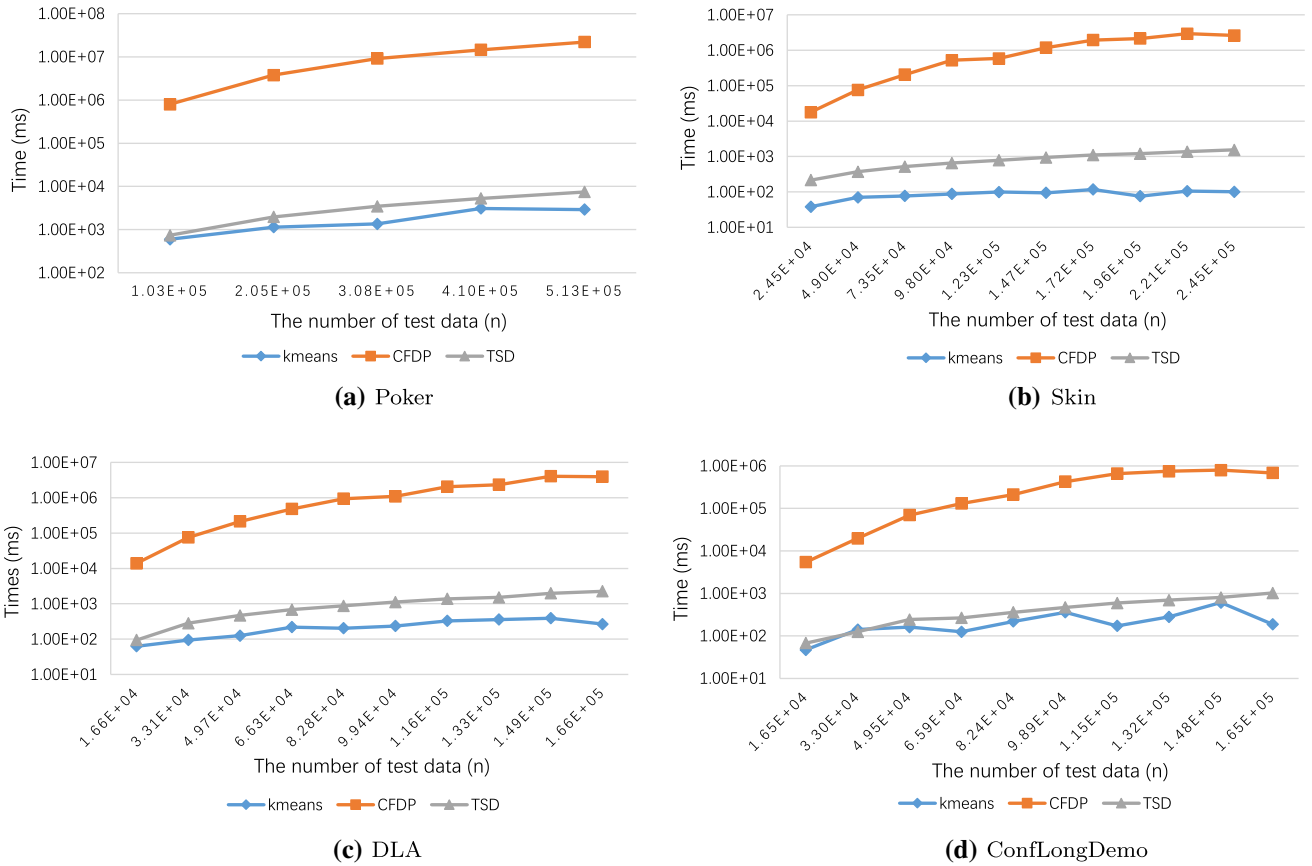
First, we compare the TSD algorithm with several state-of-the-art density peak clustering algorithms on the Matlab platform, including SNN-DPC, FKNN-DPC, DenPEHC, CFSFDP+A and CFSFDP+DE. CFSFDP+A and CFSFDP +DE are two fast optimization algorithms for CFDP (Bai et al. 2017). For the SNN-DPC algorithm, memory overflow occurred when computing datasets with more than 30,000 instances. Therefore, we select three datasets, DLA0.2, Magic and Penbased, to quantify the efficiency of the TSD algorithm, where DLA0.2 is a 20% random sample of DLA.

Figure 7 shows the relationship between the size n of the training dataset and the runtime. Note that we use logarithmic coordinates for the runtime. The results show that the TSD algorithm effectively improves efficiency and reduces running time. It is at least two orders of magnitude faster than SNN-DPC, DenPHEC and FKNN-DPC. For example, on the DLA0.2 dataset with $n = 3.3 \times 10^4$, the runtime for TSD is 1891 ms, compared with 1.7×10^6 ms for SNN-DPC,



(a) DLA0.2 **(b)** Magic **(c)** Penbased

Fig. 7 Runtime comparison on Matlab platform



(a) Poker **(b)** Skin **(c)** DLA **(d)** ConfLongDemo

Fig. 8 Runtime comparison on JAVA platform

6.1×10^5 ms for FKNN-DPC, 2.4×10^7 ms for DenPEHC, 21,561 ms for CFSFDP+A and 624 ms for CFSFDP+DE. That is, TSD is 12,728 times faster than DenPEHC, 923 times faster than SNN-DPC, 323 times faster than FKNN-DPC and 11 times faster than CFSFDP+A. TSD is slightly slower than CFSFDP+DE algorithm.

Second, we use some larger datasets to compare the TSD algorithm with the classical CFDP and k-means algorithms on the JAVA platform. Figure 8 shows the runtime comparison for four large datasets. We use four datasets with more than 160,000 instances, Poker0.5, Skin, DLA and ConfLongDemo, to quantify the efficiency of the TSD algorithm, where Poker0.5 is a 50% random sample of Poker. The number of instances in Poker0.5, Skin, DLA and ConfLongDemo were 512,504, 245,057, 165,633 and 164,860, respectively. The TSD algorithm is two or more orders of magnitude faster than CFDP and is able to process millions of datasets within an acceptable runtime. For example, on the Poker0.5 dataset, with $n = 5.1 \times 10^5$, the runtimes for CFDP and TSD are 2.2×10^7 ms and 7.4×10^3 ms, respectively. That is, TSD is 2692 times faster than CFDP.

The experimental results demonstrate that the TSD algorithm was slightly slower than the k-means algorithm. For example, for the Poker0.5 dataset, with $n = 5.1 \times 10^5$, the runtimes for TSD and k-means are 7.4×10^3 ms and 2.9×10^3 ms, respectively. Thus, k-means is 3 times faster than CFDP.

Table 15 Runtime comparison (s)

Dataset	Algorithms	
	DPCG	TSD
S1	0.5330	**0.0578**
R15	0.5450	**0.0042**
S2	0.7178	**0.0582**
Aggregation	0.1796	**0.0066**
Flame	0.1727	**0.0017**
A2	1.3728	**0.0636**
Twenty	0.7178	**0.0094**
Five-cluster	0.5601	**0.0183**
Iris	0.4531	**0.0011**
Wine	0.4298	**0.0018**
Soybean	0.4455	**0.0006**
Ionosphere	0.5002	**0.0046**
Sonar	0.5802	**0.0036**
Seeds	0.5248	**0.0017**

The best results are highlighted in boldface

Table 16 Runtime comparison (s)

Dataset	Algorithms		
	GDPC	CDPC	TSD
Twenty	0.1975	0.1638	**0.0094**
Forty	0.2284	0.1833	**0.0091**
Five-cluster	2.0201	2.2162	**0.0183**
S2	2.9277	3.3265	**0.0582**
A3	1.6744	2.5849	**0.1360**
A2	4.4183	6.2855	**0.0636**
Iris	0.5999	0.0764	**0.0011**
Seeds	0.5165	0.0839	**0.0017**
Waveform	6.6914	5.5386	**0.1414**
Twonorm	16.4424	23.7432	**0.2647**
Pendigits	30.3098	59.9011	**0.3935**
Gamma	91.8880	110.2128	**0.6912**

The best results are highlighted in boldface

Also, for the DLA dataset, with $n = 1.6 \times 10^5$, k-means is 8 times faster than TSD.

According to Table 3, the time complexity of the TSD algorithm is $O(mn^{\frac{3}{2}})$, while the time complexity for the k-means algorithm is $O(mn)$. The experimental results demonstrate that the theoretical analysis is correct.

Finally, we compare the TSD algorithm on the MATLAB platform with three CFDP optimization algorithms adopting the original results in the reference paper (Xu et al. 2018a, b). Table 15 compares the running time of the TSD and DPCG algorithms. For the 14 datasets list in reference (Xu et al. 2018a), TSD is faster than DPCG algorithm. Table 16 compares the running time of the TSD and GDPC, CDPC algorithms. Similarly, for 12 datasets, TSD is faster than GDPC and CDPC algorithms.

4.6 Optimization of the TSD algorithm

Essentially, TSD is a two-stage hierarchy clustering algorithm. Therefore, we will further optimize the TSD algorithm from two aspects of time and performance. For time optimization, it is important to reduce the complexity of two-round-means. We adopt the idea of grid or wavelet clustering to optimize the two-round-means to reduce the complexity to $O(n)$. In this way, the complexity of the optimized TSD algorithm is $O(n)$.

For performance optimization, the selection of virtual centers is crucial. If initial centers are well adapted to the distribution of the data, we will obtain better performances. Therefore, we design a preprocessing module to make a preliminary estimation on the dataset. The two-stage clustering algorithms will be selected based on preprocessing judgments. In this way, we design an optimization algorithm for TSD, namely TSD-Improved (TSD-I).

Table 17 compares TSD-I with the TSD and CFDP algorithms on synthetic datasets. TSD-I effectively improves the performance of TSD on synthetic datasets. For six synthetic datasets, TSD-I achieved performance improvements in four of them. Table 18 compares TSD-I with the TSD and CFDP algorithms on benchmark datasets. For fifteen synthetic datasets, TSD-I achieved performance improvements in eleven of them.

4.7 Discussions

We are now able to answer the questions posed at the beginning of this work.

(1) The TSD algorithm is more adaptive than popular clustering algorithms. It is effective for datasets of different types, shapes and sizes.
(2) The TSD algorithm is more accurate than classical and state-of-the-art clustering algorithms on benchmark and domain data.
(3) The TSD algorithm is efficient and scalable.

5 Conclusions and future works

In this paper, we proposed a TSD clustering algorithm that was highly efficient, and had good adaptability to multiple types of datasets, with minimal parameter setting. The new algorithm explored the use of two-round-means for pre-clustering so that the time and space complexity could be minimized. The time complexity of the algorithm was

Table 17 Comparison TSD-I with the TSD and CFDP algorithms on synthetic datasets

Dataset	Purity			JC			FMI			RI		
	CFDP	TSD	TSD-I	CFDP	TSD	TSD-I	CFDP	TSD	TSD-I	CFDP	TSD	TSD-I
Flame	**1.0000**	0.8929	0.9917	**1.0000**	0.7409	0.9694	**1.0000**	0.8456	0.9845	**1.0000**	0.8248	0.9834
Spiral	**1.0000**	0.3907	0.4744	**1.0000**	0.2338	0.2499	**1.0000**	0.3832	0.4021	**1.0000**	0.5438	0.5740
Jain	0.8928	0.8515	**1.0000**	0.7135	0.6441	**1.0000**	0.8348	0.7810	**1.0000**	0.8080	0.7392	**1.0000**
Compound	0.8321	0.8191	**0.8947**	0.4874	0.4721	**0.7673**	0.6627	0.6424	**0.8720**	0.8516	0.8336	**0.9283**
R15	**0.9967**	0.8817	**0.9967**	**0.9866**	0.7339	**0.9866**	**0.9975**	0.8433	0.9932	**0.9991**	0.9778	**0.9991**
Aggregation	**0.9987**	0.9292	**0.9987**	**0.9966**	0.6931	**0.9966**	**0.9983**	0.8171	**0.9983**	**0.9993**	0.9261	**0.9993**

The best results are highlighted in boldface

Table 18 Comparison TSD-I with the TSD and CFDP algorithms on benchmark datasets

Dataset	Purity			JC			FMI			RI		
	CFDP	TSD	TSD-I	CFDP	TSD	TSD-I	CFDP	TSD	TSD-I	CFDP	TSD	TSD-I
Iris	0.9067	0.8727	**0.9733**	0.7248	0.6793	**0.9007**	0.8407	0.8088	**0.9478**	0.8923	0.8651	**0.9656**
Seeds	**0.9048**	0.8443	0.8952	**0.7022**	0.6108	0.6803	**0.8251**	0.7566	0.8098	**0.8843**	0.8317	0.8723
Glass	0.4813	**0.5537**	0.5421	0.3082	0.2906	**0.3558**	0.5247	0.4565	**0.5662**	0.5587	**0.6713**	0.6543
Ecoli	0.6786	0.6875	**0.7530**	0.4383	0.3512	**0.5411**	0.6115	0.5256	**0.7032**	0.7708	0.7808	**0.8300**
Led7digit	0.7520	0.6754	**0.7580**	0.4043	0.3519	**0.4088**	0.5758	0.5211	**0.5804**	0.9151	0.8980	**0.9156**
Texture	0.1818	**0.6141**	0.5942	0.1299	**0.4521**	0.3445	0.3538	0.5005	**0.5546**	0.4191	**0.8843**	0.8701
Ring	0.5066	**0.5634**	0.5116	0.4819	0.4060	**0.4999**	0.6820	0.5499	**0.7070**	0.5000	**0.5095**	0.5002
Twonorm	0.6415	0.9472	**0.9601**	0.4400	0.8188	**0.8578**	0.6210	0.9002	**0.9235**	0.5400	0.9001	**0.9234**
Penbased0.1	**0.7352**	0.6695	0.5314	**0.4585**	0.3842	0.2286	**0.6292**	0.5917	0.3821	**0.9230**	0.9067	0.8381
Magic0.1	0.6430	0.6728	**0.7035**	0.2428	**0.6434**	0.5416	0.4044	0.7169	**0.7350**	0.4767	0.5694	**0.5826**
Kr-vs-k0.1	0.2011	0.2332	**0.2346**	0.0891	0.0662	**0.0967**	0.1868	0.1242	**0.2191**	0.7816	0.8118	**0.8397**
ConfLongDemo0.01	0.3277	0.3259	**0.3313**	0.0906	0.0930	**0.1095**	0.1682	0.1720	**0.1974**	0.7271	0.7173	**0.7494**
DLA0.01	0.6872	**0.7613**	0.6902	0.3621	**0.5349**	0.4217	0.5344	**0.7061**	0.5989	0.7326	**0.7912**	0.7581
Skin0.01	0.7935	0.7935	**0.8539**	0.4628	0.4551	**0.7208**	0.6328	0.6256	**0.8445**	0.5112	0.5074	**0.7504**
Poker0.01	0.0914	0.0929	**0.0956**	0.0618	0.0566	**0.0768**	0.1357	0.1165	**0.2743**	0.7867	0.7743	**0.8175**

The best results are highlighted in boldface

$O(mn^{\frac{3}{2}})$, which was lower than $O(mn^2)$ for CFDP. Experiments on 21 datasets demonstrated that the new algorithm was more accurate than state-of-the-art algorithms. It was two or more orders of magnitude faster than CFDP.

From the viewpoint of algorithms and applications, the following research problems merit further investigation:

(1) Revising the TSD algorithm for active learning. The TSD algorithm can greatly reduce the time complexity. Naturally, it can be applied to clustering-based active learning algorithms to improve efficiency.
(2) Nonparametric setting. The TSD algorithm requires the user to provide the final number of clusters. A better solution is to avoid the parameter setting altogether without sacrificing clustering performance.
(3) Adopting other clustering algorithms. We proposed a specific algorithm under the new local–global granule idea. We can replace the two-stage clustering algorithm with others (e.g., DBSCAN (Ester et al. 1996), EM (Langford et al. 2010) and HierarchicalCluster(HC) (Johnson 1967)) under this framework. A comparison study on this issue may be interesting.

Acknowledgements This work was supported by the National Natural Science Foundation of China (41604114); the Sichuan Province Youth Science and Technology Innovation Team (2019JDTD0017).

Compliance with ethical standards

Conflicts of interest The authors declared that they have no conflicts of interest to this work.

References

Bai L, Cheng XQ, Liang JY, Shen HW, Guo YK (2017) Fast density clustering strategies based on the *k*-means algorithm. Pattern Recogn 71:375–386

Blake C, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html
Cattral R, Oppacher F (2007) Discovering rules in the poker hand dataset. In: GECCO, pp 1870–1870
Chang MS, Chen LH, Hung LJ, Rossmanith P, Wu GH (2014) Exact algorithms for problems related to the densest *k*-set problem. Inf Process Lett 114(9):510–513
Chen XL, Cai D (2011) Large scale spectral clustering with landmark-based representation. In: AAAI, p 14
Chen DG, Zhang XX, Li WL (2015) On measurements of covering rough sets based on granules and evidence theory. Inf Sci 317:329–348
Chen M, Li LJ, Wang B, Chen JJ, Pan LN, Chen XY (2016) Effectively clustering by finding density backbone based-on KNN. Pattern Recongn 60:486–498
Chiroma H, Gital AY, Abubakar A, Zeki A (2014) Comparing performances of markov blanket and tree augmented Naïve Bayes on the iris dataset. Lect Notes Eng Comput Sci 2209(1):328–331
Chuang PJ, Yang SH, shin Lin C (2009) An energy-efficient balanced clustering algorithm for wireless sensor networks. In: WiCOM, pp 1–4
Dasgupta S (2002) Performance guarantees for hierarchical clustering. Comput Learn Theory 2375:351–363
Dengel A, Althoff T, Ulges A (2011) Balanced clustering for content-based image browsing. In: Gi-Informatiktage
Du MJ, Ding SF, Jia HJ (2016) Study on density peaks clustering based on k-nearest neighbors and principal component analysis. Knowl Based Syst 99:135–145
Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp 226–231
Guo LJ, Ai CY, Wang XM, Cai ZP (2009) Real time clustering of sensory data in wireless sensor networks. In: Performance computing & communications conference, pp 33–40
Hu BQ, Wong H, Yiu KFC (2016) The aggregation of multiple three-way decision spaces. Knowl Based Syst 98:241–249
Huang CC, Li JH, Mei CL, Wu WZ (2017) Three-way concept learning based on cognitive operators: an information fusion viewpoint. Int J Approx Reason 84(1):1–20
Johnson CS (1967) Hierarchical clustering schemes. Psychometrika 32(3):241–254
Kaufman LJP (2008) Rousseeuw: finding groups in data: an introduction to cluster analysis. Wiley, London
Kriegel HP, Kröger P, Sander J, Zimek A (2011) Density-based clustering. Wiley interdisciplinary reviews: data mining and knowledge discovery 1(3):231–240
Langford J, Zhang XH, Brown G, Bhattacharya I, Getoor L, Zeugmann T (2010) EM clustering. Springer, Boston, MA
Leong SH, Ong SH (2017) Similarity measure and domain adaptation in multiple mixture model clustering: an application to image processing. PLOS ONE 12(7):e0180307
Li YFW, Tsang I, Zhou ZH (2009) Tighter and convex maximum margin clustering. AISTATS 5:344–351
Li JH, Mei CL, Lv YJ (2011) A heuristic knowledge-reduction method for decision formal contexts. Comput Math Appl 61(4):1096–1106
Li JH, Huang CC, Qi JJ, Qian YH, Liu WQ (2016) Three-way cognitive concept learning via multi-granularity. Inf Sci 361(1):1–15
Li XN, Sun BZ, She YH (2017) Generalized matroids based on three-way decision models. Int J Approx Reason 90:192–207
Liang Z, Chen P (2016) Delta-density based clustering with a divide-and-conquer strategy. Elsevier Science Inc., New York
Liu D, Liang DC (2017) Three-way decisions in ordered decision system. Knowl Based Syst 137:182–195
Liu HY, Han JW, Nie FP, Li XL (2017a) Balanced clustering with least square regression. AAA I:2231–2237
Liu YH, Ma ZM, Yu F (2017b) Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy. Knowl Based Syst 133:208–220
Liu R, Wang H, Yu XM (2018) Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Inf Sci 450:200–226
Lu JY, Zhu QS (2017) An effective algorithm based on density clustering framework. IEEE Access 5(99):4991–5000
MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, pp 281–297
Min F, He HP, Qian YH, Zhu W (2011) Test-cost-sensitive attribute reduction. Inf Sci 181:4928–4942
Pappas NT (1992) An adaptive clustering algorithm for image segmentation. IEEE Trans Signal Process 40(4):901–914
Qian J, Lv P, Yue XD, Liu CH, Jing ZJ (2015) Hierarchical attribute reduction algorithms for big data using mapreduce. Knowl Based Syst 73:18–31
Reyes O, Altalhi AH, Ventura S (2018) Statistical comparisons of active learning strategies over multiple datasets. Knowl Based Syst 145:274–288
Rodriguez A, Laio A (2014) Machine learning clustering by fast search and find of density peaks. Science 344(6191):1492–1496
Sarma TH, Viswanath P, Reddy BE (2013) A hybrid approach to speed-up the k-means clustering method. Int J Mach Learn Cybern 4(2):107–117
Shuji S, Masanori K, Takashi I, Yutaka A (2015) Faster sequence homology searches by clustering subsequences. Bioinformatics 31(8):1183–1190
Stenger J (2011) Disability living allowance. Mental Health Today 1(6):277
Ugulino W, Vega K, Velloso E (2012) Wearable computing: accelerometers data classification of body postures and movements. Adv Artif Intell SBIA 2012:52–61
Wang XZ, Musa AB (2014) Advances in neural network based learning. Int J Mach Learn Cybern 5(1):1–2
Wang Y, Jiang Y, Wu Y, Zhou ZH (2011) Spectral clustering on multiple manifolds. IEEE Trans Neural Netw 22(7):1149–1161
Wang M, Min F, Wu YX, Zhang ZH (2017) Active learning through density clustering. Exp Syst Appl 85:305–317
Wilderjans TF, Cariou V (2016) Clv3w : a clustering around latent variables approach to detect panel disagreement in three-way conventional sensory profiling data. Food Qual Prefer 47:45–53
Xie JY, Gao HC, Xie WX, Liu XH, Grant PW (2016) Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. Inf Sci 354:19–40
Xu J, Wang GY, Deng WH (2016) Denpehc: density peak based efficient hierarchical clustering. Inf Sci 373(12):200–218
Xu X, Ding SF, Du MJ, Xue Y (2018a) DPCG: an efficient density peaks clustering algorithm based on grid. Int J Mach Learn Cybern 9(5):743–754
Xu X, Ding SF, Shi ZZ (2018b) An improved density peaks clustering algorithm with fast finding cluster centers. Knowl Based Syst 158:65–74
Yao JT, Yao YY (2002) Induction of classification rules by granular computing. Springer, Berlin
Yu J, Cheng QS (2001) The upper bound of the optimal number of clusters in fuzzy clustering. Inf Sci 44(2):119–125
Yu H, Jiao P, Yao YY, Wang GY (2016) Detecting and refining overlapping regions in complex networks with three-way decisions. Inf Sci 373:21–41
Zhang HR, Min F, Shi B (2017) Regression-based three-way recommendation. Inf Sci 378:444–461
Zhang HR, Min F, Zhang ZH, Wang S (2019) Efficient collaborative filtering recommendations with multi-channel feature vectors. Int J Mach Learn Cybern 10(5):1165–1172

Zhao H, Wang P, Hu QH (2016a) Cost-sensitive feature selection based on adaptive neighborhood granularity with multi-level confidence. Inf Sci 366:134–149

Zhao YX, Li JH, Liu WQ, Xu WH (2016b) Cognitive concept learning from incomplete information. Int J Mach Learn Cybern 7(4):1–12

Zhou ZH (2016) Machine learning. Tsinghua Press, Beijing