Complex & Intelligent Systems (2021) 7:203–223
https://doi.org/10.1007/s40747-020-00191-y

ORIGINAL ARTICLE

Enhanced synchronization-inspired clustering for high-dimensional data

Lei Chen[1] · Qinghua Guo[1] · Zhaohua Liu[1] · Shiwen Zhang[1] · Hongqiang Zhang[1]

Received: 16 May 2020 / Accepted: 17 August 2020 / Published online: 3 September 2020

Abstract

The synchronization-inspired clustering algorithm (Sync) is a novel and outstanding clustering algorithm, which can accurately cluster datasets with any shape, density and distribution. However, the high-dimensional dataset with high dimensionality, high noise, and high redundancy brings some new challenges for the synchronization-inspired clustering algorithm, resulting in a significant increase in clustering time and a decrease in clustering accuracy. To address these challenges, an enhanced synchronization-inspired clustering algorithm, namely SyncHigh, is developed in this paper to quickly and accurately cluster the high-dimensional datasets. First, a PCA-based (Principal Component Analysis) dimension purification strategy is designed to find the principal components in all attributes. Second, a density-based data merge strategy is constructed to reduce the number of objects participating in the synchronization-inspired clustering algorithm, thereby speeding up clustering time. Third, the Kuramoto Model is enhanced to deal with mass differences between objects caused by the density-based data merge strategy. Finally, extensive experimental results on synthetic and real-world datasets show the effectiveness and efficiency of our SyncHigh algorithm.

Keywords Synchronization-inspired · Clustering · High-dimensional dataset · Local density

Introduction

Clustering uses an unsupervised way to uncover the hidden rules and patterns of human society; it is an indispensable mean to mine the complex real-world data [1]. Over the past few decades, a large number of excellent clustering algorithms have been proposed and expanded, and have demonstrated their power in various fields, such as transportation, meteorology, biology, and so on [2]. However, with the advent of the era of big data, complex data in various applications have hundreds of thousands of dimensions, and are characterized by high noise, irregularity and imbalance [3]. In addition, the data dimension is getting higher and higher, showing exponential growth. Faced with these new features, traditional clustering algorithms perform poorly and are unsatisfactory. The main reasons are as follows: (1) complex data are in a high-dimensional space and are difficult to process; (2) there are a lot of redundancy and noise attributes in high-dimensional data; (3) the distribution of data is uneven, and the datasets present various irregular shapes; and (4) a lot of outliers are hidden in high-dimensional data.

To complete the high-dimensional data clustering, existing methods are mainly divided into the following two categories. The first category is subspace clustering, which first sees each dimension as a subspace to perform local clustering, and then integrates all local clustering results in different subspaces to obtain the final result based on the local correlation. The core of this algorithm is to find the appropriate local correlation between different subspaces. For example, Agrawal et al. [4] weighed each dimension to determine the correlation of different subspaces; Chen et al. [5] further divided high-dimensional attributes into differ-

✉ Qinghua Guo
timelessstar.gqh@mail.hnust.edu.cn

Lei Chen
chenlei@hnust.edu.cn

Zhaohua Liu
163liuzhaohua@163.com

Shiwen Zhang
544085870@qq.com

Hongqiang Zhang
1040080@hnust.edu.cn

[1] School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan, China

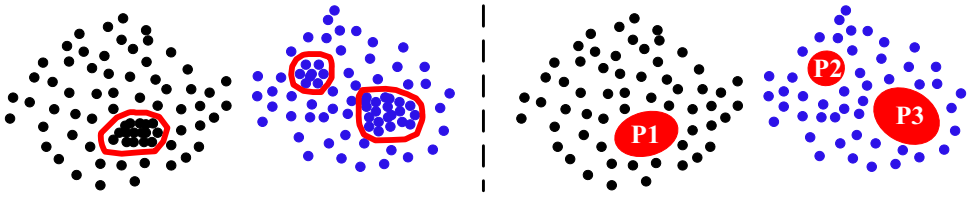

Fig. 1 Illustration of density-based data merge strategy

(a) Synthetic dataset **(b)** Density-based merging data

ent feature groups, and set different weights on the feature groups to determine the relevance of different subspaces; Yan and Lakshmi used multi-view [6] and rough set [7], respectively, to measure the local correlation between different subspaces. However, this algorithm cannot find outliers well and is also sensitive to data distribution. The second category is projected-based clustering, which first projects high-dimensional data into low-dimensional space, and then uses a traditional clustering algorithm to perform clustering in low-dimensional space. For example, PCA(Principal Component Analysis) + K-means [8], LLE(Locally Linear Embedding) + SpectralClustering [9], T-SNE(t-distributed stochastic neighbor embedding) + DbScan [10]. Generally speaking, PCA, LLE, Fisher, LDA (Latent Dirichlet Allocation), SNE, T-SNE are widely used dimensionality reduction techniques. Among them, the accuracy of the PCA algorithm is slightly worse than the LLE and T-SNE algorithms, but the time efficiency is the best. The accuracy of the LLE and T-SNE algorithms is the best, but the time complexity is very high. Similarly, Kmeans, SpectralClustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and DynamicClustering are the most commonly used data clustering algorithms. Among them, Kmeans and SpectralClustering have better time efficiency, but are sensitive to noise and data distribution. The robustness and clustering accuracy of DBSCAN and DynamicClustering algorithms are better, but the time complexity is higher. In summary, under the big data environment, the existing high-dimensional data clustering algorithm has the following problems: (1) the time complexity is high, which cannot meet the real-time requirements of big data applications; (2) the clustering results of the datasets with irregular shapes and uneven distributions are not satisfactory; and (3) the outliers in the data set cannot be well identified.

Based on the above challenges, the motivation of this paper is to find an efficient and accurate high-dimensional data clustering algorithm, which can quickly find clustering results and outliers from the high-dimensional datasets with different shapes and different distributions under the big data environment. According to the above motivation, synchronization-inspired clustering algorithm (Sync) [11] is chosen as the basic clustering algorithm after careful consideration. Because the Sync algorithm is a novel dynamic clustering algorithm, it can accurately cluster different datasets with arbitrary shape, density and distribution [12]. However, high time complexity and high-dimensional attributes with high noise and high redundancy are new challenges for Sync algorithm to cluster high-dimensional data. To address these challenges, this paper proposes an enhanced synchronization-inspired clustering for high-dimensional data, simply called SyncHigh. First, for removing the noise and redundancy of high-dimensional attributes, PCA method is used as preprocessing to find the core attributes and speed up the dimensionality reduction. Second, for further decreasing the time complexity, the idea of density is introduced into the SyncHigh algorithm, and a density-based data merge strategy is proposed to reduce the number of data objects participating in the synchronization-inspired clustering algorithm. Figure 1 shows the basic idea of density-based data merge strategy. Figure 1a depicts the distribution of a synthetic dataset with 2 clusters, where points with same color belong to the same cluster. It is not difficult to find that there are multiple dense regions in the dataset, as shown by the red solid circle. Each dense region contains a large number of data points, and these data points must belong to the same cluster. That is to say, synchronization-inspired clustering algorithm does not need to spend any time to determine the class labels of these dense data, they can be merged into a "big" data object with bigger mass, as shown by the red big point in Fig. 1b. After merging these regions, the amount of data participating in the synchronization-inspired clustering algorithm will be greatly reduced, the number of data objects participating in the clustering process will be greatly reduced, and the time complexity of the clustering algorithm will be greatly improved.

In summary, the main contributions of this paper are listed as follows:

- PCA method is used as a preprocessing to quickly purify dimensions of high-dimensional data, by removing noise and redundant attributes.
- A density-based data merge strategy is developed to reduce the amount of data participating in the synchronization-inspired clustering process and, thus, improve the time efficiency, by merging all points in each density region into one big point with bigger mass.
- An improved Kuramoto model is designed to replace the traditional Kuramoto model for processing the data points

with high weight formed by the density-based data merge strategy.
- An enhanced synchronization-inspired clustering algorithm (SyncHigh) is proposed to cluster high-dimensional data, by integrating above three contributions.

The remainder of this paper is organized as follows: section "Traditional synchronization-inspired clustering (Sync)" shows some related preliminary knowledge about Kuramoto Model and Sync algorithm. Section "Enhanced Synchronization-Inspired Clustering for High-dimensional data" presents the details of the enhanced synchronization-inspired clustering algorithm (SyncHigh). Extensive experimental evaluation is stated in section "Experimental Evaluation". Finally, the conclusion of this paper is summarized.

Traditional synchronization-inspired clustering (Sync)

Synchronization is a universal phenomenon hidden in the development and evolution of nature. The way it behaves is that similar things naturally cluster together to form a community. Inspired by the idea of synchronization, lots of synchronization-inspired clustering algorithms have been proposed [13, 14], Sync algorithm is a typical case [11]. This algorithm can accurately cluster datasets of any shape, size, and density without any previous distribution assumptions. Meanwhile, this algorithm can effectively find abnormal data and noise. The Kuramoto model is a common clustering model for the synchronization-inspired clustering algorithm (Sync). In this section, the details of the Kuramoto model and the Sync algorithm are introduced separately.

Kuramoto model

For a given dataset DS, it contains N data objects, and each object x consists of M dimensions $x = [x_1,\ldots,x_i]$, $i \in (1,\ldots,M)$. The Kuramoto Model is used to cluster DS to form multiple communities. This model treats each object x as a coupling oscillator, and the M-dimensional attributes of object x are regarded as the initial phase θ. Through a dynamic coupling interaction processing, each object x interacts dynamically with its neighbors to gradually change the initial phase θ. After multiple interactions, objects with similar attributes have the same phase, and objects with different attributes have different phase. Finally, the objects are easily grouped according to its phase information.

Definition 1 (ε-*neighborhood of an object x*). In Kuramoto Model, each object x interacts with its ε-neighborhood. The ε-neighborhood of object x is defined as:

$$Nb_\varepsilon(x) = \{y \in DS \mid \mathrm{dist}(y, x) \le \varepsilon\}, \tag{1}$$

where DS is the dataset, x and y are two different objects in DS, dist(y,x) is the distance between y and x (Euclidean distance is default), ε is a threshold called the neighborhood radius.

Based on the ε-neighborhood, the M attributes of each object x are seen as the initial phase, and participate in a dynamic coupling interaction process. The dynamic coupling process of i-th dimension x_i of object x is as follows:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \omega + \frac{S}{|Nb_\varepsilon(x)|} \sum_{y \in Nb_\varepsilon(x)} \sin(y_i - x_i). \tag{2}$$

Let $\mathrm{d}t = \Delta t$, then:

$$x_i(t+1) = x_i(t) + \Delta t \cdot \omega + \frac{\Delta t \cdot S}{|Nb_\varepsilon(x(t))|} \sum_{y \in Nb_\varepsilon(x(t))} \sin(y_i(t) - x_i(t)). \tag{3}$$

In the above formula, ω is the frequency of the object x. To simplify, Bohm [11] assumes that all objects have a common frequency ω, and can safely ignore the term $\Delta t \cdot \omega$. S is a constant indicating the coupling strength between all points. Similarly, $\Delta t \cdot S$ is also a constant for all objects, and can be simplified to 1. Therefore, the dynamic coupling process of i-th dimension x_i of object x is expressed as:

$$x_i(t+1) = x_i(t) + \frac{1}{|Nb_\varepsilon(x(t))|} \sum_{y \in Nb_\varepsilon(x(t))} \sin(y_i(t) - x_i(t)). \tag{4}$$

It is worth noting that the dynamic coupling interaction process will perform multiple iterations. In each iteration, each dimension of each object will be dynamically coupled with its ε-neighborhood as shown in the formula above.

To determine whether the dynamic coupling process is over, a local order parameter rc is introduced to measure the phase coherence of all objects, and defined as:

$$rc = \frac{1}{|DS|} \sum_{x \in DS} \frac{1}{|Nb_\varepsilon(x)|} \sum_{y \in Nb_\varepsilon(x)} e^{-\|y-x\|}. \tag{5}$$

When rc converges to 1, it means all objects with similar attributes are coupled together. Therefore, the local order parameter rc indicates the end of the dynamic coupling process in the Kuramoto Model.

Synchronization-inspired clustering (Sync)

According to the Kuramoto model, a synchronization-inspired clustering algorithm, called Sync [11], is developed to cluster datasets with any shape, density and distribution. This algorithm is very flexible, and can find noise and abnormal objects in the dataset. The basic idea of the Sync algorithm is as follows. Each object in the dataset is regarded

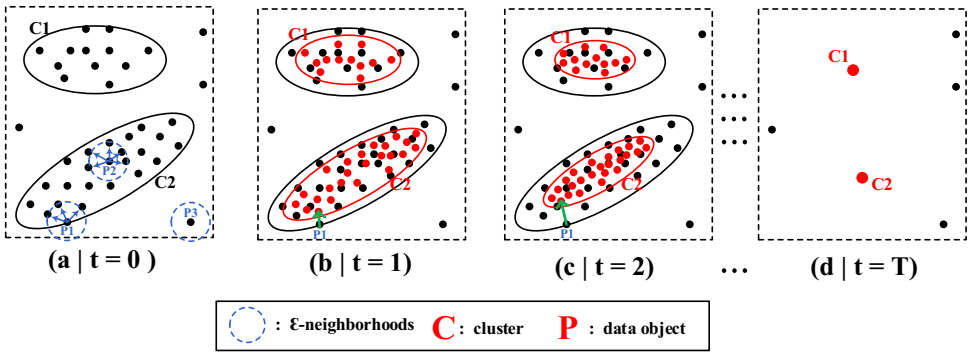

Fig. 2 Dynamic coupling interaction process of the Sync algorithm. **a** Initial state of all objects ($t = 0$): the blue circles are the ε-neighborhood of three samples. **b**, **c** Coupling changes of object's phase state in one iteration ($t = 1$ or $t = 2$): the black and red points are the state before and after dynamic coupling, respectively. **d** Final state of all objects

as an oscillator, and the M attributes are regarded as the initial phase. Then, a dynamic coupling interaction process is started to perform multiple iterations. In each iteration, each dimension of each oscillator dynamically couples with its ε-neighborhood, thereby changing its own phase value. After each iteration, the local order parameter *rc* of all oscillators increases gradually. Through many iterations, the *rc* converges to 1, the Sync algorithm is over. The dynamic coupling interaction process of the Sync algorithm is described in Fig. 2.

In the figure, a two-dimensional dataset is displayed, and consists of two clusters C1 and C2. In the initial step ($t = 0$), each point is regarded as an oscillator, and the two dimensions are set as the initial phase. Meanwhile, each oscillator draws its ε-neighborhood. From $t = 0$ to $t = T - 1$, a dynamic coupling process is started to go through multiple iterations to change the initial phase, as shown in Fig. 2b, c. In each iteration, some oscillators with similar attributes gradually move closer, and the distance between them gradually decreases. When $t = T$, the local order parameter *rc* converges to 1, all oscillators are separated into two clusters C1, C2 and some outliers, as shown in Fig. 2d. It is worth noting that there are three kinds of data points in the dynamic clustering process, as shown in P1, P2 and P3 of Fig. 1a. The P1 is a boundary point; its neighbors are located on one side. With the dynamic clustering process, this point quickly moves closer to its neighbors; P2 is a center point, and has more neighbors. Its neighbors are located around it. During dynamic clustering process, P2 moves relatively slowly; P3 is a noise with no neighbors around it. In the dynamic clustering process, this point stays almost unchanged in its place.

In summary, the procedure of the Sync algorithm is as follows. For simplicity, if you need more detailed pseudo code, please refer to Ref. [11].

Step 1 At initial time ($t = 0$), without any interaction, all objects in the dataset have their own initial phases. According to a threshold ε, each object marks its ε-neighborhood

Step 2 Start a dynamic coupling interaction process (from $t = 1$ to $t = T$). As time evolves, based on Eq. 4, each object interacts dynamically with its ε-neighborhood to change its phase value. Objects with similar attributes gradually move closer, objects with different attributes move away from each other, and the outliers stay almost in its place. After multiple iterations, all objects are naturally divided into multiple clusters and outliers

Step 3 Finally, when the local order parameter *rc* converges to 1, the dynamic clustering process and the Sync algorithm are over

Enhanced synchronization-inspired clustering for high-dimensional data

In the era of big data, complex dataset is characterized by irregularity, high noise, and unevenness. The synchronization-inspired clustering algorithm is very suitable for these characteristics to complete data clustering. However, high dimensions and high time complexity are new challenges for the synchronization-inspired clustering algorithm to cluster high-dimensional data. Therefore, we propose an enhanced synchronization-inspired clustering algorithm, namely SyncHigh, to cluster high-dimensional data in this section. In "PCA-based dimension purification", a PCA-based dimensional purification strategy is designed to remove noise and redundancy of high-dimensional attributes. In "Speeding up synchronization clustering based on local density", a density-based data merge strategy is developed to speed up the clustering time. Combining the above two strategies, the SyncHigh algorithm is described in detail in "SyncHigh-Enhanced Synchronization-Inspired Clustering for high-dimensional data".

PCA-based dimension purification

High dimensions, together with irregularity, unevenness and high noise, have become the basic features of current com-

plex datasets. These features have brought great challenges to traditional clustering algorithms, resulting in a serious decline in clustering quality. The main reasons are as follows. First, high dimensions of dataset increase the time and space complexity of the clustering algorithm. Second, irregular and uneven data distribution makes it impossible to accurately select clustering models and algorithms. Third, some redundancy and noise attributes may have a negative impact on the clustering algorithm and severely weaken the clustering accuracy. Therefore, the dimension purification of high-dimensional dataset is very important.

The high time complexity of the synchronization-inspired clustering algorithm is the core optimization goal in this paper. When choosing a dimensional purification method that can perfectly embed the synchronization-inspired clustering algorithm, we need to focus on time complexity, and appropriately weaken the precision of dimensional purification. When comparing the PCA, LLE, LDA, SNE, T-SNE algorithms, we found that the accuracy of dimensional purification of PCA is in the middle, but the time efficiency is the fastest. Therefore, in this paper, PCA is selected as the preprocessing method of the synchronization-inspired clustering algorithm for dimensional purification of high-dimensional dataset.

The basic process of PCA method is as follows [15]. First, the high-dimensional dataset is formalized as a matrix where each row is a data object and each column is a dimension. Second, the matrix is regularized row by row to eliminate differences in dimensional distribution. Then, the covariance between any two dimensions is calculated to form a covariance matrix. Next, the largest k eigenvalues and eigenvectors of the covariance matrix are selected as the k principle components of high-dimensional dataset. Lastly, the product of the k eigenvalues and the original matrix is the final low-dimensional dataset. However, PCA-based dimension reduction strategy still has one problem needs to be optimized. In the PCA algorithm, the parameter k needs to be set manually and is not robust to various datasets. In the real world, due to different data distribution, each high-dimensional dataset has a different number of principal components; it may be 5, or 50, or even hundreds or thousands. In other words, we cannot use a uniform k value to select the principal components of all high-dimensional datasets.

To optimize the above problem, this paper attempts to find a more suitable and more general solution to replace the parameter k. In the PCA, the largest k eigenvalues are regarded as principal components. This means that we can use the eigenvalues to determine the importance of each dimension in the high-dimensional dataset. Therefore, based on the eigenvalues, a new principal component score of each dimension is designed as shown in Eq. 6. Based on this score, a principal component holding rate sc is proposed as a new parameter to replace the traditional parameter k, as shown in Eq. 7. This new parameter is also a threshold, which is the sum of the maximum principal component score in the high-dimensional dataset. Comparing the two parameters, it is not difficult to find that the parameter sc is more general and robust than the traditional parameter k, and it is not sensitive to the distribution and density of a high-dimensional dataset. For example, the dataset DS1 has 1000 dimensions where 10 dimensions are the principal components, and the sc value of 10 dimensions is 95%. The dataset DS2 has 2000 dimensions where 300 dimensions are the principal components, and the sc value of these dimensions is also 95%. If the traditional parameter k is used, we need to set $k = 10$ and $k = 200$ for DS1 and DS2. Moreover, in the real world, we cannot pre-know the number of principal components. However, if the new parameter sc is used, we only need to set a threshold 0.95 for two datasets.

Definition 2 (*principal component score of a dimension*) Principal component score refers to the importance of each dimension in all attributes, and is defined as:

$$\text{PC-score}(i) = \frac{\text{eigenvalue}_i}{\sum_j \text{eigenvalue}_j}\, i, j \in [1, M], \tag{6}$$

where i or j is a dimension of high-dimensional data, eigenvalue_i is the eigenvalue of i-th dimension of the covariance matrix.

Definition 3 (***principal component holding rate of high-dimensional dataset***) Based on the descending order of PC score, principal component holding rate sc is a threshold to select the data attribute with the highest PC score, and is defined as:

$$\text{PC} = \left\{ i \in \underset{i\in[1,M]}{\text{DESC}}[\text{PC-score}(i)] \,\middle|\, \sum_{i\in[1,M]} \text{PC-score}(i) \le sc \right\}, \tag{7}$$

where i is a dimension of high-dimensional data, M is the total number of dimensions, sc needs to be set manually, and the range usually is [0.8–0.95].

According to the above optimization strategy, a PCA-based dimension purification method is designed, as shown in Algorithm 1.

Algorithm 1: PCA-based dimension purification

Input: (i) high-dimensional dataset *DS*, (ii) the threshold *sc*.
Output: low-dimensional data matrix after dimensionality reduction *LDM*.
Procedure: pca_dimension_purification(*DS*, *sc*):

1: Building an N× M data matrix *DM* from *DS*, each row is a data object and each column is a dimension.
2: Performing zero-mean processing for each row of *DM*, each row minus the mean of each dimension.
3: Generating the covariance matrix *CM* of the matrix *DM*, and getting the covariance between any two dimensions.
4: Calculating all eigenvalues and eigenvectors of the covariance matrix *CM*, and calculating the principal component score *PC-score* of each eigenvalue according to Equation 6.
5: Selecting top dimensions with the largest *PC-score* as the principal component, according to Equation 7.
6: Producing the final low-dimensional matrix *LDM* by the product of the eigenvalues of selected principal components and the original matrix *DM*.
7: return *LDM*.

Speeding up synchronization clustering based on local density

High time complexity is a prominent challenge of synchronization clustering algorithm to cluster high-dimensional data. For this, a density-based data merge strategy is proposed in this paper to speed up the dynamic clustering process. In the real world, irregular and uneven complex dataset usually has multiple dense regions, as shown by the red solid circle in Fig. 1a. Each dense region usually has many data points, and these data points must belong to the same cluster. That is to say, these data points can determine their true label without going through any clustering algorithm. Therefore, merging each dense region into a 'big' point with bigger mass can greatly reduce the time complexity of the synchronization clustering algorithm, as shown in Fig. 1b. Based on this strategy, the clustering algorithm can be speeded up in two ways. First, by merging each dense region into one bigger-mass point, the amount of data participating in the dynamic clustering algorithm will be greatly reduced. Second, after merging dense regions, the neighbors of each data point that need to be interacted in the dynamic clustering process will be greatly reduced. To achieve the above strategy, two problems need to be solved.

(1) How to find dense regions? The purpose of merging dense regions is to optimize the time complexity of the synchronization-inspired clustering algorithm. Therefore, the discovery scheme of dense region must fit the process of synchronization-inspired clustering algorithm. At the beginning of the synchronization clustering algorithm, each data point needs to determine its Ɛ-neighborhood based on the distance between data points. That is to say, the distance between any two points is an effective way to measure dense region, and it is very consistent with the synchronization-inspired clustering algorithm. For that, we defined local density to measure whether two data points belong to the same dense region.

Definition 4 (*local density*). The goal of local density is to determine whether two points are in the same dense region. It is defined as:

$$\text{Local\_density}(i\ ,\ j) = \begin{cases} 1, & \text{dist}(i\ ,\ j) \le \text{d}t \\ 0, & \text{dist}(i\ ,\ j) > \text{d}t \end{cases}, \tag{8}$$

$$\text{d}t = \text{d}c \times \text{Average}(\text{dist}(i\ ,\ j)),\ [i, j \in N], \tag{9}$$

where i and j are two points, dist(i, j) is the distance between i and j, dt is a distance threshold. To avoid differences in the data distribution of different datasets, we need to optimize dt and increase its robustness to adapt to all datasets. Therefore, we use the average distance of all data points as the basis, and use another parameter dc, ranging from 0 to 1, to increase the robustness of dt, as shown in Eq. 9. Here, we believe that if the distance between two points is greater than the average distance of all points, then these two points are definitely not dense points. Through the following experimental analysis, the optimized value range of the parameter dc is [0–0.15].

Based on the local density, it is very easy to determine whether two data points belong to the same dense region, so as to find all dense regions in the dataset, as shown in Fig. 3a. In the figure, there is a dense region circled by the red solid line. If we enlarge this dense region, there is a shortest path connecting all the data points in the region, and the distance between adjacent nodes on the path is less than or equal to dt. That is to say, as long as all data points with distances less than or equal to dt are found, the dense area of the data set will also be found naturally.

(2) How to make the merged "big" data point perfectly embedded in the synchronization-inspired clustering algorithm? When dense regions are found, we merge each dense region into a "big" data point with bigger mass, thereby reducing the time complexity of the synchronization clustering algorithm. However, the merged "big" data point has different mass and different neighbors compared to the reg-

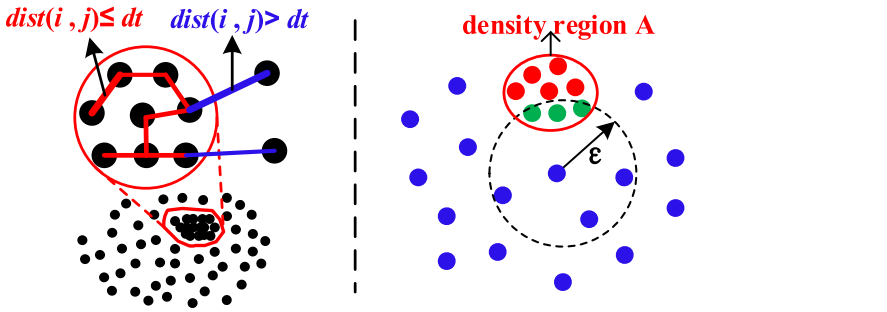

Fig. 3 Illustration of dense region

(a) Finding dense region

(b) Compressing the Ɛ-neighborhood of one node based on the dense region

ular data point. To solve this problem, two aspects should be considered.

Each merged "big" data point must dynamically interact with its ε-neighborhood. Although all data points in a dense region belong to the same label, it cannot be guaranteed that each dense region is a separate cluster. In the other words, multiple "big" data points may belong to a cluster. Therefore, each merged "big" data point needs to be embedded into the synchronization clustering process for getting the final class label through dynamic interaction with its ε-neighborhood. For this purpose, we need to define new attributes and ε-neighborhood for a dense region.

Definition 5 (*dense region*). A dense region dr is a merged "big" data point, and its attributes are defined as:

$$\begin{aligned} \mathrm{d}r &= \langle \dim,\ \mathrm{mass} \rangle, \\ \mathrm{mass} &= \| \mathrm{d}r \|, \\ \dim &= \left\{ \mathrm{avg}\left(\dim_{ij}\right) \,\middle|\, i \in [1, M],\ j \in [1 \| \mathrm{d}r \|] \right\}, \end{aligned} \tag{10}$$

where the attributes of dr consist of dimension dim and quality mass; the term mass indicates the number of data points in the dr; the term dim represents the average value of each dimension of all data points in the dr, $\dim_{ij}$ is the i-th dimension of the j-th data point.

Definition 6 (Ɛ-*neighborhood of a dense region*) The ε-neighborhood of a dense region dr is the union of the ε-neighborhood of all points in the dr (excluding all points of dr). It is defined as:

$$Nb_{\varepsilon}(\mathrm{d}r) = \left\{ y \in \bigcup_{x \in \mathrm{d}r} Nb_{\varepsilon}(x) \text{ and } y \notin \mathrm{d}r \right\}. \tag{11}$$

Based on the above definitions, each dense region can be regarded as a merged "big" data point to participate in the synchronization-inspired clustering algorithm.

The ε-neighborhood of each regular data point needs to be updated and compressed after the dense regions are merged. It is inevitable that multiple neighbors of one regular data point are in the same dense region. When this dense region is merged as a "big" data point, we need to update the ε-neighborhood of this data point and compress the neighbors belonging to the same dense region, so as to reduce the amount of neighbors of this regular data point, as shown by Fig. 3b. In the figure, the ε-neighborhood of a regular data point (black dotted circle) contains 6 neighbors and intersects with a dense region (red solid circle) at 3 green data points. Since the 3 green points belong to the same cluster, these 3 green neighbors with a mass of 1 can be compressed into a neighbor with a mass of 3. Through this strategy, the number of ε-neighborhood of each data point will be greatly reduced, thereby reducing the time overhead of each iteration in the dynamic interaction process of the synchronization clustering algorithm.

Based on the above optimizations, a density-based data merge strategy is proposed to speed up the synchronization-inspired algorithm. The main process is as follows, and the pseudo code is shown in Algorithm 2:

Step1, calculating the ε-neighborhood and local density for each data point. The distance between any two data points is first calculated based on the Euclidean distance. When the distance is less than or equal to ε, then the two data points are neighbors to each other and join each other's ε-neighborhood. When the distance is less than or equal to dt, the two data points are locally dense and belong to the same dense area. When all distances are calculated, the ε-neighborhood of each point is identified, and all dense points are also found.

Step2, Finding and merging the dense regions. When all dense points are found, multiple independent shortest path trees can be quickly found by judging whether there are common data points, and each shortest path tree is a dense region. After finding all the dense regions, the attributes and ε-neighborhood of each dense region are calculated based on Eqs. 10 and 11 respectively.

Step3, Compressing the ε-neighborhood of each regular data point. Based on all dense regions, the ε-neighborhood of each data point is compressed to reduce the number of neighbors, so as to speed up the time overhead of the

synchronization-inspired clustering algorithm. Finally, the optimized dataset and ε-neighborhood of each data point are used as the input of the synchronization clustering algorithm to find the final clustering result.

the same mass or weight, and the coupling force between two objects is the same.

$$x_i(t+1) = x_i(t) + \frac{1}{|Nb_\varepsilon(x(t))|}\sum_{y\in Nb_\varepsilon(x(t))}\sin(y_i(t) - x_i(t)). \tag{12}$$

Algorithm 2: density-based data merge strategy

Input: (i) the dataset *DS*, (ii) the threshold *dc*, (iii) the threshold ε.
Output: the optimized dataset *ODS* and ε-neighborhood of each data point *Nb*.
Procedure data_merging(*DS*, *dc*, ε):

```
Initialize dense region list drlist=null, ODS=DS, Nb=null.
  //step 1
For any i , j in DS:
        //calculate ε-neighborhood
        If dist(i,j) ≤ ε:
            insert i into Nb_ε(j);
            insert j into Nb_ε(i);
        //calculate local density
        If dist(i,j) ≤ (dc*AVERAGEDISTANCE):
            insert <i,j> into drlist;
End
  //step 2
For any <i,j> and <k,r> in drlist:
        If i==j or i==r or j==k or j==r:
            insert <i,j,k,r> into drlist, and remove <i,j> and <k,r> from drlist;
            calculate attributes and ε-neighborhood of <i,j,k,r>;
End
remove all points in drlist from ODS;
insert the merged "big" data into ODS;
  //step 3
For any Nb_ε(i) in Nb:
        compress the Nb_ε(i) based on the drlist;
End
```

SyncHigh-enhanced synchronization-inspired clustering for high-dimensional data

Enhanced Kuramoto model

The traditional Kuramoto Model is a data model, which is abstracted from the synchronization phenomenon in nature. The core idea is that similar objects in nature will attract each other and produce coupled motion. Similar objects reach a synchronized state after multiple coupled motions, and all objects have the same phase value. Dissimilar objects have larger and larger phase differences after multiple coupling movements. Equation 4 describes the definition of a dynamic coupling motion of two objects. It should be noted that the traditional Kuramoto Model assumes that each object has

However, after using the density-based data merge strategy, the mass of different data points in a dataset is not uniform, and the mass of different neighbors of one data point is not the same. For example, the mass of a merged "big" data point is 10, 20, or 30, and the mass of a regular data point is 1; for one data point, one neighbor has a mass of 5 and the other has a mass of 1. The different mass of data points makes the traditional Kuramoto Model unsuitable. Therefore, an enhanced Kuramoto Model is proposed in this paper to handle this problem. Generally speaking, data points with bigger mass have more influence on the neighbors than data points with lower mass. So, we optimize the dynamic coupling function between a data point and its neighbors, and consider the mass difference of data points in each dynamic interaction process. The optimized dynamic coupling function is defined as follows:

$$x_i\,(t+1) = x_i\,(t) + \frac{1}{\text{mass}\,(x)} \times \frac{\sum_{y\in Nb_\varepsilon(x(t))} \sin\,(y_i\,(t) - x_i\,(t)) \times \text{mass}\,(y)}{\sum_{y\in Nb_\varepsilon(x(t))} \text{mass}\,(y)}, \tag{12}$$

where mass(x) is the mass of object x, x_i is i-th dimension of object x. When the mass of all objects is 1, the above equation can be simplified to Eq. 4.

In addition, the mass difference of data points also affects the judgment of the current clustering state. Therefore, to determine if the dynamic coupling process is over, a new local order parameter rc_{new} is defined as:

$$rc_{\text{new}} = \frac{1}{\sum_{x\in DS} \text{mass}(x)} \sum_{x\in DS} \frac{1}{\sum_{y\in Nb_\varepsilon(x(t))} \text{mass}(y)} \sum_{y\in Nb_\varepsilon(x)} \left(e^{-\|y-x\|} \times \text{mass}(y)\right), \tag{13}$$

when rc_{new} converges to 1, it means that all objects with similar properties are coupled together, and indicates the end of the dynamic clustering process. When the mass of all objects is 1, the above equation can be simplified to Eq. 5.

SyncHigh algorithm

In summary, by integrating the above strategies, an enhanced synchronization-inspired clustering algorithm for high dimensional data, called SyncHigh, is proposed in this paper. The main process of SyncHigh algorithm is as follows, the pseudo code is shown in Algorithm 3.

Step1, PCA-based dimension purification. The original high-dimensional dataset and parameter *sc* are used as inputs, and the optimized PCA is used to perform dimensional purification to remove redundant and noise dimensions for getting the low-dimensional dataset. The process of PCA-based dimensional purification is shown in Algorithm 1.

Step2, density-based data merge. Taking the purified low-dimensional dataset, parameters ε and d*c* as inputs, all dense regions are discovered, each dense region is merged into a "big" data point with bigger mass, and the ε-neighborhood of each data point is updated and compressed, thereby reducing the amount of data participating in the next dynamic interaction process. The process of density-based data merge strategy is shown in Algorithm 2.

Step3, dynamic interaction based on the enhanced Kuramoto Model. Taking the optimized dataset and the compressed ε-neighborhood as inputs, the attributes of each data are regarded as its initial phase. Based on the enhanced Kuramoto Model, a dynamic interaction process is started and gradually changes the phase value of each data point through Eqs. 12 and 13. After multiple iterations, the data points belonging to the same cluster will have the same phase, and the local order parameter rc_{new} converges to 1, then the dynamic interaction process and the SyncHigh algorithm are over.

Algorithm 3: Enhanced Synchronization-Inspired Clustering for High-dimensional Data

Input: (i) high-dimensional dataset *DS*, (ii) parameter *sc*, (iii) parameter ε, (iv) parameter *dc*.
Output: final clustering results *rs*.
Procedure SyncHigh(*DS*, *sc*, ε, *dc*)

```
//step 1 dimension purification
Low-dimensional dataset LDS = pca_dimension_purification(DS, sc).
//step 2 merging dense regions
The optimized dataset ODS, compressed ε-neighborhood Nb= data_merging(LDS, dc, ε).
//step 3 dynamic interaction
While (r_Cnew < 1- Δ ):
    For x in ODS:
        getting its ε-neighborhood Nb_ε(x) from Nb.
        using the equation 12 to calculate new phase of data x based on the Nb_ε(x).
    EndFor
    calculating the new round of local order parameter rc_new
EndWhile
Getting the final clustering result rs.
Return rs.
```

Complexity Analysis. Suppose N is the total number of data points in DS, M is the number of data points in the compressed dataset ODS. The time complexity of PCA-based dimension purification is ignored because it is a preprocessing. The time complexity of density-based data merge strategy is $O(N \log N)$. And, the time complexity of the dynamic interaction process is $O(T \times M \times S \times L)$, where T is the number of iterations, $M \ll N$, S is the average number of neighbors of each data point, L is the dimension after dimension purification. Therefore, the total time complexity of our SyncHigh algorithm is $O(N \times \log N) + O(T \times M \times S \times L)$.

Experimental evaluation

Evaluation setup

Comparison Algorithms. To effectively verify the performance of the SyncHigh algorithm, six representative clustering algorithms are selected as competitors. All comparison algorithms are listed in Table 1, where the Kmeans, EMA, Spectral Clustering and FAKM algorithms are considered to be the best clustering algorithms for normal datasets [16, 17], but the number of clusters needs to be manually pre-specified in advance; The DbScan algorithm is a density-based dynamic method, and the Sync algorithm is a native algorithm inspired the synchronization dynamics; DbScan and Sync algorithms have stronger adaptability, no need to pre-specify the number of clusters. For all comparison algorithms, recommended values of all parameters are used to get the best experimental results.

Evaluation metrics. To reasonably compare six algorithms with respect to effectiveness, three widely used metrics are selected to evaluate the quality of clustering. (1) The first metric is the clustering accuracy (ACC) [23], which represents the percentage of the obtained cluster label is the true label; (2) the second metric is Normalized Mutual Information (NMI) [24], which is defined as measuring the probability of having common information in two partitions of one dataset; (3) the third metric is the popular Adjusted Rand index (RI) [25], which both considers the probability of the obtained cluster label is the true label and the probability of the obtained cluster label is the wrong label. Three metrics scale between 0 and 1, and have the same characteristics: a larger ACC/NMI/RI means a better performance.

Experimental Platform. We use a normal PC server (ThinkPad L490) as the experimental platform. The server is equipped with one 8-core i7 CPU, 8 GB main memory and Windows 10 operating system. The SyncHigh algorithm is programmed with Python and run with PyCharm as IDE. For the other algorithms, the official python implementation is downloaded from the websites of the corresponding authors.

Table 1 Comparison Algorithms

Algorithm	Full name	Implement
Kmeans [18]	An efficient approximation to the K-means clustering for massive data	Python
DbScan [19]	DSets-DBSCAN: a parameter-free clustering algorithm	Python
Spectral Clustering [20]	Global discriminative-based nonnegative spectral clustering	Python
EMA [21]	An Expectation–Maximization algorithm for the Wishart mixture model: Application to movement clustering	Python
FAKM [22]	Fast Adaptive K-Means Subspace Clustering for High-Dimensional Data	Python
Sync [11]	Clustering by synchronization	Python
SyncHigh	Enhanced Synchronization-Inspired Clustering for High Dimensional Data	Python

Sensitivity analysis of two parameters

The first objective of experimental evaluation is to observe and validate the sensitivity of two parameters *sc* and d*c* in the dimension purification strategy and the density-based data merge strategy, respectively.

Sensitivity of parameter *sc*

Parameter *sc* is defined as a threshold to determine the percentage of principal components of high-dimensional attributes, then to decide the number of attributes that need to participate in the synchronization-inspired clustering algorithm. Usually, a larger *sc* value means a larger proportion of principal components, means more core attributes will be retained from the original high-dimensional attributes, and indicates more calculation time will be spent. By adjusting the parameter *sc*, SyncHigh can dynamically determine the number of core attributes participated in synchronization-inspired clustering, thereby balance and further optimize the accuracy and timeliness of the clustering algorithm itself.

To evaluate the sensitivity of parameter *sc*, we select DS2 (synthetic dataset) and ring (real-world dataset) as the experimental dataset, and observe the performance changes of the algorithm from the ACC, NMI and RI while gradually changing the value of *sc*. Figure 4a plots the sensitivity of the

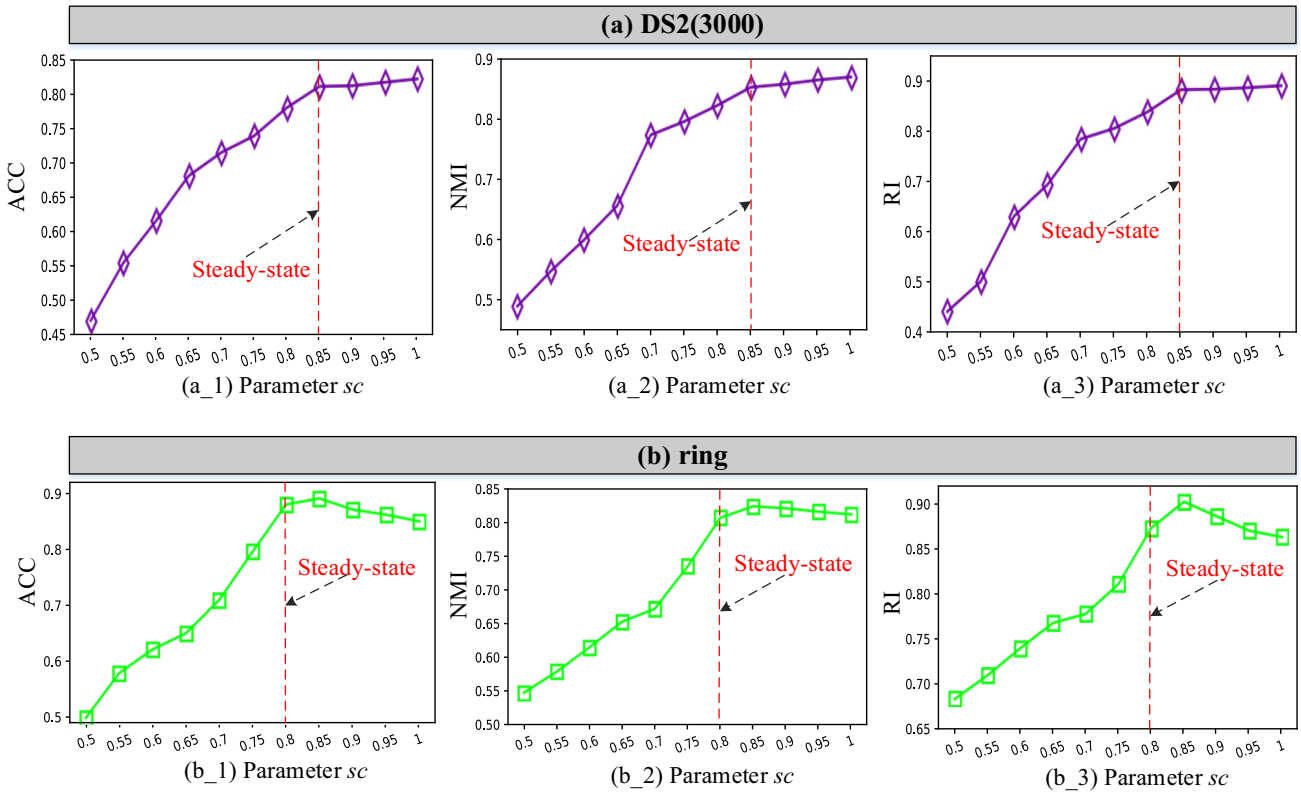

Fig. 4 Sensitivity of the parameter *sc* on two datasets

parameter *sc* on the DS2 dataset. From Fig. 4(a_1)–(a_3), it is not difficult to find that, as *sc* continues to expand (from 0.5 to 1), the number of core attributes participating in the clustering process is more and more, and the clustering quality (ACC, NMI, and RI) of SyncHigh algorithm is better and better (from 0.4 to 0.9). It is worth noting that, when *sc* value is greater than 0.85, the clustering quality (ACC, NMI, and RI) of SyncHigh algorithm begins to enter a stable state, as shown by the red dotted line in the figure. That is to say, after entering the steady state, ACC, NMI and RI will not change significantly with the increase of *sc* value. Figure 4b plots the sensitivity of the parameter *sc* on the ring dataset. From Fig. 4(b_1)–(b_3), we can easily find that, with the continuous increase of *sc* (from 0.5 to 1), the ACC, NMI, RI gradually increases (from 0.5 to 0.9), the clustering quality of SyncHigh algorithm continues to improve. Like DS2, when *sc* is greater than 0.8, the clustering quality of SyncHigh algorithm tends to smooth and starts the steady state.

In addition, we repeatedly carry out a large number of experiments on multiple synthetic and real-world datasets to more fully observe the sensitivity of parameter *sc*. By summarizing all experimental results, a simple conclusion is obtained. That is SyncHigh algorithm usually gets a good performance on the accuracy and effectiveness within the range $sc = [0.8–0.95]$. Moreover, compared with the native Sync algorithm, SyncHigh algorithm has less clustering time, and has similar and better clustering accuracy. Therefore, we set parameter $sc = 0.85$ as the default value in the following experiments.

Sensitivity of parameter dc

To avoid differences in the data distribution of different datasets, the average distance between any two data points is used as a basis, and parameter dc ranging from 0 to 1 is used as a threshold for judging the local density, so as to determine whether two data points belong to a same dense region. By combining d*c* and average distance, the robustness of predicting local density in different datasets are greatly improved. If the distance between two points is less than the product of d*c* and average distance, then these two data points are dense, and belong to a same density region. If all dense data points in the same region are merged into one big point, the number of data objects participating in the clustering process will be greatly reduced, and the clustering time will be greatly optimized. Usually, a larger d*c* value indicates that the data points are more likely to be dense, which means that more data points will be predicted as the same label without participating in the clustering process, and the required clustering time is less. By adjusting the parameter dc, SyncHigh

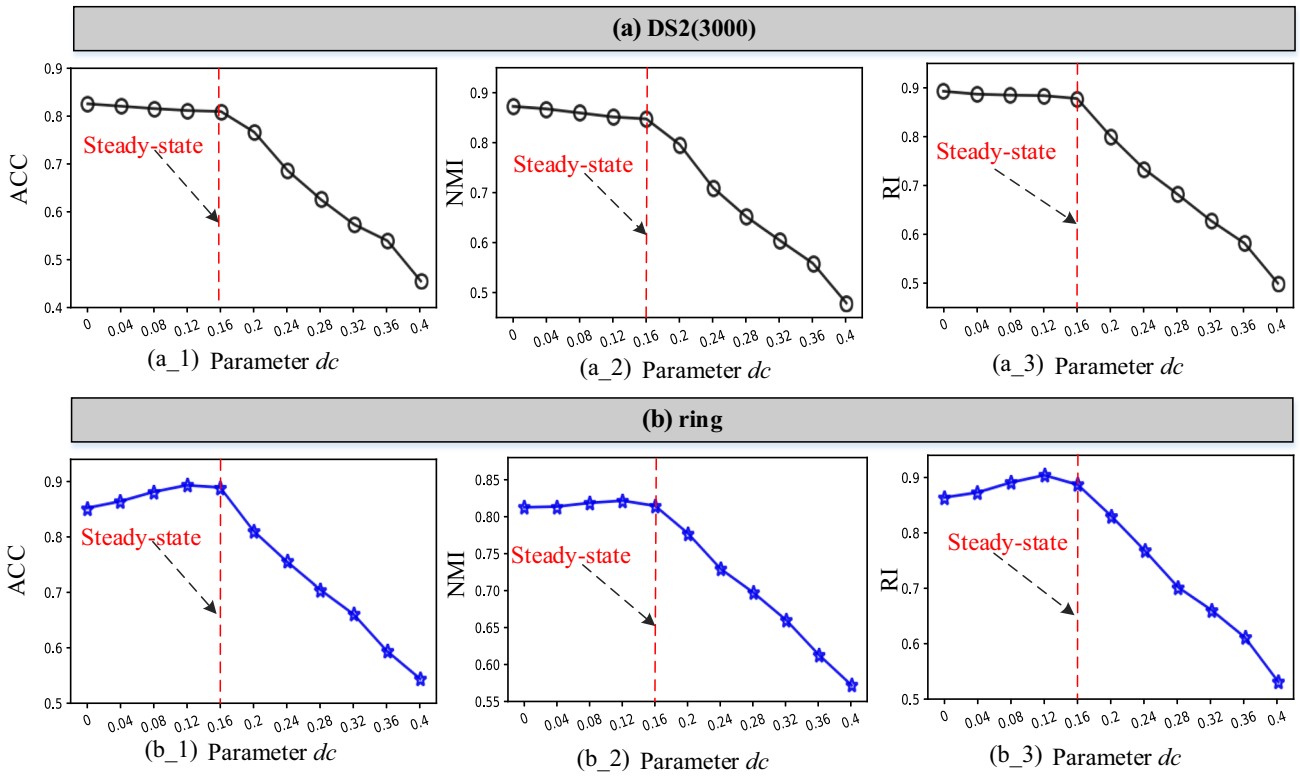

Fig. 5 Sensitivity of the parameter dc on two datasets

can better balance and further optimize the accuracy and time overhead.

For fairness, we also select DS2 and ring as the experimental datasets to observe the sensitivity of parameter dc from ACC, NMI and RI. Figure 5a plots the curve of the performance change of SyncHigh algorithm on the DS2 dataset. From Fig. 5a_1–a_3, it is easy to know that parameter dc is negatively correlated with ACC, NMI, ARI; these three metrics become smaller and smaller(from 0.9 to 0.4) with the increasing parameter dc (from 0 to 0.4). It is worth noting that, when dc value is smaller than 0.16, the clustering quality (ACC, NMI, and RI) of SyncHigh algorithm has been in a stable state, as shown by the red dotted line in the figure. That is to say, as the dc value increases gradually, the ACC, NMI, RI values do not fall fast. Like Fig. 5a, b plots the curve of the performance change on the ring dataset. Based on Fig. 5b_1–b_3, we can find similar situations. With the continuous increase of dc (from 0 to 0.4), the ACC, NMI, RI values gradually decrease (from 0.9 to 0.5), the clustering quality of SyncHigh algorithm continues to deteriorate. When dc is smaller than 0.16, the clustering quality of SyncHigh algorithm tends to smooth and keeps a steady state. In other words, when dc is smaller than 0.16, multiple dense points are merged reasonably and correctly to reduce the clustering time overhead, and optimize clustering quality. However, when dc is greater than 0.16, multiple dense points are mistakenly merged, which seriously affects the clustering quality of our SyncHigh algorithm.

Based on the extensive experiments on multiple synthetic and real-world datasets, a simple conclusion is obtained experimentally. That is, within the range dc = [0–0.15], SyncHigh algorithm can greatly reduce the time overhead while ensuring the quality of clustering. Therefore, we set parameter dc = 0.1 as the default value in the following experiments.

Evaluation results

Synthetic DataSets

The second objective of evaluation is to test the clustering performance of the SyncHigh algorithm on multiple synthetic datasets, from the two optimization strategies, there metrics (ACC, NMI, RI), and time efficiency.

DataSets generation To reasonably compare the effectiveness of seven clustering algorithms, we use Scikit-learn, a third-party machine learning python library, as a tool and use the *make_blobs* function to generate several synthetic high-dimensional datasets with Gaussian noise. The dataset generation model is defined as *genGaussSample*(S#, D#, C#, N#), where S# is the total number of data objects; D# is the

Table 2 Synthetic datasets

Name	Size (*S*#)	Dimension (*D*#)	Number of cluster (*C*#)	Gaussian noise (*N*#) (%)
DS1	1000	100	3	10
DS2	3000	70	3	7
DS3	5000	60	3	6
DS4	7000	70	3	5
DS5	10000	80	3	6
DS6	12,000	90	3	9
DS7	15,000	60	3	5

Table 3 Results of two optimization strategies on synthetic datasets

DataSets	(1) Dimension purification strategy			(2) Density-based data merge strategy		
	Original attribute size	Attribute size after purification	Purification rate (%)	Original dataset size	Dataset size after dense point merging	Dense point merging rate (%)
DS1	100	69	31	1000	715	29
DS2	70	51	27	3000	2154	28
DS3	60	46	23	5000	3627	27
DS4	70	55	21	7000	5341	24
DS5	80	64	20	10,000	7918	21
DS6	90	58	36	12,000	9427	21
DS7	60	36	40	15,000	11,156	26

data dimension; *C*# is the number of ground-truth cluster; *N*# is the percentage of Gaussian noise in dataset.

By adjusting four parameters, we generate seven synthetic datasets with ground-truth, as listed in Table 2. For reasonableness and effectiveness, seven datasets have different data scale, attribute dimension, and noise percentage. The purpose of this generation scheme is to make synthetic datasets closer to the real-world datasets.

Results of two optimization strategies To weaken the impact of high-dimensional attributes, based on the synchronization-inspired clustering algorithm, two optimization strategies are proposed to balance and optimize time efficiency and clustering accuracy. (1) The first strategy is dimension purification; its purpose is to find the core principal components from the high-dimensional attributes. By reducing the redundancy or noise attributes, SyncHigh algorithm can effectively improve time efficiency and optimize cluster accuracy. In Table 3, the optimization results of the first strategy on 7 artificial datasets are listed from the second column to the fourth column. Based on these statistics, it is not difficult to find that the dimensional purification strategy can effectively eliminate redundancy and noise attributes. The average rate of dimensional purification is greater than 25%, and the highest value is 40%(DS7). (2) The second strategy is density-based data merge strategy; its purpose is to reduce the amount of data participating in the synchronization-inspired clustering process by merging local dense points, thereby effectively improving the clustering time. The last three columns (five to seven columns) in Table 3 show the optimization results of the second strategy. It is easy to know that the average rate of merging local dense points is greater than 24%, and the highest value is 29%(DS1 with highest dimension).

Clustering influence of two optimization strategies To fairly verify the clustering influence of two optimization strategies, we integrate the two strategies with the native synchronization-inspired clustering algorithm (Sync) to form two new algorithms, simple called Sync + PCA and Sync + DDM. So far, native Sync, Sync + PCA, Sync + DDM and SyncHigh have formed four comparison algorithms. And, we will verify the clustering influence of the two optimization strategies on 7 synthetic datasets with respect to clustering accuracy and clustering time.

Table 4 lists the clustering accuracy influence of two optimization strategies on synthetic datasets with respect to three distinct metrics (ACC, NMI, and RI). In the table, the first column is 7 synthetic datasets, the 2–5 columns are the ACC performance of the 4 algorithms, the 6–9 columns are the NMI performance of the 4 algorithms, and the 10–13 columns are the RI performance of the 4 algorithms, respectively. From Table 4, we make the following observations. (1) Four algorithms achieve a good clustering performance on 7 syn-

Table 4 Clustering accuracy influence of two optimization strategies on synthetic datasets

DataSets	(1) ACC				(2) NMI				(3) RI			
	Sync	Sync + PCA	Sync + DDM	Sync High	Sync	Sync + PCA	Sync + DDM	Sync High	Sync	Sync + PCA	Sync + DDM	Sync High
DS1	**0.741**	**0.755**	**0.774**	**0.793**	**0.811**	**0.825**	**0.827**	**0.833**	**0.741**	**0.749**	**0.752**	**0.776**
DS2	0.824	0.832	0.809	0.813	0.872	0.889	0.869	0.883	0.891	0.879	0.894	0.882
DS3	0.857	0.862	0.831	0.848	0.893	0.862	0.911	0.884	0.891	0.895	0.898	0.904
DS4	0.811	0.834	0.781	0.808	0.933	0.944	0.911	0.921	0.891	0.889	0.869	0.873
DS5	0.861	0.853	0.878	0.854	0.862	0.869	0.878	0.874	0.881	0.872	0.878	0.867
DS6	**0.821**	**0.834**	**0.851**	**0.859**	**0.844**	**0.854**	**0.861**	**0.873**	**0.861**	**0.873**	**0.881**	**0.894**
DS7	0.871	0.862	0.883	0.878	0.891	0.898	0.867	0.874	0.852	0.841	0.842	0.844

thetic datasets, and the average values of ACC, NMI and RI both are greater than 0.8. (2) On the high-dimensional datasets DS1 and DS6 (two rows in bold), the clustering performance (ACC, NMI and RI) of Sync + PCA and Sync + DDM algorithms both are better than the native Sync algorithm, and slightly worse than the SyncHigh algorithm. This shows that two optimization strategies are highly compatible with the synchronization-inspired clustering algorithm, and can improve the clustering accuracy on high-dimensional datasets. (3)On other datasets, the Sync + PCA algorithm has the best performance (ACC, NMI and RI), the SyncHigh algorithm is closer to the Sync algorithm, and the Sync + DDM algorithm has the worst performance. (4) In summary, on 7 synthetic datasets, the PCA-based dimension purification strategy can effectively purify high-dimensional data attributes and improve clustering performance. However, the performance of the density-based data merge strategy is mediocre, and some non-dense nodes are misjudged. The reason is that we use the Gaussian model to generate the synthetic dataset, and its density distribution is relatively uniform.

Figure 6 further draws the clustering time influence of two optimization strategies on synthetic datasets. In the figure, the first bar is the clustering time of the sync algorithm (as a basis), the second bar is the time of the Sync + PCA algorithm, the third bar is the time of the Sync + DDM algorithm, the fourth bar is the time of the SyncHigh algorithm, and the black or red digit is the rate of time saving. From Fig. 6, we can get the following observations. (1) On the 7 datasets, two optimization strategies both can effectively reduce the clustering time of the synchronization-inspired algorithm. The average rate of time saving of the PCA-based dimension purification strategy is about 9%, the average rate of time saving of the density-based data merge strategy is about 19%. This shows that the performance of the second optimization strategy is slightly higher than the first optimization strategy. (2) With the increase of data dimensions, the performance of the two optimization strategies is getting better and better. For example, the time saving rate on DS1 and DS6 is significantly higher than that on other datasets (DS2, DS3, DS4, DS5 and DS7). (3) Focusing on Sync and SyncHigh algorithms, it is easy to find that SyncHigh has a better time efficiency than the Sync algorithm, the average rate of time saving is about 28%, and the maximum rate is 32% on DS6.

Clustering performance comparison In this section, we further compare the SyncHigh algorithm with multiple state-of-the-art algorithms from the clustering accuracy and clustering time. Figure 7 shows the clustering accuracy (ACC, NMI and RI) of multiple algorithms on 7 synthetic datasets. In the 7 datasets, DS1 and DS6 have the highest dimensions, respectively, 100 attributes and 90 attributes, as shown by

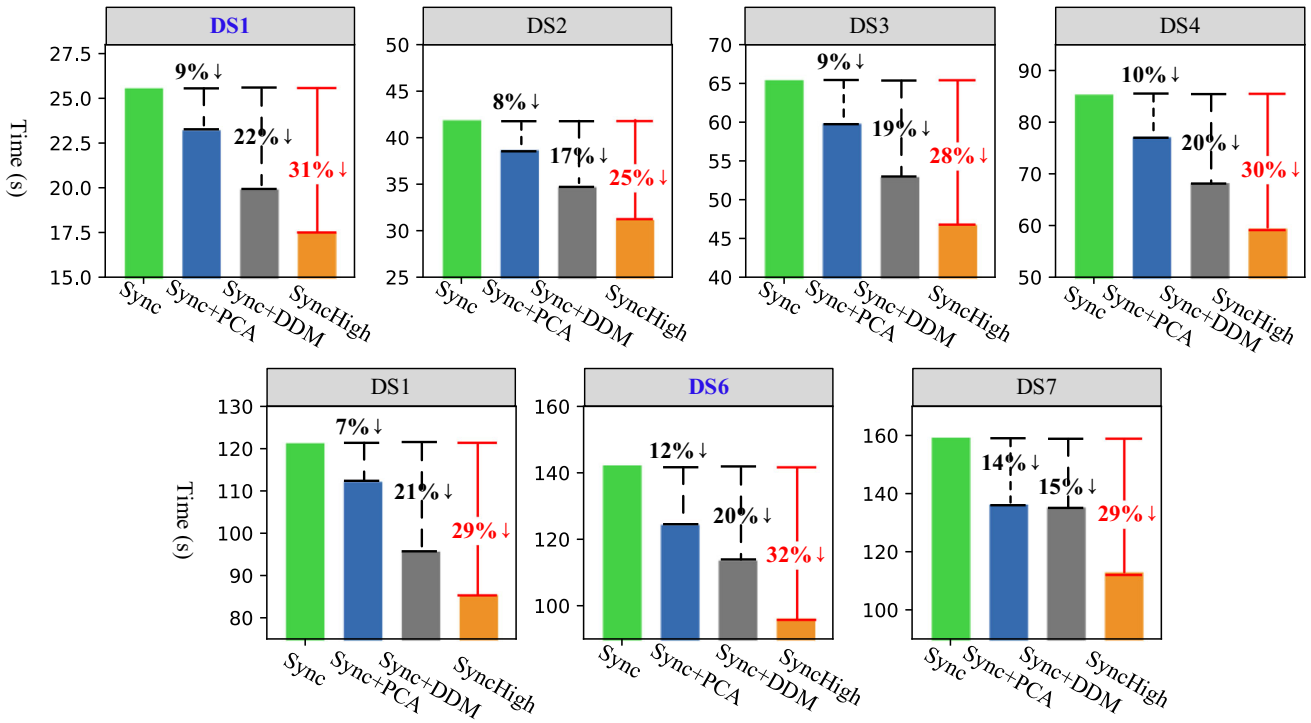

Fig. 6 Clustering time influence of two optimization strategies on synthetic datasets

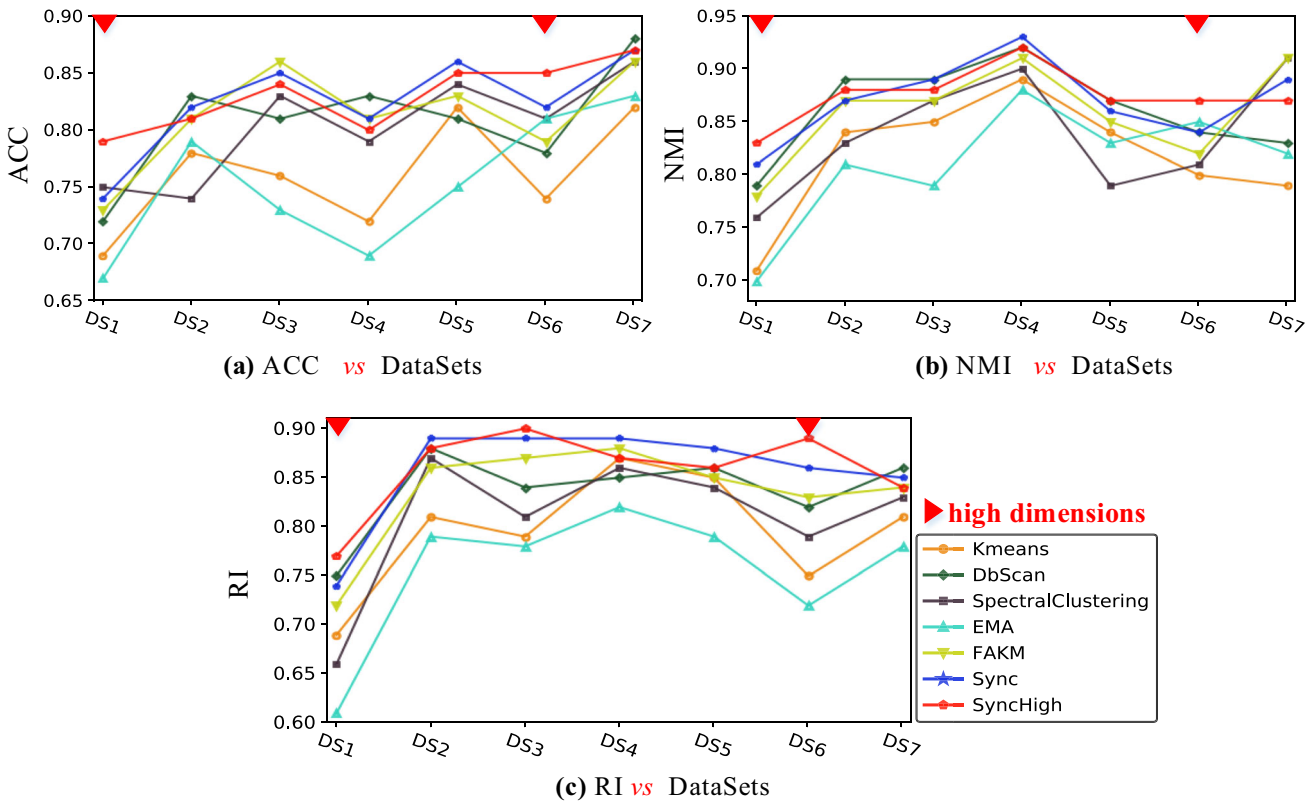

Fig. 7 Accuracy results on synthetic datasets

Table 5 Timeliness results on synthetic datasets

Datasets/time(s)	DS1	DS2	DS3	DS4	DS5	DS6	DS7
Kmeans	7.12	15.92	29.53	42.44	73.97	79.02	83.56
DbScan	34.78	66.14	84.02	98.47	150.87	176.48	198.47
Spectrual clustering	11.22	24.07	48.41	67.39	108.55	119.44	148.94
EMA	36.17	71.67	89.34	108.21	168.97	198.75	217.24
FAKM	23.73	42.78	68.85	79.49	133.94	156.51	165.17
Sync	25.59	41.92	65.45	85.45	121.45	142.45	159.45
SyncHigh	17.57(**31%↓**)	31.26(**25%↓**)	46.89(**28%↓**)	59.62(**30%↓**)	85.77(**29%↓**)	96.62(**32%↓**)	113.3(**29%↓**)

the red triangles in the figure. From Fig. 7, it is easy to get the following observations. (1) Considering ACC, NMI and RI together, seven algorithms both have good results, the average value of three metrics is greater than 0.8. Moreover, seven algorithms both show an unstable trend, and the metric lines float up or down. Comparing the seven algorithms, Sync, FAKM and SyncHigh are most stable on the 7 datasets, DbScan and SpectralClustering are next, Kmeans and EMA are the worst. (2) Focusing on the attribute dimension, the clustering accuracy (ACC, NMI, RI) of seven algorithms on the low-dimensional datasets (DS2–DS5, DS7) is generally better than that on the high-dimensional datasets (DS1 and DS6). More specifically, seven algorithms have the lowest ACC, NMI, RI on the DS1 and DS6. (3) Only consider Sync and SyncHigh algorithms, we can see that the clustering accuracy (ACC, NMI, RI) of the two algorithms is relatively stable and very close on the 7 datasets. More specifically, Sync algorithm has a slight advantage over SyncHigh algorithm on low-dimensional datasets (DS4 and DS7). However, on the high-dimensional datasets (DS1 and DS6), the ACC, NMI and RI values of SyncHigh are better than that of Sync algorithm. The main reason is that 7 datasets are synthetic datasets; their data distribution is relatively uniform and does not show strong density. This results in the mediocre performance of the density-based data merge strategy, the performance of the SyncHigh algorithm is affected, and the clustering accuracy of the SyncHigh algorithm is slightly worse than that of the native Sync algorithm.

Table 5 further lists the time overhead of seven algorithms on the 7 synthetic datasets. As shown in Table 5, the time overhead of the Kmeans algorithm is least, followed by the SyncHigh and SpectrualClustering algorithm, then Sync, FAKM and DbScan, and EMA algorithm is slowest. Moreover, it is easy to find that two dynamic clustering algorithms (DbScan and Sync) have more computation time than Kmeans and SpectrualClustering. Considering DbScan, FAKM and Sync algorithms, the time efficiency of the three high-dimensional data clustering algorithms is relatively close, and the time overhead of three algorithms is 3–5 times that of the Kmeans algorithm. Focusing on the SpectrualClustering and SyncHigh algorithms, when the dataset is small (DS1, DS2, and DS3), the time cost of the SpectrualClustering algorithm is less than that of the SyncHigh algorithm; However, as the dataset grows larger, the time cost of the SyncHigh algorithm with the help of the dimension purification strategy and density-based data merge strategy is gradually smaller than the SpectrualClustering algorithm. Finally, comparing the last two rows, the time overhead of the SyncHigh algorithm is much smaller than that of the native Sync algorithm on the 7 synthetic datasets, and the average rate of time saving is 28%, as shown by the bold digit in parentheses.

Table 6 Real-world datasets

Name	Size	Dimension	Number of cluster
Wine	178	13	3
Glass	214	9	6
Monks2	432	7	2
Emotions	593	72	6
Yeast	2417	103	14
Pageblocks	5473	10	5
Ring	7400	20	2
Penbased	10,092	16	10

Real-World DataSets

The third objective of evaluation is to test the clustering performance of the SyncHigh algorithm on several real-world datasets, from the two optimization strategies, there metrics (ACC, NMI, RI), and time efficiency.

DataSet selection To further evaluate the performance of seven clustering algorithms, eight typical real-world datasets with truth label are selected for this experiment, as listed in Table 6. All chosen real-world datasets are publicly available from the UCI machine learning data repository (http://archive.ics.uci.edu/ml/datasets/) and Knowledge extraction beads on evolutionary learning dataset collection (https://sci2s.ugr.es/keel/datasets.php). The eight real-world datasets belong to different types, where wine is a chemical dataset,

glass is a product dataset, monks2 is a problem dataset, emotions is a music dataset, yeast is a bio-information dataset, polblocks is a page dataset, penbased is a handwritten digits dataset. Moreover, these eight datasets have different sizes, dimensions and clusters where yeast and emotions are typical high-dimensional datasets.

Results of two optimization strategies Like synthetic datasets, we also validate the performance of dimension purification strategy and density-based data merge strategy on the 8 real-world datasets, as shown in Table 7. In the table, the first column is 8 datasets, the second to fourth columns are the optimization results of the first strategy, the five to seven columns are the optimization results of the second strategy. Through careful observation, it is easy to find the following phenomenon. (1) From 2 to 4 columns, the rate of dimension purification on 8 real-world datasets is very high (the average value exceeds 50% and the highest value has reached 71%), and significantly higher than the rate on the artificial datasets (Table 3). This phenomenon implies that there are more redundant and noise attributes in real-world datasets than artificial datasets. (2) From 5 to 7 columns, the rate of merging local dense points on the 8 real-world datasets is very close, the average value is 25% and the maximum value does not exceed 30%. Moreover, based on the comprehensive comparison of Table 3 and Table 7, the rate of merging local dense points on the real-world datasets is very close to the synthetic datasets. This phenomenon shows that whether in real-world datasets or synthetic datasets, there must be some local dense areas.

Clustering influence of two optimization strategies In this section, we also use Sync, Sync + PCA, Sync + DDM and SyncHigh as the comparison algorithms, to verify the clustering influence of the two optimization strategies on 8 real-world datasets from the aspects of clustering accuracy and clustering time.

Table 8 presents the clustering accuracy influence of two optimization strategies on real-world datasets. In the table, the first column is 8 real-world datasets, the 2–5 columns are the ACC value of the 4 algorithms, the 6–9 columns are the NMI value, and the 10–13 columns are the RI value, respectively. In addition, in the 8 real-world datasets, emotions and yeast have the highest dimensions, 72 and 103, respectively. From Table 8, some points can be easily discovered. (1) On the 8 real-world datasets, two optimization strategies both have a good clustering accuracy, the average value of ACC, NMI and RI both are greater than 0.7. And the clustering accuracy (ACC, NMI and RI) of Sync + PCA and Sync + DDM algorithms both are better than the native Sync algorithm, and slightly worse than the SyncHigh algorithm. (2) As the attribute dimension gradually increases, the performance of the two optimization strategies is getting better and better. The performance improvement of ACC, NMI and RI on two high-dimensional datasets (emotions and yeast, as shown by two rows in bold) both are better than those on other datasets. (3) With the help of two optimization strategies, the clustering accuracy (ACC, NMI, and RI) of SyncHigh algorithm is better than that of the native Sync algorithm. (4) Compared with synthetic datasets (Table 4), the two optimization strategies perform better on 8 real-world datasets, and have a greater improvement on ACC, NMI and RI. The main reason is that the data distribution of the real-world datasets is more dense and regional, leading to greater effectiveness of the two optimization strategies.

Figure 8 further plots the clustering time influence of two optimization strategies, and analyzes the time speeded-up performance on the 8 real-world datasets. In the figure, the first bar is the time overhead of the native Sync algorithm (as a basis), the second bar is the time overhead of the Sync + PCA algorithm, the third bar is the time overhead of the Sync + DDM algorithm, the fourth bar is the time overhead of the SyncHigh algorithm, and the black or red digit is the rate of time saving. Based on Fig. 8, it is easy to find the following things. (1) With the help of two optimization strategies, the clustering time of the synchronization-inspired algorithm has been greatly reduced. On the 8 real-world datasets, the average rate of time saving of the PCA-based dimension purification strategy is about 10%, the average rate of time saving of the density-based data merge strategy is about 24%. This shows that the time saving performance of the density-based data merge strategy is significantly higher than that of the PCA-based dimension purification strategy. (2) Focusing on the attribute dimension, the time saving rate of the two optimization strategies is proportional to the attribute dimension of the dataset. That is to say, the higher the attribute dimension of the dataset, the better the time saving rate of the two optimization strategies. For instance, the time saving rate on emotions and yeast is significantly higher than that on other datasets (wine, glass, monks2, and so on). (3) Considering Sync and SyncHigh algorithms, the time efficiency of SyncHigh algorithm is significantly higher than that of the native Sync algorithm, as shown by the red digits in the figure. The average rate of time saving of the SyncHigh algorithm is about 33%, and the maximum rate is 38% on emotions dataset. (4) Compared with synthetic datasets (Fig. 6), two optimization strategies show better performance on 8 real-world datasets. The main reason is that the real-world dataset has more noise and redundant properties, and presents the stronger and clearer region dense feature.

Clustering performance comparison To verify the overall performance of the SyncHigh algorithm, we further compare the SyncHigh algorithm with multiple state-of-the-art algorithms from the clustering accuracy and clustering time.

Table 7 Results of two optimization strategies on real-world datasets

DataSets	(1) Dimension purification strategy			(2) Density-based data merge strategy		
	Original attribute size	Attribute size after purification	Purification rate	Original dataset size	Dataset size after dense point merging	Dense point merging rate
Wine	13	6	54%↓	178	138	22%↓
Glass	9	4	56%↓	214	168	21%↓
Monks2	6	4	33%↓	432	337	22%↓
Emotions	72	21	71%↓	593	449	24%↓
Yeast	103	44	57%↓	2417	1774	27%↓
Pageblocks	10	3	70%↓	5472	4255	22%↓
Ring	20	15	25%↓	7400	5517	25%↓
Penbased	16	6	53%↓	10,992	8227	25%↓

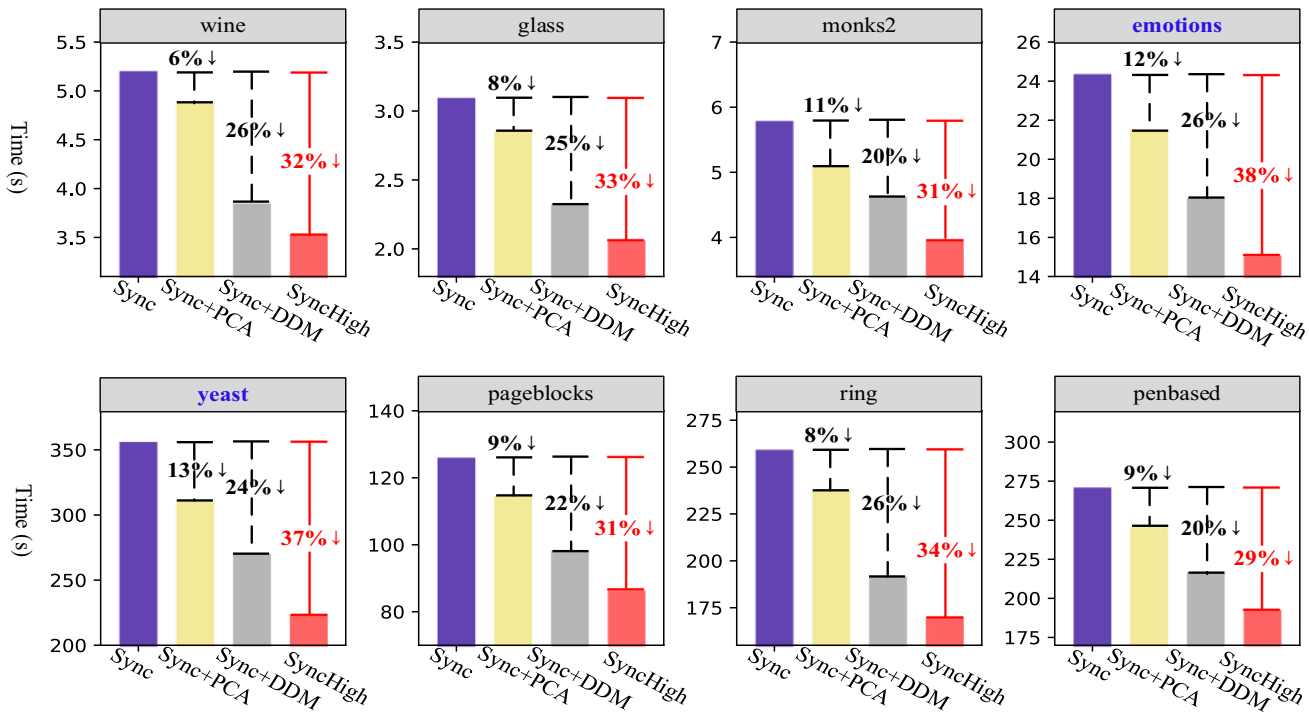

Fig. 8 Clustering time influence of two optimization strategies on real-world datasets

Figure 9 shows the clustering accuracy of seven algorithms on real-world datasets, where Fig. 9a plots the ACC results, Fig. 9b plots the NMI results, Fig. 9c plots the RI results, where a red triangle represents a high-dimensional dataset in the figure. According to Fig. 9, we can also get the following observations. (1) For three metrics (ACC, NMI, and RI), seven algorithms display different benefits on the real-world datasets. Overall, on the 8 real datasets, the ACC, NMI, and RI value of seven algorithms change rapidly, as shown in Fig. 9. Comparing the seven algorithms, Sync, SyncHigh, FAKM and DbScan have the best clustering accuracy; SpectralClustering is next; Kmeans and EMA are the worst. Moreover, Sync, FAKM and SyncHigh are more stable than other four algorithms on 8 real datasets. (2) When we consider the attribute dimension, the ACC, NMI, and RI value of six algorithms on the low-dimensional datasets (wine, monk2 and pageblocks) are better than those on the high-dimensional datasets (emotions and yeast). More specifically, seven algorithms have the lowest ACC, NMI, RI on the emotions and yeast datasets. And, the higher the dimension of a dataset, the worse the clustering stability of seven algorithms. (3) Only focusing on the Sync and SyncHigh algorithms, the ACC, NMI, RI values of two algorithms are relatively stable and very close on the 8 real datasets. SyncHigh algorithm has an advantage over Sync algorithm on both the low-dimensional datasets (wine, monk2 and pageblocks) and the

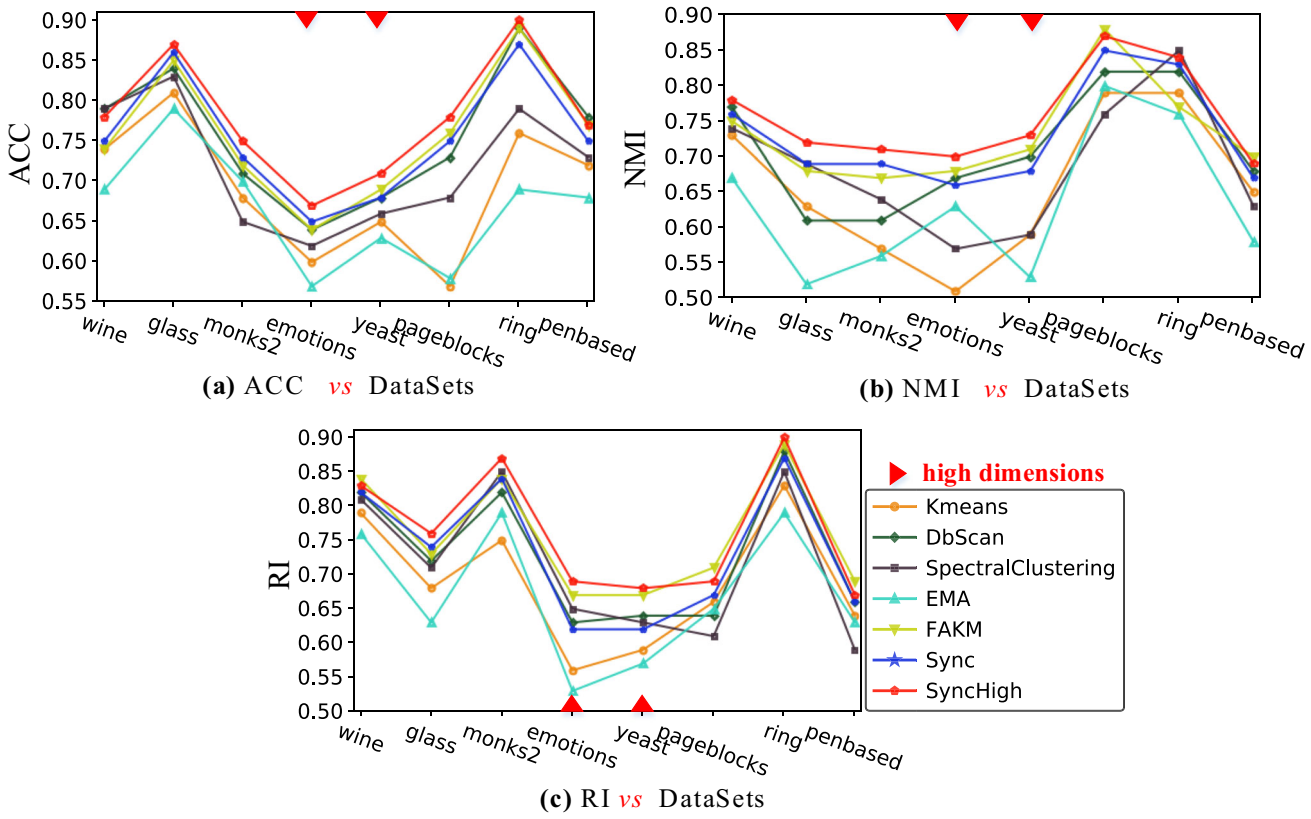

Fig. 9 Accuracy results on real-world Datasets

high-dimenisonal datasets (emotions and yeast). (4) When considering the synthetic datasets and real-world datasets together (Figs. 7 and 9), we can find that the clustering accuracy of seven algorithms on the synthetic datasets is better than that on the real-world datasets. The average of ACC, NMI and RI of seven algorithms on the real datasets is only 0.7, which is significantly lower than that on the synthetic datasets (0.8).

Table 9 further lists the time cost of seven algorithms on the 8 real-world datasets. From Table 9, some observations can be easily discovered. (1) For the time cost, the clustering time of Kmeans algorithm is minimum, SpectrualClustering and SyncHigh algorithm is next, then is Sync, FAKM and DbScan, and EMA algorithm is maximum. (2) DbScan, Sync, and SyncHigh as the dynamic clustering algorithms usually need more time overhead than Kmeans and SpectrualClustering algorithms. (3) When we only consider the SpectrualClustering and SyncHigh algorithms, we can get similar observation with the synthetic datasets (Table 5). When the scale of dataset is small (wine, glass, and monks2), the time cost of the SpectrualClustering algorithm is less than that of the SyncHigh algorithm. However, when the scale of dataset is greater and greater, with the help of the dimension purification strategy and density-based data merge strategy, the time cost of the SyncHigh algorithm is gradually smaller than that of the SpectrualClustering algorithm. (4) Focusing on the last two rows, it is not difficult to find that the time cost of the SyncHigh algorithm is much less than the time cost of the native Sync algorithm on 8 real-world datasets. The time saving rate on the two high-dimensional datasets (emotions and yeast) has reached 38% and 37%, and the smallest rate of time saving is 29% (penbased), as shown by the bold digit in parentheses. (5) Comparing Tables 5 and 9, we can find that the average rate of time saving of the SyncHigh algorithm on the real-world datasets is 32%, which is greater than that on the synthetic datasets (28%).

Conclusion

Timeliness is a great challenge for using the synchronization-inspired algorithm to cluster the high-dimensional dataset with high noise and high redundancy. In this paper, we proposed an enhanced synchronization-inspired clustering algorithm, namely SyncHigh, to quickly and accurately cluster high-dimensional datasets. To reduce the dimensions, we design a PCA-based dimension purification strategy to remove redundancy and noise of all attributes, and then find the principal components of the high-dimensional datasets . To speed up the clustering time, a density-based data

Table 8 Clustering accuracy influence of two optimization strategies on real-world datasets

DataSets	(1) ACC				(2) NMI				(3) RI			
	Sync	Sync + PCA	Sync + DDM	Sync High	Sync	Sync + PCA	Sync + DDM	Sync High	Sync	Sync + PCA	Sync + DDM	Sync High
Wine	0.751	0.772	0.761	0.784	0.761	0.772	0.769	0.787	0.821	0.833	0.829	0.836
Glass	0.862	0.871	0.865	0.876	0.693	0.714	0.708	0.726	0.742	0.759	0.749	0.764
Monks2	0.731	0.745	0.751	0.757	0.692	0.709	0.700	0.714	0.843	0.853	0.867	0.876
Emotions	**0.653**	**0.662**	**0.659**	**0.674**	**0.662**	**0.684**	**0.673**	**0.708**	**0.621**	**0.645**	**0.674**	**0.695**
Yeast	**0.682**	**0.702**	**0.694**	**0.718**	**0.681**	**0.708**	**0.719**	**0.737**	**0.621**	**0.653**	**0.638**	**0.684**
Pageblocks	0.751	0.759	0.771	0.785	0.853	0.862	0.869	0.873	0.674	0.684	0.679	0.698
Ring	0.873	0.886	0.891	0.908	0.832	0.839	0.842	0.849	0.871	0.897	0.882	0.906
Penbased	0.751	0.762	0.769	0.779	0.671	0.684	0.688	0.697	0.663	0.672	0.665	0.677

Table 9 Timeliness results on real-world datasets

DataSets/time(s)	Wine	Glass	Monks2	Emotions	Yeast	Pageblocks	Ring	Penbased
Kmeans	1.93	0.87	1.02	4.35	82.24	31.43	66.91	77.64
DbScan	5.29	2.65	5.14	32.13	413.61	184.63	344.39	398.83
SpectrualClustering	3.71	1.41	2.97	19.97	285.72	91.79	197.97	231.17
EMA	5.84	3.38	5.97	36.70	493.49	196.64	367.83	423.82
FAKM	4.01	2.32	4.37	20.19	328.47	119.76	232.72	256.35
Sync	5.21	3.10	5.8	24.4	356.09	126.10	259.34	271.18
SyncHigh	3.54 (32%↓)	2.07 (33%↓)	3.99 (31%↓)	15.13 (38%↓)	224.5 (37%↓)	86.61 (31%↓)	170.61 (34%↓)	193.06 (29%↓)

merge strategy is developed to reduce the number of data objects participating in the synchronization-inspired clustering process. Additionally, to avoid the mass difference of data points caused by the density-based data merge strategy, we further improved the Kuramoto Model to ensure clustering accuracy. The parameter sensitivity, clustering accuracy, computational complexity and the influence of two optimization strategies have also been analyzed. Numerical results on several synthetic datasets and real-world datasets have demonstrated the timeliness and effectiveness of the proposed algorithm.

Funding This study was funded by National Defense Basic Research Program of China (Grant No. JCKY2019403D006), National Natural Science Foundation of China (Grant No. 61702180), Natural Science Foundation of Hunan Province (Grant Nos.2018JJ2134, 2019JJ50167, 2020JJ5199), Hunan Provincial Young Talents Project (Grant No. 2018RS3095), Scientific Research Fund of Hunan Provincial Education Department (Grant Nos. 18C0296, 19B200), PhD research startup foundation of Hunan University of Science and Technology (Grant No. E51863).

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Saxena A, Prasad M, Gupta A et al (2017) A review of clustering techniques and developments. Neurocomputing 267:664–681
2. Zhao H, Xu L, Guo Z et al (2019) A new and fast waterflooding optimization workflow based on INSIM-derived injection efficiency with a field application. J Pet Sci Eng 179:1186–1200
3. Mittal M, Goyal LM, Hemanth DJ et al (2019) Clustering approaches for high-dimensional databases: a review. Wiley Interdiscip Rev Data Min Knowl Discov 9(3):e1300
4. Agrawal R, Gehrke J, Gunopulos D et al (2005) Automatic subspace clustering of high dimensional data. Data Min Knowl Discov 11(1):5–33
5. Chen X, Ye Y, Xu X et al (2012) A feature group weighting method for subspace clustering of high-dimensional data. Pattern Recognit 45(1):434–446
6. Yan F, Wang X-D, Zeng Z-Q et al (2020) Adaptive multi-view subspace clustering for high-dimensional data. Pattern Recognit Lett 130:299–305
7. Lakshmi BJ, Shashi M, Madhuri KJ et al (2020) A rough set based subspace clustering technique for high dimensional data. J King Saud Univ-Comput Inf Sci 32(3):329–334
8. Jin J, Wang W (2016) Influential features PCA for high dimensional clustering. Ann Stat 44(6):2323–2359
9. Yin W, Ma Z (2019) LE & LLE Regularized Nonnegative Tucker Decomposition for clustering of high dimensional datasets. Neurocomputing 364:77–94
10. Linderman G, Steinerberger S (2019) Clustering with t-SNE, provably. SIAM J Math Data Scie 1(2):313–332
11. Böhm C, Plant C, Shao J, et al. (2010) Clustering by synchronization. In: proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 583–592
12. Bae J, Helldin T, Riveiro M et al (2020) Interactive clustering: a comprehensive review. ACM Comput Surv (CSUR) 53(1):1–39
13. Chen L, Zhang J, Cai L et al (2017) Fast community detection based on distance dynamics. Tsinghua Sci Technol 22(6):564–585
14. Sheng GL, Su Y-L, Wang W-D (2019) A new fractal approach for describing induced-fracture porosity/permeability/compressibility in stimulated unconventional reservoirs. J Petrol Sci Eng 179:855–866
15. Honda K, Notsu A, Ichihashi H (2009) Fuzzy PCA-guided robust k-means clustering. IEEE Trans Fuzzy Syst 18(1):67–79
16. Pal R, Yadav S, Karnwal R (2020) EEWC: energy-efficient weighted clustering method based on genetic algorithm for HWSNs. Complex Intell Syst 6(1):1–10
17. Dey A, Son L, Pal A (2020) Fuzzy minimum spanning tree with interval type 2 fuzzy arc length: formulation and a new genetic algorithm. Soft Comput 24(6):3963–3974
18. Capó M, Pérez A, Lozano JA (2017) An efficient approximation to the K-means clustering for massive data. Knowl-Based Syst 117:56–69
19. Hou J, Gao H, Li X (2016) DSets-DBSCAN: a parameter-free clustering algorithm. IEEE Trans Image Process 25(7):3182–3193
20. Shang R, Zhang Z, Jiao L et al (2016) Global discriminative-based nonnegative spectral clustering. Pattern Recogn 55:172–182
21. Hidot S, Saint-Jean C (2010) An expectation-maximization algorithm for the Wishart mixture model: application to movement clustering. Pattern Recogn Lett 31(14):2318–2324
22. Wang X-D, Chen R-C, Yan F et al (2019) Fast adaptive K-means subspace clustering for high-dimensional data. IEEE Access 7:42639–42651
23. Amigó E, Gonzalo J, Artiles J et al (2009) A comparison of extrinsic clustering evaluation metrics based on formal constraints. Inf Retr 12(4):461–486
24. Domeniconi C, Gunopulos D, Ma S et al (2007) Locally adaptive metrics for clustering high dimensional data. Data Min Knowl Disc 14(1):63–97
25. Peng X (2019) New similarity measure and distance measure for Pythagorean fuzzy set. Complex Intell Syst 5(2):101–111