EasyChair Preprint

№ 1455

A combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters

Xinquan Chen

September 4, 2019

Title Page

A combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters

Xinquan Chen[1, 2]

[1]School of Computer & Information, Anhui Polytechnic University, Wuhu, 241000, China

[2]Key Laboratory of Intelligent Information Processing and Control, Chongqing Three Gorges University, Chongqing, 404100, China

chenxqscut@126.com

Author: Xinquan Chen

Corresponding Author: Xinquan Chen

*** Corresponding author**. Tel.: 0086-15123428097.

E-mail address: chenxqscut@126.com (X. Chen).

Post Address:

Xinquan Chen

School of Computer & Information, Anhui Polytechnic University, Wuhu, 241000, China

Conflicts of interest: None

A combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters

Abstract: ESynC algorithm is inspired by SynC algorithm and a linear version of Vicsek model. When facing complex data distributions, ESynC algorithm may regard a whole irregular cluster as some micro-clusters. In order to conquer this shortcoming, a Combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters (CESynC) is presented. CESynC algorithm uses ESynC algorithm to detect clusters or micro-clusters and a merging judgement process to merge those conjoint micro-clusters. For some data sets that ESynC algorithm and SynC algorithm cannot detect correct clusters, CESynC algorithm can obtain natural clusters. From some experiments of some artificial data sets, we observe that parameter δ in CESynC algorithm has better valid interval than ESynC algorithm and SynC algorithm in some cases. From the experiments of nine artificial data sets, we observe that the valid interval of parameter σ is affected by parameters δ and *MinPts*. From the experiments of eight UCI data sets, we observe that CESynC algorithm gets better (or the same) clustering results than (or as) that of ESynC algorithm. From many experiments, we observe that the clustering results of CESynC algorithm and ESynC algorithm are often better than that of SynC algorithm. So we can say CESynC algorithm can often obtain better clustering quality than ESynC algorithm and SynC algorithm in some kinds of data sets. Further comparison experiments with some classical clustering algorithms demonstrate the clustering effect of CESynC algorithm.

Keywords: Synchronization clustering; SynC algorithm; ESynC algorithm; micro-cluster; merging judgement

1. Introduction

Clustering is an important method in data mining and machine learning. Clustering tries to uncover the hidden structure by exploring the systematic partitioning of the unlabeled data set. According to a predefined similarity(or dissimilarity) measure, clustering algorithms can be used in pattern recognition, image segmentation, document organization, data compression, information retrieval, and bioinformatics.

The traditional clustering algorithms are mainly classified into partitioning-based methods (Bezdek, 1981; MacQueen, 1967), hierarchy-based methods (Guha et al., 1998; Karypis et al., 1999; Zhang et al., 1996), density-based methods (Ankerst et al., 1999; Ester et al., 1996; Roy et al., 2005), grid-based methods (Agrawal et al., 1998;

Wang et al., 1997), model-based methods (Theodoridis et al., 2006), and graph-based methods (Tan et al., 2005; Theodoridis et al., 2006; Zahn, C. T., 1971). Quantum clustering algorithms (Horn et al., 2002), spectral clustering algorithms (Luxburg, 2007; Schölkopf et al., 1998), affinity propagation clustering algorithms (Frey et al., 2007), synchronization clustering algorithms (Böhm et al., 2010; Chen, 2014, 2017, 2018; Hang et al., 2017; Huang et al., 2013; Shao et al., 2013a, 2013b, 2016, 2017a, 2017b) are some recent clustering methods.

Recent ten years, several famous clustering algorithms were published. Affinity propagation (named as AP) algorithm (Frey et al., 2007) is a new type of clustering algorithm based on probability graph models. After AP algorithm was published, clustering based on probability graph models grew a new research direction. As we know, the first synchronization clustering algorithm (named as SynC) was proposed by Böhm et al.(2010). After that, synchronization clustering attracts some researchers. Some synchronization clustering methods (Chen, 2014, 2017, 2018; Hang et al., 2017; Huang et al., 2013; Shao et al., 2013a, 2013b, 2016, 2017a, 2017b) were published from different views. Clustering by fast search and find of density peaks (named as DP) algorithm (Rodriguez et al., 2014) was developed based on the assumption that "cluster centers can be characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities". In DP algorithm, the number of clusters can be detected automatically, outliers can be identified easily, and even nonspherical clusters can be explored quickly. Very likely, DP algorithm can lead a new research branch in clustering field.

Synchronization means that some natural objects with similar rhythm will come into co-occurrence progressively. Some synchronization clustering models (Böhm et al., 2010; Chen, 2014; Chen, 2017) simulate the basic synchronization process by imposing the interactions on local near neighbor objects. Synchronization clustering is also a new kind of clustering approach. The original SynC algorithm declares that it can detect the intrinsic structure of the unlabelled data set and handle outliers without any distribution assumption (Böhm et al., 2010). Inspired by the idea of SynC algorithm and a linear version of Vicsek model, an effective synchronization clustering algorithm (named as ESynC) was presented (Chen, 2017). ESynC algorithm can often get better local synchronization effect than SynC algorithm and a similar synchronization clustering algorithm based on the original version of Vicsek model. But when facing complex distributions, ESynC algorithm may regard a whole irregular

cluster as some micro-clusters. In order to conquer this shortcoming, this paper researches a Combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters (named as CESynC). CESynC algorithm uses ESynC algorithm to detect clusters or micro-clusters and a merging judgement process to merge those conjoint micro-clusters.

The idea of "preclustering and merging" in CESynC algorithm is useful for some data sets with complex distribution. When ESynC algorithm and SynC algorithm cannot explore some irregular clusters with complex distributions, it is necessary that adding a merging judgement process to merge those conjoint micro-clusters that are detected by ESynC algorithm.

The remainder of this paper is organized as follows. Section 2 lists some related work. Section 3 gives some basic knowledge. Section 4 introduces CESynC algorithm. Section 5 validates CESynC algorithm by some simulated experiments. Conclusions and future work are presented in Section 6.

2. Related work

This paper is inspired by several papers (Böhm et al., 2010; Chen, 2014, 2017) and the idea of "preclustering and merging".

2.1 The origin and advance of synchronization clustering

In 2010, a novel synchronization clustering (SynC) algorithm was presented by Böhm et al. (2010). SynC algorithm attempts to explore the intrinsic distribution structure of the data set and handle isolates by dynamic synchronization. In order to implement automatic clustering, those natural clusters can be detected by using the minimum description length (named as MDL) principle (GrÄunwald, 2005).

In 2013, Huang et al. (2013) proposed a synchronization-based hierarchical clustering method basing on the work of Böhm et al. (2010). In order to find the intrinsic patterns of a complex graph, a novel and robust graph clustering algorithm, RSGC (Shao et al., 2013a), was proposed by regarding the graph clustering as a dynamic process towards synchronization. In order to explore meaningful levels of the hierarchical cluster structure, a novel dynamic hierarchical clustering algorithm, hSync (Shao et al., 2013b), was presented based on synchronization and the MDL principle.

In 2014, inspired by the work of Böhm et al. (2010) and Vicsek model (Vicsek et al., 1995; Jadbabaie et al., 2003; Wang et al., 2009), Chen (2014) presented a shrinking synchronization clustering algorithm by using a linear weighted Vicsek model.

In 2016, an effective scalable synchronization clustering algorithm for large datasets named as CIPA (Shao, et al., 2016) was presented. CIPA algorithm can handle very large datasets by iteratively partitioning them into thousands of subsets and clustering each subset separately.

In 2017, inspired by the work of Böhm et al. (2010) and Vicsek model, Chen (2017) proposed ESynC algorithm based on a linear version of Vicsek model. Simulated experiments validate that the linear version of Vicsek model is an effective synchronization model for clustering. Based on the metaphor of gravitational kinematics and central force optimization method, Hang et al. (2017) presented a local synchronization clustering algorithm, which can find clusters of those data sets with arbitrary size, shape, and density, and determine the number of clusters automatically. Qin et al. (2017) investigated group synchronization problem for multiple interacting clusters of nonidentical systems that are linearly or nonlinearly coupled. Based on Lyapunov method, they provided the sufficient conditions guaranteeing the group synchronization behavior and performed rigorous group synchronization analysis. To discover the co-cluster structure of gene expression data, a new synchronization-based co-clustering algorithm (named as CoSync) was presented (Shao et al., 2017a). CoSync algorithm can detect high-quality biologically relevant subgroups embedding in a given gene expression data matrix. In order to explore subspace clusters of some high-dimensional sparse data sets, a novel effective and efficient subspace clustering algorithm, ORSC (Shao et al., 2017b), was proposed. ORSC algorithm can detect correlation clusters in arbitrarily oriented subspaces and do not need to specify the subspace dimensionality or other difficult parameters.

In 2018, Chen (2018) presents three fast synchronization clustering (named as FSynC) algorithms basing on the work of Böhm et al. (2010), R-tree structure, and the grid-based index method. FSynC algorithms have three improved versions of SynC algorithm by storing all data points in a R-tree structure or by combining multidimensional grid partitioning method and Red-Black tree structure to construct the near neighbor point sets of all points in each synchronization evolution (Chen, 2018).

2.2 The idea of “merging judgement”

The idea of “merging judgement” is not presented originally. It is used in multiple fields of data mining. For example, a famous hierarchical clustering method, AGNES (Agglomerative Nesting) (Kaufman et al., 1990), uses the single link method and the

dissimilarity matrix to merge micro-clusters that have the least dissimilarity. The dissimilarity of merging two micro-clusters can use single link (smallest distance between an element in one micro-cluster and an element in the other), complete link (largest distance between an element in one micro-cluster and an element in the other), average (average distance between an element in one micro-cluster and an element in the other), or centroid (distance between the medoids of two micro-clusters).

2.3 The new advance of other clustering methods

Clustering is a pretreatment technique and a base in data mining field. New clustering methods are developed from different views. Recently, He et al. (2017) presented a kernel conditional clustering (named as KCC) algorithm by using kernel based conditional dependence measure as its objective function. KCC algorithm has no assumption about the cluster structure, the covariates, or the distribution of the data set. And it can both discover non-linearly separable clusters and detect the true cluster structures more accurately than some other alternative clustering methods.

In order to obtain a more balanced partitioning and avoid the appearance of singleton clusters, Chehreghani (2017) used the sum of the squared size of the clusters as an additive regularization term for the min cut cost function and proposed an efficient local search algorithm to optimize the objective function.

In order to avoid repeated computation of polynomial approximation when reconstructing the Laplacian matrix, a fast compressive spectral clustering (named as FCSC) algorithm (Li et al., 2017) was presented. FCSC algorithm can reduce the computation time significantly while preserving high clustering accuracy.

In joint action grouping and modeling, a hierarchical clustering multi-task learning (named as HC-MTL) method was presented by formulating the objective function into the group-wise least square loss, which was regularized by the trace norm and group sparsity terms for joint multiple action learning (Liu et al., 2017). HC-MTL method can aid in discovering both shared-action relatedness and action-specific feature subspaces.

3. Some basic knowledge

Suppose there is a data set $S = \{X_1, X_2, \ldots, X_n\}$ in a d-dimensional Euclidean space. Naturally, we use Euclidean metric as our dissimilarity measure, $dis(\cdot, \cdot)$. In order to describe our algorithm clearly, some concepts are presented first.

Definition 1 The δ near neighbor point set $\delta(P)$ of point P is defined as:

$$\delta(P) = \{X \mid 0 < dis(X, P) \le \delta, X \ne P, X \in S\}, \quad (1)$$

where $dis(X, P)$ is the dissimilarity measure between point X and point P in the data set S. Parameter δ is a predefined range threshold.

Definition 2 (Chen, 2017). The linear version of Vicsek model for clustering used in ESynC algorithm is defined as:

Point $X = (x_1, x_2, \cdots, x_d)$ is a vector in d-dimensional Euclidean space. If each point X is regarded as an agent based on a linear version of Vicsek model, with an interaction in the δ near neighbor point set $\delta(X)$, then the dynamics of point X over time according to Jadbabaie et al. (2003) and Wang et al. (2009) is described by:

$$X(t+1) = \frac{1}{\left(1 + \left|\delta(X(t))\right|\right)}\left(X(t) + \sum_{Y \in \delta(X(t))} Y\right), \quad (2)$$

where $X(t = 0) = (x_1(0), x_2(0), \cdots, x_d(0))$ represents the original location of point X, and $X(t+1)$ describes the renewal location of point X at the t-step evolution.

Definition 3 (Chen, 2017) The t-step average length of edges, $AveLen(t)$, in a t-step δ near neighbor undirected graph $G_\delta(t)$ is defined as:

$$AveLen(t) = \frac{1}{|E(t)|} \sum_{e \in E(t)} |e|, \quad (3)$$

where $E(t)$ is the t-step edge set of $G_\delta(t)$, and $|e|$ is the length (or weight) of edge e. The average length of edges in $G_\delta(t)$ decreases to its limit 0, that is $AveLen(t) \to 0$, as more δ near neighbor points synchronize together with time evolution. In ESynC algorithm, $AveLen(t)$ can be used to characterize the degree of local synchronization.

Property 1 The data set $S = \{X_1, X_2, \ldots, X_n\}$ using ESynC algorithm for clustering will obtain an effective result of local synchronization with some obvious clusters or isolates, if parameter δ satisfies:

$$\max\{longestDistance(cluster_k) \mid k = 1, 2, \ldots, K_{clu}\} \le \delta < \min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}, \quad (4)$$

where $longestDistance\ (cluster_k) = \max\{\ dis(P, Q) \mid P \in cluster_k, Q \in cluster_k, P \ne Q\}$ is the longest edge in the complete graph of the k-th cluster, $dis_{\min}(cluster_i, cluster_j) = \min\{dis(P, Q) \mid P \in cluster_i, Q \in cluster_j, P \ne Q\}$ is the weight of the minimum edge connecting the i-th cluster and the j-th cluster, and K_{clu} is the number of clusters in the final synchronization step.

Proof: Suppose the data set $S = \{X_1, X_2, \ldots, X_n\}$ has K_{clu} obvious clusters. If parameter δ is larger than or equal to $\max\{longestDistance\ (cluster_k) \mid k = 1, 2, \ldots, K_{clu}\}$,

then data points in the same cluster will synchronize to a steady location. If parameter δ is less than $\min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}$, then data points in different clusters will not interactive and can not synchronize.

Property 2 The data set $S = \{X_1, X_2, \ldots, X_n\}$ uses ESynC algorithm for clustering. If $\max\{longestDistance(cluster_k) \mid k = 1, 2, \ldots, K_{clu}\} > \min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}$, then the data set S might obtain an effective result of local synchronization with some obvious clusters or isolates. That is: if we set $\delta > \min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}$, some obvious clusters may be detected.

Property 2 is validated by some simulations.

Definition 4 A micro-cluster is a cluster or a part of a cluster that can be detected by using ESynC algorithm with a small value of parameter δ. Usually, the number of points in a micro-cluster should be larger than or equal to the threshold parameter *MinPts*.

Note: This threshold *MinPts* is the similar to the parameter *MinPts* used in DBSCAN algorithm. Parameter *MinPts* is often set from 1 to 4 for many data sets.

Property 3 For some data sets, some clusters have strange spatial distributions and $\min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}$ is small. Parameter δ of ESynC algorithm has no valid interval. But some clusters might be detected by merging some conjoint micro-clusters which can be detected by using ESynC algorithm with a small value of parameter δ.

Property 3 is also validated by some simulations. For example, seven 2-D data sets (data0 – data6) are used to validate this conclusion in subsection 5.2.

Definition 5 Suppose the data set $S = \{X_1, X_2, \ldots, X_n\}$ using ESynC algorithm for clustering obtain K_{clu} micro-clusters, $\boldsymbol{MCS} = \{MCS_i \mid i = 1, 2, \ldots, K_{clu}\}$. In K_{clu} micro-clusters, the K_{clu} steady locations (or K_{clu} mean locations in some cases) are often selected as K_{clu} micro-cores, $\boldsymbol{MC} = \{MC_i \mid i = 1, 2, \ldots, K_{clu}\}$. Then the Minimum Spanning Tree (MST) of $\boldsymbol{MC}$ can be constructed by Prim algorithm with $O(K_{clu}^2)$ time complexity.

Furthermore, based on the MST of $\boldsymbol{MC}$, an Minimum Connecting Bracket Tree (MCBT) of $\boldsymbol{MCS}$ can be constructed by replaced the weight of every edge in the MST of $\boldsymbol{MC}$, $dis(MC_i, MC_j)$, by a new weight, $dis_{\min}(MCS_i, MCS_j)$. The new weight computing equation is:

$$dis_{\min}(MCS_i, MCS_j) = \min\{dis(P, Q) \mid P \in MCS_i, Q \in MCS_j, i \neq j\}, \quad (5)$$

4. A combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters

Although we use the Euclidean metric as our dissimilarity measure in this paper, this algorithm is by no means restricted to this metric and this kind of data space. If we can construct a proper dissimilarity measure in a hybrid-attribute space, this algorithm can still be used.

4.1 The description of ESynC algorithm

ESynC algorithm is developed by Chen (Chen, 2017). In order to make a difference between ESynC algorithm and this new method, we introduce it simply below.

Algorithm name: an Effective Synchronization Clustering algorithm (ESynC).

Input: data set $S = \{X_1, X_2, \ldots, X_n\}$, dissimilarity measure $dis(\cdot, \cdot)$, and parameter δ.

Output: The final convergent result $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$ of the original data set S.

The main procedure of ESynC algorithm is described by **Table 1**.

Table 1 The main procedure of ESynC algorithm.

Step1. Initialization:

1: IterativeStep t is set as zero firstly, that is: $t \leftarrow 0$;
2: **for** ($i = 1$; $i \le n$; i++)
3: $X_i(t) \leftarrow X_i$;

Step2. Execute the iterative synchronization process of the dynamical clustering:

4: **while** ((the dynamical clustering does not satisfy its convergent condition) **and** ($t < 20$))
5: {
6: **for** ($i = 1$; $i \le n$; i++)
7: {
8: Construct the δ near neighbor point set $\delta(X_i(t))$ for each point $X_i(t)$ ($i = 1, 2, \ldots, n$) using Eq.(1) of Definition 1;
9: Compute the renewal value, $X_i(t+1)$, of $X_i(t)$ using Eq.(2) of Definition 2;
10: }
11: Compute the t-step average length of edges of all points, $AveLen(t)$, using Eq.(3) of Definition 3;
12: IterativeStep t is increased by 1, that is: t++;
13: **if** ($AveLen(t) \rightarrow 0$)
14: The dynamical clustering reaches its convergent result, and then exit from the while repetition;
15: }

Step3. Finally we get a convergent result $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$, where T is the times of the while repetition in Step2. The final convergent set $S(T)$ reflects the natural clusters or isolates of the data set S.

In the convergent result $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$, suppose there are K ($K \le n$) steady locations. If a steady location contains some points in $S(T)$ such that the

number of points is larger than or equal to the threshold parameter *MinPts*, they can be regarded as a micro-cluster or a cluster. If a steady location contains only one point or several points of $S(T)$ such that the number of points is less than *MinPts*, then it is an isolate or they are isolates. Suppose there are K_{clu} micro-clusters and K_{iso} isolates in the K steady locations. Here, $K = K_{clu} + K_{iso}$ is satisfied.

4.2 The description of CESynC algorithm

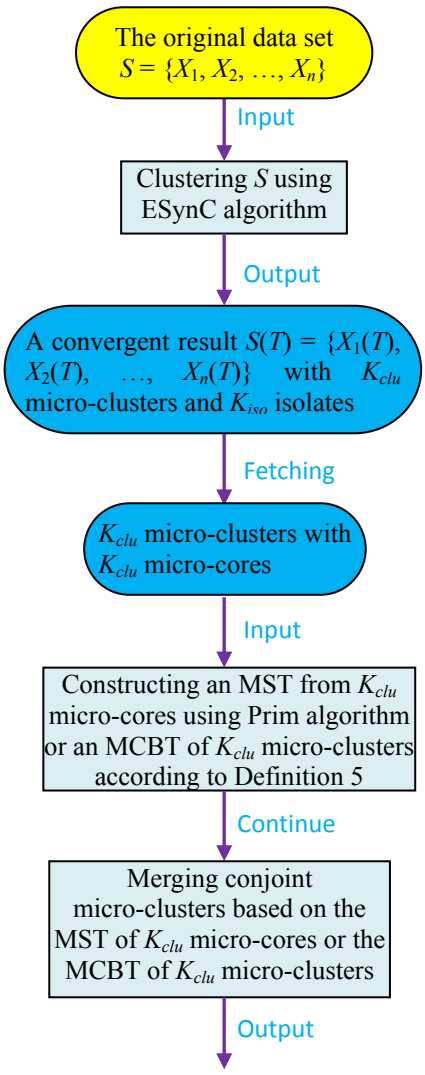
CESynC algorithm has a different process with SynC algorithm (Böhm et al., 2010) and ESynC algorithm (Chen, 2017). The basic flow of CESynC algorithm is presented by **Fig**. **1**.

Algorithm name: a Combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters (CESynC).

Input: data set $S = \{X_1, X_2, \ldots, X_n\}$, dissimilarity measure $dis(\cdot, \cdot)$, and range parameter δ, and density threshold parameter *MinPts* (the minimum number of points in a micro-cluster or a cluster).

Output: The final clustering result $\boldsymbol{FCS} = \{FCS_1, FCS_2, \ldots, FCS_k\}$ of the original data set S.

The main procedure of CESynC algorithm is described by **Table 2**.



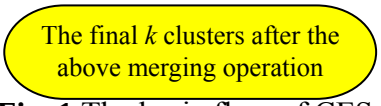


Fig. 1 The basic flow of CESynC algorithm

Table 2 The main procedure of CESynC algorithm.

Step1. Call ESynC algorithm for the original data set $S = \{X_1, X_2, \ldots, X_n\}$. The function call formal of this step is:

$S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\} \leftarrow$ ESynC(DataSet S, float δ);

/* $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$ record the final steady locations of some micro-clusters or isolates of the data set S. */

Step2. Fetch K_{clu} micro-clusters and K_{iso} isolates from $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$. Here, we mainly consider K_{clu} micro-clusters. The function call formal of this step is:

$\boldsymbol{MCS} = \{MCS_1, MCS_2, \ldots, MCS_{Kclu}\} \leftarrow$ Fetch($S(T)$, int *MinPts*);

/* $\boldsymbol{MCS} = \{MCS_1, MCS_2, \ldots, MCS_{Kclu}\}$ are K_{clu} micro-clusters of the data set S. */

Step3. Construct an MST from K_{clu} micro-cores (K_{clu} steady locations or K_{clu} mean locations of K_{clu} micro-clusters are selected as K_{clu} micro-cores), $\boldsymbol{MC} = \{MC_1, MC_2, \ldots, MC_{Kclu}\}$, using Prim algorithm. The function call formal of this step is:

$mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\} \leftarrow$ Prim($\boldsymbol{MC}$); /* Suppose $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\}$ is sorted by an increased sequence. */

Or further construct an MCBT from K_{clu} micro-clusters based on the $mst(\boldsymbol{MC})$ according to Definition 5. The function call formal of this step is:

$mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\} \leftarrow$ ReplaceWeights($\boldsymbol{MCS}$, $mst(\boldsymbol{MC})$); /* Suppose $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$ is sorted by an increased sequence such that $ce_1 \le ce_2 \ldots \le ce_{Kclu-1}$. */

Step4. Merge K_{clu} micro-clusters, $\boldsymbol{MCS} = \{MCS_1, MCS_2, \ldots, MCS_{Kclu}\}$, if they are satisfied with the merging condition, based on the $mst(\boldsymbol{MC})$ or $mcbt(\boldsymbol{MCS})$. The function call formal of this step is:

$\boldsymbol{FCS} = \{FCS_1, FCS_2, \ldots, FCS_k\} \leftarrow$ Merge($\boldsymbol{MCS}$, $mst(\boldsymbol{MC})$ or $mcbt(\boldsymbol{MCS})$);

/* $\boldsymbol{FCS} = \{FCS_1, FCS_2, \ldots, FCS_k\}$ are k clusters of the data set S after the merging process. */

Step5. Finally we get a set with k clusters, $\boldsymbol{FCS} = \{FCS_1, FCS_2, \ldots, FCS_k\}$. The final set $\boldsymbol{FCS}$ may reflect the natural clusters of the data set S.

Note: In Step4, the merging step in CESynC algorithm has two implement merging strategies and merging judgement methods that are presented in section 4.3.

4.3 The merging strategies and judgement methods of merging micro-clusters in CESynC algorithm

Here we present two concrete merging strategies and two judge methods of merging micro-clusters for CESynC algorithm.

4.3.1 ***Strategy 1***: The MST-based inflexion-point strategy

This strategy is designed by searching the inflexion-point from the $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\}$. That is:

(Sa). First fetch K_{clu} micro-cores from K_{clu} micro-clusters (for example, select K_{clu} steady locations), $\boldsymbol{MC} = \{MC_1, MC_2, \ldots, MC_{Kclu}\}$;

/* $\boldsymbol{MC} = \{MC_1, MC_2, \ldots, MC_{Kclu}\}$ are K_{clu} micro-cores of K_{clu} micro-clusters. */

(Sb). In **Step3** of CESynC algorithm, it is implemented concretely by:

$mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\} \leftarrow \text{Prim}(\boldsymbol{MC})$; /* Suppose $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$ is sorted by an increased sequence such that $e_1 \le e_2 \ldots \le e_{Kclu\text{-}1}$. */

(Sc). Select some minimum edges from $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$. These selected edges are less than the other edges. Suppose there are *NumMinMst* minimum edges. Usually, (*Kclu* - *NumMinMst*) is equal to the right number of clusters in those data sets with obvious clusters. There are four cases in $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$.

Case 1: $e_1 \le \ldots \le e_{j\text{-}1} \le e_j \le e_{j+1} \le e_{j+2} \le \ldots \le e_{Kclu\text{-}1}$.

In this case, we either consider the merging operation of $\{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$, or do not consider any merging operation.

/* If the $(K_{clu} - 1)$ edges of $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$ are increased steadily and have no change suddenly, then the K_{clu} micro-clusters are either K_{clu} disjoint clusters or one cluster that can be constructed by merging the K_{clu} micro-clusters. */

Case 2: $e_1 \le \ldots \le e_{j\text{-}1} \le e_j \ll e_{j+1} \le e_{j+2} \le \ldots \le e_{Kclu\text{-}1}$, such that $j = \underset{i=1,2,\ldots,K_{clu}-2}{\arg\max} (e_{i+1} - e_i)$, where e_j and e_{j+1} are two near neighbor edges with maximal diversification in the $mst(\boldsymbol{MC})$.

In this case, we only consider the merging operations of $\{e_1, e_2, \ldots, e_j\}$.

/* In the $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$, if the first j edges $\{e_1, e_2, \ldots, e_j\}$ are increased steadily and there is a sudden change between e_j and e_{j+1}, then $\{e_1, e_2, \ldots, e_j\}$ should be used to consider the merging operations of the corresponding $(j+1)$ micro-clusters */

Case 3: $e_1 \le \ldots \le e_{j\text{-}1} \le e_j \ll e_{j+1} \le e_{j+2} \le \ldots \le e_{j+l\text{-}1} \le e_{j+l} \ll e_{j+l+1} \le e_{j+l+2} \le \ldots \le e_{Kclu\text{-}1}$.

In this case, we either consider the merging operation of $\{e_1, e_2, \ldots, e_j\}$, or consider the merging operation of $\{e_1, e_2, \ldots, e_j, e_{j+1}, \ldots, e_{j+l}\}$.

Case 4: Like Case 3, there are three or three above sudden change in $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu\text{-}1}\}$.

In this case, we either consider the merging operation of $\{e_1, e_2, \ldots, e_j\}$, or consider the merging operation of $\{e_1, e_2, \ldots, e_j, e_{j+1}, \ldots, e_{j+l}\}$, or …..

(Sd). Merge two near neighbor micro-clusters connected by any edge in the *NumMinMst* minimum edges of $mst(\boldsymbol{MC})$ if they can be merged. That is:

for ($i = 1$; $i \le NumMinMst$; i++)
{
  if (two near neighbor micro-clusters connected by e_i can be merged)
    The two near neighbor micro-clusters should be merged.
}

4.3.2 ***Strategy 2***:The MCBT-based inflexion-point strategy

This strategy is also a concrete algorithm based on the Minimum Connecting Bracket Tree of $\boldsymbol{MCS} = \{MCS_1, MCS_2, \ldots, MCS_{Kclu}\}$. That is:

(Sa). First fetch K_{clu} micro-cores of K_{clu} micro-clusters (for example, select K_{clu} steady locations), $\boldsymbol{MC} = \{MC_1, MC_2, \ldots, MC_{Kclu}\}$;

(Sb). In **Step3** of CESynC algorithm, it is implemented concretely as follows.

First constructing an MST from K_{clu} micro-cores by using Prim algorithm, then constructing an MCBT of K_{clu} micro-clusters based on $mst(\boldsymbol{MC})$ according to Definition 5. Finally sort all edges in the MCBT of ***MCS***. After that, we can obtain an edge set $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$ such that $ce_1 \le ce_2 \ldots \le ce_{Kclu-1}$. That is:

$mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\} \leftarrow$ Prim($\boldsymbol{MC}$); /* Suppose $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\}$ is sorted by an increased sequence such that $e_1 \le e_2 \ldots \le e_{Kclu-1}$. */

$mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\} \leftarrow$ ReplaceWeights($\boldsymbol{MCS}$, $mst(\boldsymbol{MC})$); /* Suppose $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$ is sorted by an increased sequence such that $ce_1 \le ce_2 \ldots \le ce_{Kclu-1}$. */

(Sc). Just like the step (Sc) of ***Strategy 1***, select some minimum edges from $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$. These selected edges are less than the other edges. Suppose there are *NumMinMcbt* minimum edges. Usually, (*Kclu* - *NumMinMcbt*) is equal to the right number of clusters in those data sets with obvious clusters. There are also four cases in $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$ just like that in $mst(\boldsymbol{MC})$.

(Sd). Merge two near neighbor micro-clusters connected by any edge in the *NumMinMcbt* minimum edges of $mcbt(\boldsymbol{MCS})$ if they can be merged. That is:

for (i = 1; $i \le NumMinMcbt$; i++)
{
    if (two near neighbor micro-clusters connected by ce_i can be merged)
        The two near neighbor micro-clusters should be merged.
}

4.3.3 Two judgement methods of merging two micro-clusters in ***Strategy 1***

In step (Sd) of ***Strategy 1***, the following any judgement method of merging two micro-clusters can be used.

(1) ***Method 1***: The middle-point judgement method

In the front *NumMinMst* edges of the increased edge set $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\}$, suppose edge e_i ($i = 1, 2, \ldots, NumMinMst$) connects two micro-cores, MC_u and MC_v, of two micro-clusters, MCS_u and MCS_v. We compute the middle point in the line of the two micro-cores MC_u and MC_v, $Line(MC_u, MC_v)$. The computation equation of the middle point in the line $Line(MC_u, MC_v)$ is presented by:

$$middlepoint(MC_u, MC_v) = (MC_u + MC_v) / 2 \quad (6)$$

One kind of density measure of the middle point can be used to judge the merging of two micro-clusters. Suppose $\sigma(middlepoint(MC_u, MC_v))$ is the σ near neighbor point set of point $middlepoint(MC_u, MC_v)$, and $|\sigma(middlepoint(MC_u, MC_v))|$ is the number of points in $\sigma(middlepoint(MC_u, MC_v))$. The density measure index, $|\sigma(middlepoint(MC_u, MC_v))|$, can be used to judge the merging of two micro-clusters, MCS_u and MCS_v.

The judgement rule of merging two near neighbor micro-clusters in ***Strategy 1*** is:

if ($|\sigma(middlepoint(MC_u, MC_v))| \geq MinPts$)

Two micro-clusters, MCS_u and MCS_v, can be merged;

else

Two micro-clusters, MCS_u and MCS_v, can not be merged;

Here, parameter σ is a range threshold and parameter *MinPts* is a density threshold.

(2) ***Method 2***: The three-points judgement method

In the front *NumMinMst* edges of the increased edge set $mst(\boldsymbol{MC}) = \{e_1, e_2, \ldots, e_{Kclu-1}\}$, suppose edge e_i ($i = 1, 2, \ldots, NumMinMst$) connects two micro-cores, MC_u and MC_v, of two micro-clusters, MCS_u and MCS_v. We compute three points in the line of the two micro-cores MC_u and MC_v, $Line(MC_u, MC_v)$.

The first point is the middle point of $Line(MC_u, MC_v)$, $middlepoint(MC_u, MC_v)$. The computation equation of $middlepoint(MC_u, MC_v)$ is presented by Eq. (6).

The second point of $Line(MC_u, MC_v)$, $leftpoint_\sigma(MC_u, MC_v)$, is a left point near $middlepoint(MC_u, MC_v)$ with a distance that is equal to parameter σ.

The third point of $Line(MC_u, MC_v)$, $rightpoint_\sigma(MC_u, MC_v)$, is a right point near $middlepoint(MC_u, MC_v)$ with a distance that is equal to parameter σ.

Easily, we can get two computation equations of the second point and the third point in the line $Line(MC_u, MC_v)$.

$$leftpoint_\sigma(MC_u, MC_v) = middlepoint(MC_u, MC_v) - \frac{\sigma}{dis(MC_u, MC_v)} \bullet (MC_v - MC_u) \quad (7)$$

$$rightpoint_{\sigma}(MC_u, MC_v) = middlepoint(MC_u, MC_v) + \frac{\sigma}{dis(MC_u, MC_v)} \bullet (MC_v - MC_u) \quad (8)$$

One kind of density measure of three points, $leftpoint_{\sigma}(MC_u, MC_v)$, $middlepoint(MC_u, MC_v)$, and $rightpoint_{\sigma}(MC_u, MC_v)$, can be used to judge the merging of two micro-clusters, MCS_u and MCS_v. Just like the above, the three density measure indexes, $|\sigma(leftpoint_{\sigma}(MC_u, MC_v))|$, $|\sigma(middlepoint(MC_u, MC_v))|$, and $|\sigma(rightpoint_{\sigma}(MC_u, MC_v))|$, are used to judge the merging of two micro-clusters, MCS_u and MCS_v.

The judgement rule of merging two micro-clusters is:

if $((|\sigma(leftpoint(MC_u, MC_v))| \geq MinPts)$ **and** $(|\sigma(middlepoint(MC_u, MC_v))| \geq MinPts)$ **and** $(|\sigma(rightpoint(MC_u, MC_v))| \geq MinPts))$

Two micro-clusters, MCS_u and MCS_v, can be merged;

else

Two micro-clusters, MCS_u and MCS_v, can not be merged;

4.3.4 Two judgement methods of merging two micro-clusters in ***Strategy 2***

In step (Sd) of ***Strategy 2***, the following any judgement method of merging two micro-clusters can be used.

(1) ***Method 1***: The middle-point judgement method

In the front *NumMinMcbt* edges of the increased edge set $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$, suppose edge ce_i ($i = 1, 2, \ldots, NumMinMcbt$) connects two nearest points, MP_u and MP_v, of two micro-clusters, MCS_u and MCS_v. We compute the middle point in the line of two nearest points MP_u and MP_v, $Line(MP_u, MP_v)$. The computation equation of the middle point in the line $Line(MP_u, MP_v)$ is presented by:

$$middlepoint(MP_u, MP_v) = (MP_u + MP_v) / 2 \quad (9)$$

One kind of density measure of the middle point in the line $Line(MP_u, MP_v)$ can be used to judge the merging of two micro-clusters. Suppose $\sigma(middlepoint(MP_u, MP_v))$ is the σ near neighbor point set of point $middlepoint(MP_u, MP_v)$, and $|\sigma(middlepoint(MP_u, MP_v))|$ is the number of points in $\sigma(middlepoint(MP_u, MP_v))$. The density measure index, $|\sigma(middlepoint(MP_u, MP_v))|$, can be used to judge the merging of two micro-clusters, MCS_u and MCS_v.

The judgement rule of merging two near neighbor micro-clusters in ***Strategy 2*** is:

if $(|\sigma(middlepoint(MP_u, MP_v))| \geq MinPts)$

Two micro-clusters, MCS_u and MCS_v, can be merged;

else

Two micro-clusters, MCS_u and MCS_v, can not be merged;

(2) ***Method 2***: The three-points judgement method

In the front *NumMinMcbt* edges of the increased edge set $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu-1}\}$, suppose edge ce_i ($i = 1, 2, \ldots, NumMinMcbt$) connects two nearest points, MP_u and MP_v, of two micro-clusters, MCS_u and MCS_v. We compute three points in the line of two nearest points MP_u and MP_v, $Line(MP_u, MP_v)$.

The first point is the middle point of $Line(MP_u, MP_v)$, $middlepoint(MP_u, MP_v)$. The computation equation of $middlepoint(MP_u, MP_v)$ is presented by Eq. (9).

The second point of $Line(MP_u, MP_v)$, $leftpoint_\sigma(MP_u, MP_v)$, is a left point near the middle point with a distance that is equal to parameter σ.

The third point of $Line(MP_u, MP_v)$, $rightpoint_\sigma(MP_u, MP_v)$, is a right point near the middle point with a distance that is equal to parameter σ.

Easily, we can get two computation equations of the second point and the third point in the line $Line(MP_u, MP_v)$.

$$leftpoint_\sigma(MP_u, MP_v) = middlepoint(MP_u, MP_v) - \frac{\sigma}{dis(MP_u, MP_v)} \bullet (MP_v - MP_u) \quad (10)$$

$$rightpoint_\sigma(MP_u, MP_v) = middlepoint(MP_u, MP_v) + \frac{\sigma}{dis(MP_u, MP_v)} \bullet (MP_v - MP_u) \quad (11)$$

Here, according to Definition 5, there is $dis(MP_u, MP_v) = dis_{\min}(MCS_u, MCS_v)$.

One kind of density measure of three points in the line $Line(MP_u, MP_v)$, $leftpoint_\sigma(MP_u, MP_v)$, $middlepoint(MP_u, MP_v)$, and $rightpoint_\sigma(MP_u, MP_v)$, can be used to judge the merging of two micro-clusters, MCS_u and MCS_v. Just like the above, the three density measure indexes, $|\sigma(leftpoint_\sigma(MP_u, MP_v))|$, $|\sigma(middlepoint(MP_u, MP_v))|$, and $|\sigma(rightpoint_\sigma(MP_u, MP_v))|$, are used to judge the merging of two micro-clusters, MCS_u and MCS_v.

The judgement rule of merging two micro-clusters is:

if $((|\sigma(leftpoint(MP_u, MP_v))| \geq MinPts)$ **and** $(|\sigma(middlepoint(MP_u, MP_v))| \geq MinPts)$ **and** $(|\sigma(rightpoint(MP_u, MP_v))| \geq MinPts))$

Two micro-clusters, MCS_u and MCS_v, can be merged;

else

Two micro-clusters, MCS_u and MCS_v, can not be merged;

4.4 Time complexity analysis of CESynC algorithm

The time complexity of the original ESynC algorithm is Time = $O(Tdn^2)$, where T

is the times of synchronization, n is the number of data points, and d is the dimension of data points. An improved version of ESynC algorithm by using some efficient index structures (such as R-tree, R^*-tree, et al.), Time = $O(Tdn \cdot logn)$ can be obtained in many low-dimensional data sets. So Step1 of CESynC algorithm needs Time = $O(Tdn^2)$, even Time = $O(Tdn \cdot logn)$.

In Step2, the simplest implementation of fetching K_{clu} steady locations and K_{iso} isolates from $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$ needs Time = $O(dn^2)$. Another implementation with a little trick by recording current steady locations when scanning each point of the data set needs Time = $O(dn \cdot (K_{clu} + K_{iso}))$.

In Step3, constructing an MST from K_{clu} micro-cores using Prim algorithm needs Time = $O(d \cdot (K_{clu})^2)$. Constructing an MCBT from K_{clu} micro-clusters based on the $mst(\boldsymbol{MC})$ needs Time = $O\left(d \cdot \sum_{e(u,v) \in MST} \left(\left|cluster_u\right| \cdot \left|cluster_v\right|\right)\right)$.

In Step4, if each edge that connects two micro-clusters needs to judge the merging operation, CESynC algorithm needs $O(K_{cl} - 1)$ judgements. If two micro-clusters need to be merged, then two subsets that contain their corresponding data points need to do a union operation. If use a disjoint-set data structure, the union operation is very simple and efficient.

Step5 needs Time = $O(n)$ if using a disjoint-set data structure.

4.5 The setting of parameters in CESynC algorithm

4.5.1 The setting of range parameter δ in CESynC algorithm

Parameter δ is important for clustering quality in CESynC algorithm and ESynC algorithm. In CESynC algorithm, ESynC algorithm is called at first. In the stage of merging micro-clusters, parameter δ is not used again. So the setting of range parameter δ in CESynC algorithm is the same as that in ESynC algorithm and SynC algorithm. In Böhm et al. (2010), parameter δ is optimized by the MDL principle. Similarly, two other methods were presented to estimate parameter δ that is used in CNNI algorithm (Chen, 2015). How to select a proper value for parameter δ is discussed in Chen (2017, 2018). They are summarized below.

(1) The optimization of parameter δ in Böhm et al. (2010)

Parameter δ can affect the results of clusters. In Böhm et al. (2010), parameter δ is optimized by a heuristic method and the MDL principle. In the heuristic method presented by Böhm et al. (2010), parameter δ is initialized with the average value of the k-nearest neighbor distance determined from the data set for a small k. For example,

k = 3 is recommended in their experiments. Then parameter δ is increased with a reasonable step size.

(2) The heuristic selection of parameter δ

If parameter δ is satisfied with Eq. (4), then the data set $S = \{X_1, X_2, \ldots, X_n\}$ using ESynC algorithm for clustering will obtain an effective result of local synchronization with some obvious clusters or isolates.

But sometimes there is $\max\{longestDistance(cluster_k) \mid k = 1, 2, \ldots, K_{clu}\} > \min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}$ for some data sets. In this case, if we set $\delta > \min\{dis_{\min}(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K_{clu}\}$, some obvious clusters may be detected.

(3) A linear-searching exploring method of parameter δ

Usually, parameter δ has a very long valid interval for many kinds of data sets with obvious clusters. Some simulated experiments in Chen (2017) and this paper also validate this conclusion. Some times, parameter δ have several long valid intervals for different clustering levels. So we can explore the valid interval of parameter δ by the linear-searching method.

4.5.2 The setting of the range parameter σ in *Method 1* and *Method 2* of CESynC algorithm

Parameter σ affects the merging selection in CESynC algorithm. Parameter σ is often designed by the following rule:

$$\sigma \in [\sigma_{\min}, \sigma_{\max}]. \tag{12}$$

Usually, $\sigma_{\min} \approx \left\lceil \frac{ce_{NumMinMcbt}}{2} \right\rceil$ and $\sigma_{\max} \approx \left\lfloor \frac{ce_{(NumMinMcbt+1)}}{2} \right\rfloor$ are satisfied in many data sets. In the $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu\text{-}1}\}$, $(ce_{NumMinMcbt}/2, ce_{(NumMinMcbt+1)}/2)$ is often an inflexion-point of $mcbt(\boldsymbol{MCS})$. Here $\left\lceil \frac{ce_{NumMinMcbt}}{2} \right\rceil$ is the integer ceiling of $ce_{NumMinMcbt}/2$ and $\left\lfloor \frac{ce_{(NumMinMcbt+1)}}{2} \right\rfloor$ is the integer floor of $ce_{(NumMinMcbt+1)}/2$).

In subsection 5.3.4, we explore the relation between parameter σ and parameter δ in CESynC algorithm by using nine artificial data sets. We observe that the valid interval of parameter σ is affected by parameters δ and *MinPts*.

4.5.3 The setting of parameter *MinPts* in CESynC algorithm

Parameter *MinPts* is a density threshold that is used to filter isolates and to judge the merging operation of two micro-clusters. In our simulations, it is set from 1 to 4. When *MinPts* = 1, it means that there is no isolates in the synchronization clustering

stage of CESynC algorithm. When *MinPts* = 2, 3, or 4, those isolates that its number of points is less than *MinPts* can be filtered after the synchronization clustering.

Usually, the valid interval of parameter δ in CESynC algorithm is longer when *MinPts* = 1 or 2, and it is shorter when *MinPts* = 3 or 4.

4.6 The improvement of CESynC algorithm

In CESynC algorithm, one improved version of ICESynC algorithm can be obtained by combining multidimensional grid partitioning method and Red-Black tree structure or using R-tree structure to construct the near neighbor point sets of all data points. The improving method based on spatial index structures is introduced in Chen (2018).

4.7 The convergence of CESynC algorithm

Because the merging operation of CESynC algorithm is a judging-and-merging process, so the convergence of CESynC algorithm is completely depended on the convergence of ESynC algorithm. According to Chen (2017) and the simulation, CESynC algorithm is also convergent.

4.8 Two extreme cases of CESynC algorithm

CESynC algorithm has two extreme cases. One case is the number of initial micro-clusters after the synchronization clustering stage of CESynC algorithm is equal to the actual number of clusters. In this case, parameter δ in CESynC algorithm is set in the valid interval of parameter δ in ESynC algorithm. At this time, *NumMinMst* or *NumMinMcbt* is equal to zero. So the merging judgement of micro-clusters is not needed.

Another case is when *MinPts* = 1 and parameter δ in CESynC algorithm is set as a small value that is less than the minimum distance of the data set, the number of initial micro-clusters after the synchronization clustering stage of CESynC algorithm is equal to the number of points. At this time, *NumMinMst* or *NumMinMcbt* is equal to $(n - 1)$. So CESynC algorithm needs a merging judgement process of all points. In this case, the MST of micro-clusters is the same as the MCBT of micro-clusters and CESynC algorithm is similar to an MST-based clustering algorithm (Chen, 2013).

5. Simulated experiments

5.1 Experimental design

5.1.1 Experimental environment and the description of experimental data sets

Our experiments are finished in a personal computer (Capability Parameters:

Intel(R) Celeron(R) CPU 3855U 1.6GHz, 8.00G Memory). Experimental programs are developed using C and C++ language under Windows 7.

To verify the improvements in clustering quality or clustering validity of CESynC algorithm, there will be some experimental comparison among SynC algorithm, ESynC algorithm and several classical clustering algorithm on some artificial data sets and eight UCI data sets (Frank and Asuncion, 2010) in the next sections. To validate the improvements in time cost between ICESynC algorithm (an Improved version of CESynC algorithm in time complexity) and CESynC algorithm, there will be an experimental comparison of four data sets.

The original location of some 2-D and 3-D experimental data sets (data0 - data 6, DS0 - DS5, DS9) are presented in **sfig. 1** and **sfig. 2** of Online Resource 1 of Supplementary Material. Seventeen kinds of artificial data sets (DS0 - DS16) are produced in an interval [0, 600] in each dimension by two functions presented in Online Resource 2 of Supplementary Material. Eight UCI data sets are standardized to an interval [0, 600] in each dimension. The original location of three 2-D and 3-D artificial UCI data sets are presented in **sfig. 3** of Online Resource 1 of Supplementary Material. **Table 3** presents the description of the experimental data sets.

Table 3 The description of experimental data sets

(a) The description of seven 2-D data sets (data0 is created by a program and data1- data6 are drawn by hand in a 2-D region referencing the original DBSCAN paper)

Data Sets	Number of Points (*n*)	Number of clusters	With noise
data0	400	1 or 2 or 3 or 4	yes
data1	300	2	no
data2	300	3	no
data3	300	4	no
data4	300	4	no
data5	300	4	yes
data6	300	1 or 2 or 3	no

(b) The description of sixteen kinds of artificial data sets

Data Sets	Predefined (Actual) Number of Clusters	With Noise	Cluster Semidiameter	Dimension (*d*)
DS0	9 (8)	no	30	2
DS1	5 (5)	yes	40	2
DS2	5 (4)	no	50	2
DS3	7 (6)	yes	30	2
DS4	7 (5)	no	40	2
DS5	12 (11)	no	30	2
DS6	12 (12)	no	30	4
DS7	12 (12)	no	30	6
DS8	12 (12)	no	30	8
DS9	5 (5)	no	30	2
DS10	5 (5)	no	30	4
DS11	5 (5)	no	30	6
DS12	5 (5)	no	30	8
DS13	5 (5)	no	30	20
DS14	5 (5)	no	30	40

DS15	5 (5)	no	30	80
DS16	5 (5)	no	30	100

(c) The description of eight UCI data sets (Frank and Asuncion, 2010)

UCI Data Sets	**Number of Points (*n*)**	**Dimension (*d*)**	**Class Distribution**	**Number of Classes**
Iris	150	4	{Setosa: 50, Versicolor: 50, Virginica: 50}	3
Wine	178	13	{1: 59, 2: 71, 3: 48}	3
Wdbc	569	30	{B: 357, M: 212}	2
Glass	214	9	{Window: {FB: 70, FV: 17, NFB: 76}, Non-window: {C: 13, T: 9, H: 29}}	6
Ionosphere	351	34	{Good: 225, Bad: 126}	2
Letter-recognition	20000	16	{A: 443, B: 460, C: 449, …, Z: 408}	26
Segmentation	210	19	{Brickface: 30, Sky: 30, Foliage: 30, Cement: 30, Window: 30, Path: 30, Grass: 30}	7
Cloud	2048	10	{1: 2014, 2: 2014}	2

5.1.2 Measure criteria of performance of algorithms

All comparison results of these algorithms are presented by some tables and figures. The clustering quality and efficiency of CESynC algorithm are evaluated and compared using the following several criteria.

(1) The efficiency of algorithms is measured by time cost (second). The smaller the time cost is, the higher the efficiency is.

(2) Clustering quality of clustering algorithms is measured by display figures of clustering results, the final number of clusters, and three robust information-theoretic measures (Adjusted Mutual Information (AMI) (Vinh et al., 2010), Adjusted Variation of Information (AVI) (Vinh et al., 2010), and Normalized Mutual Information (NMI) (Strehl et al., 2002)). According to Vinh et al. (2010), the higher the value of three measures gets, the better the clustering quality of algorithms is. In simulations, we use the Matlab code from Vinh et al. (2010) to compute AMI and NMI. According to two papers (Strehl et al., 2002; Vinh et al., 2010), we implement a Matlab program to compute AVI.

5.1.3 The description of experiments and the setting of parameters

In our simulated experiments, the maximum times of synchronization evolution in the while repetition of SynC algorithm, ESynC algorithm, and CESynC algorithm is set as 20.

In section 5.2, CESynC algorithm (using *Strategy 1* + *Method 1*) will be compared with SynC algorithm, ESynC algorithm, and some other classic clustering algorithms (K-Means (MacQueen, 1967), FCM (Bezdek, 1981), AP (Frey et al., 2007), DBSCAN (Ester et al., 1996), Mean Shift (Fukunaga et al., 1975; Comaniciu et al., 2002) in clustering quality by using seven 2-D artificial data sets (data0 – data6).

In section 5.3, CESynC algorithm (using (*Strategy 1* + *Method 1*) and (*Strategy 2*

+ *Method 1*)) will be compared with some other clustering algorithms in clustering quality and time cost by using some artificial data sets (from DS0 - DS16). The relation between parameter σ and parameter δ in CESynC algorithm is explored by using nine artificial data sets (from DS5 - DS16) in this section.

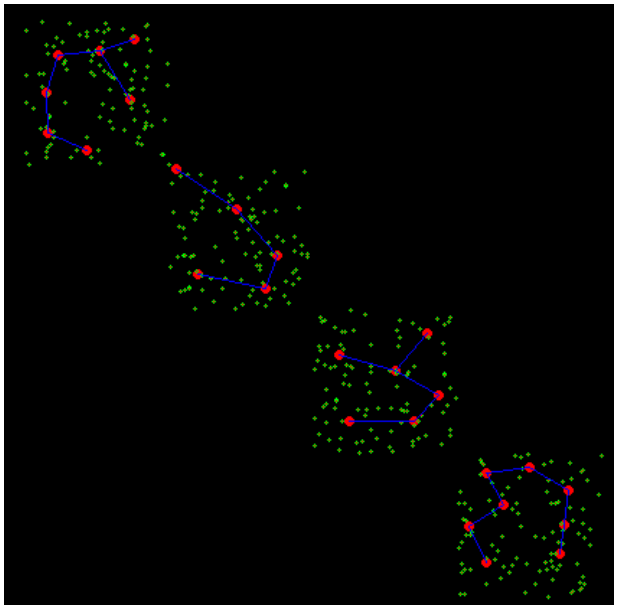
In section 5.4, CESynC algorithm (using *Strategy 2 + Method 1*) will be compared with some other clustering algorithms in clustering quality by using eight UCI data sets.

In the experiments, range parameters δ and σ, and density threshold parameter *MinPts* are used in CESynC algorithm. Range parameter δ is also used in ESynC algorithm and SynC algorithm. Range parameter *Eps* (has similar function to parameter δ) and density threshold parameter *MinPts* are used in DBSCAN algorithm. In the simulations, parameter *MinPts* is set as 4 in DBSCAN algorithm. Bandwidth parameter h (has similar function to parameter δ) is used in Mean Shift algorithm. Parameter k, the predefined number of clusters, is used in K-Means algorithm and FCM algorithm.

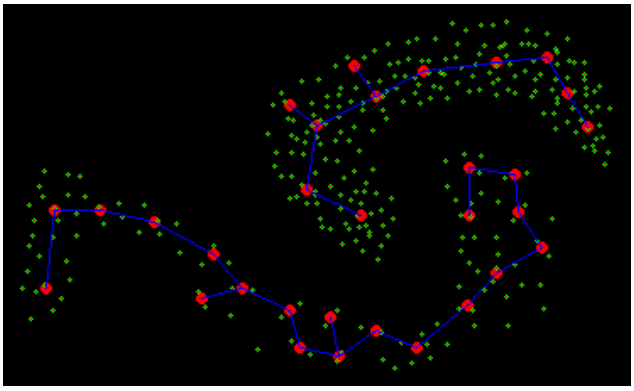
5.2 Experimental results of seven 2-D artificial data sets (data0 – data6)

5.2.1 Comparison of the clustering results among CESynC algorithm (using *Strategy 1 + Method 1*) and some other clustering algorithms

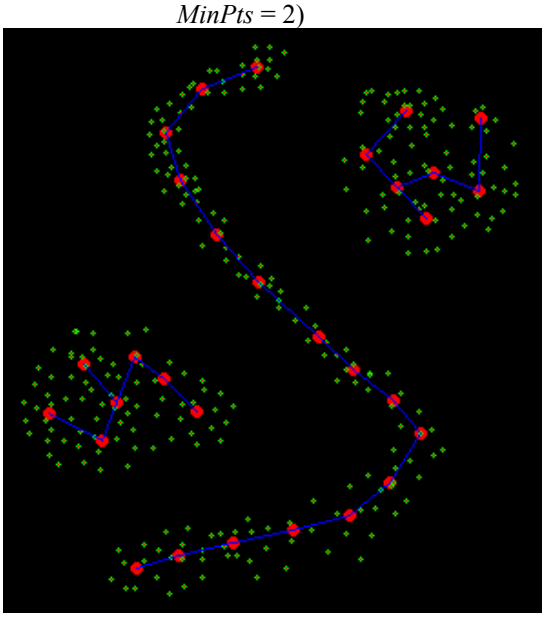
Fig. 2 presents the clustering results of seven 2-D artificial data sets (data0 – data6) by using CESynC algorithm (using *Strategy 1 + Method 1*). In Online Resource 3 of Supplementary Material of this paper, **sfig. 4** presents the clustering results of seven 2-D artificial data sets (data0 – data6) by using SynC algorithm, ESynC algorithm, DBSCAN algorithm, Mean Shift algorithm, AP algorithm, K-Means algorithm, and FCM algorithm.



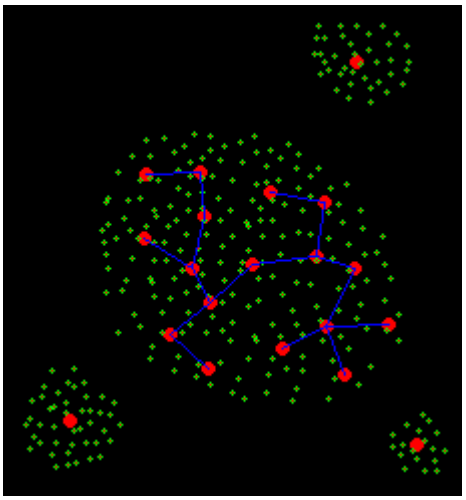
(a) 4 Clusters of data0 identified by CESynC algorithm (using *Strategy 1 + Method 1,* $\delta = \sigma = 18$, *MinPts* = 4)



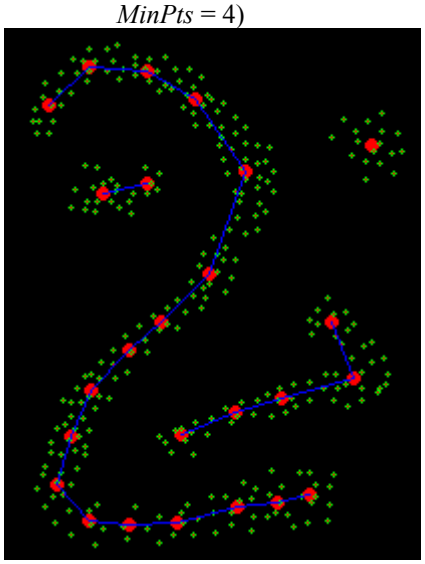
(b) 2 Clusters of data1 identified by CESynC algorithm (using *Strategy 1 + Method 1,* $\delta = \sigma = 14$, *MinPts* = 2)



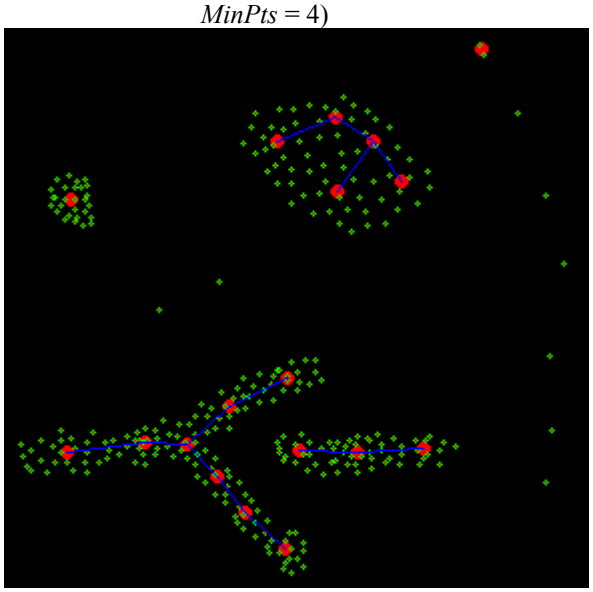
(c) 3 Clusters of data2 identified by CESynC algorithm (using *Strategy 1 + Method 1,* $\delta = \sigma = 14$, *MinPts* = 2)



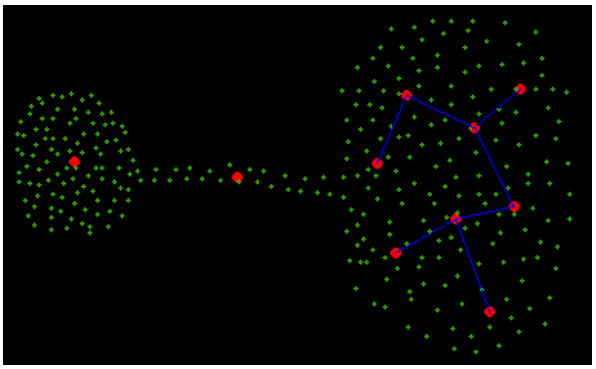
(d) 4 Clusters of data3 identified by CESynC algorithm (using *Strategy 1 + Method 1,* $\delta = \sigma = 14$, *MinPts* = 4)



(e) 4 Clusters of data4 identified by CESynC algorithm (using *Strategy 1 + Method 1,* $\delta = \sigma = 12$, *MinPts* = 4)



(f) 4 Clusters and 9 isolates of data5 identified by CESynC algorithm (using *Strategy 1 + Method 1,* $\delta = \sigma = 12$, *MinPts* = 4)



(g) 3 Clusters of data6 identified by CESynC algorithm (using *Strategy 1* + *Method 1*, $\delta = 27$, $\sigma = 18$, *MinPts* = 4)

Fig. 2 The clustering results of seven 2-D artificial data sets (data0 – data6) by using CESynC algorithm. In Fig. 2, the original location of data points are draw by small square with green color, the steady locations of K_{clu} microclusters that are obtained by synchronization evolution based on ESynC algorithm are draw by large sphere with red color, and the (K_{clu} - 1) edges connected K_{clu} steady locations are draw by thin line with blue color.

5.2.2 Comparison of the valid interval of parameters δ, *Eps*, or *h* among CESynC algorithm, SynC algorithm, ESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm

Table 4. Comparison of the valid interval of parameter δ, *Eps*, or *h* among CESynC algorithm, SynC algorithm, ESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm using seven 2-D artificial data sets. In CESynC, parameter *MinPts* is set as 2 for data1 and data2, and it is set as 4 for other five data sets. In CESynC, parameter σ is set according to Eq. (12).

(a) The valid interval of parameter δ, *Eps*, or *h* among CESynC, SynC, ESynC, DBSCAN, and Mean Shift

Data sets	The valid interval of parameter δ, *Eps*, or *h*						The $[e_k, e_{k+1}]$ in the MST of the complete graph of the data set
	CESynC (*Strategy1* + *Method 1*)	CESynC (*Strategy2* + *Method 1*)	SynC	ESynC	DBSCAN	Mean Shift	
data0	**[14, 15]** or **[18, 83]**	**[15, 17]** or **[18, 83]**	Null	[35, 40] ∪ [54, 78] ∪ {47, 51, 52, 80}	{18}	[46, 113]	[16.23, 24.19]
data1	**[14, 16]**	**[14, 31]**	Null	Null	[24, 31]	Null	[23.35, 31.06]
data2	**[12, 19]**	**[12, 19]**	Null	Null	[17, 41]	Null	[15.3, 41.68]
data3	**[13, 18]** ∪ [34, 62]	**[15**, 62]	Null	[34, 62]	[13, 28]	[44, 49]	[12.37, 28.43]
data4	**[11, 14]**	**[13, 14]**	Null	Null	[13, 22]	Null	[11.66, 22.85]
data5	**{12}**	**{12}**	Null	Null	[11, 18]	Null	[10, 18.68]
data6	[26, 46] or [18, 25] ∪ {77} ∪ [83, 91] ∪ [93, 130] or [131, +∞)	[24, 28]	Null	[47, 66] or {77} ∪ [83, 91] ∪ [93, 130] or [131, +∞)	[19, +∞)	[72, 156] or {169} ∪ [171, +∞)	Null

(b) The number of clusters / isolates obtained by using CESynC, SynC, ESynC, DBSCAN, Mean Shift, and MST-based clustering algorithm

Data sets	The number of clusters / isolates						
	CESynC (*Strategy1* + *Method 1*)	CESynC (*Strategy2* + *Method 1*)	SynC	ESynC	DBSCAN	Mean Shift	MST-based clustering
data0	3 / 0 or 4 / 0	3 / 0 or 4 / 0	-	4 / 0	4 / 0	4 / 0	3 / 0
data1	2 / 0	2 / 0	-	-	2	-	2 / 0
data2	3 / 0	3 / 0	-	-	3	-	3 / 0
data3	4 / 0	4 / 0	-	4 / 0	4	4 / 0	4 / 0
data4	4 / 0	4 / 0	-	-	4	-	4 / 0
data5	4 / 9	4 / 9	-	-	4 / 9	-	4 / 9
data6	3 / 0 or 2 / 0 or 1 / 0	1 / 0	-	3 / 0 or 2 / 0 or 1 / 0	2 / 0 or 1 / 0	2 / 0 or 1 / 0	1 / 0

Table 4 presents the comparison results among these clustering algorithms. Here, $[e_k , e_{k+1}]$ can be obtained from Eq.(8) of Chen (2015). In Table 4, intercomparing CESynC algorithm (using (*Strategy1* + *Method 1*) and (*Strategy2* + *Method 1*)), SynC algorithm, ESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm, we observe that the valid interval of parameters δ in CESynC algorithm is longer or has some improvements than that in ESynC algorithm in some cases. In this simulation, the valid interval of parameter δ between CESynC algorithm (*Strategy1* + *Method 1*) and CESynC algorithm (*Strategy1* + *Method 2*) has little difference.

5.3 Experimental results of some artificial data sets (from DS0 - DS16)

5.3.1 Comparison of the valid interval of parameter δ, *Eps*, or h among CESynC algorithm, SynC algorithm, ESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm using some artificial data sets (from DS0 - DS16, n = 2000)

Table 5 presents the comparison results among these clustering algorithms. Here, $[e_k , e_{k+1}]$ can be obtained from Eq.(8) of Chen (2015). In Table 5, intercomparing CESynC algorithm (using (*Strategy1* + *Method 1*) and (*Strategy2* + *Method 1*)), SynC algorithm, ESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm, we observe that the valid interval of parameter δ in CESynC algorithm is longer or has some improvements than that in ESynC algorithm in some cases. In this simulation, the valid interval of parameter δ between CESynC algorithm (*Strategy1* + *Method 1*) and CESynC algorithm (*Strategy1* + *Method 2*) has little difference.

Table 5. Comparison of the valid interval of parameter δ, *Eps*, or *h* among CESynC algorithm, SynC algorithm, ESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm using seventeen artificial data sets (from DS0 – DS16). In CESynC algorithm, parameter *MinPts* is set as 2 for DS0 – DS16, and parameter σ is set according to Eq. (12).

(a) The valid interval of parameter δ, *Eps*, or *h* among CESynC, SynC, ESynC, DBSCAN, and Mean Shift

Data sets	The valid interval of parameter δ, *Eps*, or *h*						The $[e_k , e_{k+1}]$ in the MST of the complete graph of the data set
	CESynC (*Strategy1* + *Method 1*)	CESynC (*Strategy2* + *Method 1*)	SynC	ESynC	DBSCAN	Mean Shift	
DS0	[**16**, **43**]	[**16**, **43**]	Null	[19, 40]	[10, 15]	[29, 57]	[8.54, 15.34]
DS1	[**22**, 33]	[**22**, 33]	Null	[24, 33]	[10, 14]	Null	[9.00, 14.94]
DS2	[**20**, 142]	[**20**, 142]	Null	[56, 142]	[12, 55]	[86, 123]	[8.94, 55.19]
DS3	[**16**, 28]	[**16**, 28]	Null	[19, 28]	[9, 16]	[29, 50]	[8.54, 16.53]
DS4	{**11, 18**} ∪ [**14**, **15**] ∪ [**22**, **38**]	{**11, 18**} ∪ [**14**, **15**] ∪ [**22**, **38**]	Null	Null	[10, 14]	Null	[9.90, 14.29]
DS5	{**12**, **16**} ∪ [**19**, 43]	{**12**, **16**} ∪ [**19**, 43]	Null	[21, 43]	[10, 17]	[31, 34]	[9.06, 17.29]
DS6	[27, 145]	[27, 145]	Null	[27, 145]	[27, 112]	[35, 145]	[26.42, 112.41]
DS7	[39, 191]	[39, 191]	Null	[39, 191]	[39, 150]	[44, 204]	[38.25, 150.64]
DS8	[53, 276]	[53, 276]	Null	[53, 276]	[53, 241]	[53, 272]	[52.80, 241.80]
DS9	[**11**, **14**] ∪ [**17**, 72]	[**11**, **14**] ∪ [**17**, 72]	Null	[18, 72]	[7, 35]	[26, 69]	[6.00, 35.95]
DS10	[24, 187]	[24, 187]	Null	[24, 187]	[24, 151]	[35, 198]	[23.17, 151.11]
DS11	[35, 223]	[35, 223]	Null	[35, 223]	[35, 188]	[39, 244]	[34.45, 188.28]
DS12	[46, 275]	[46, 275]	Null	[46, 275]	[46, 240]	[47, 288]	[45.88, 240.39]
DS13	[88, 826]	[88, 826]	Null	[88, 826]	[88, 800]	[88, 842]	[87.94, 800.24]
DS14	[129, 1239]	[129, 1239]	Null	[129, 1239]	[139, 1220]	[139, 1266]	[138.37, 1220.47]
DS15	[178, 1824]	[178, 1824]	Null	[178, 1824]	[204, 1805]	[188, 1848]	[203.83, 1805.51]
DS16	[201, 2078]	[201, 2078]	Null	[201, 2078]	[232, 2060]	[223, 2100]	[231.34, 2060.60]

(b) The number of clusters / isolates obtained by using CESynC, SynC, ESynC, DBSCAN, Mean Shift, and MST-based clustering algorithm

Data sets	The number of clusters / isolates						
	CESynC (*Strategy1 + Method 1*)	CESynC (*Strategy2 + Method 1*)	SynC	ESynC	DBSCAN	Mean Shift	MST-based clustering
DS0	8 / 0	8 / 0	-	8 / 0	8 / 0	8 / 0	8 / 0
DS1	5 / 9	5 / 9	-	5 / 9	5 / 9	5 / 9	5 / 9
DS2	4 / 0	4 / 0	-	4 / 0	4 / 0	4 / 0	4 / 0
DS3	6 / 17	6 / 17	-	6 / 17	6 / 17	6 / (16 ~ 22)	6 / 17
DS4	5 / 0	5 / 0	-	-	5 / 0	-	5 / 0
DS5	11 / 0	11 / 0	-	11 / 0	11 / 0	11 / 0	11 / 0
DS6	12 / 0	12 / 0	-	12 / 0	12 / 0	12 / 0	12 / 0
DS7	12 / 0	12 / 0	-	12 / 0	12 / 0	12 / 0	12 / 0
DS8	12 / 0	12 / 0	-	12 / 0	12 / 0	12 / 0	12 / 0
DS9	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS10	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS11	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS12	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS13	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS14	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS15	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0
DS16	5 / 0	5 / 0	-	5 / 0	5 / 0	5 / 0	5 / 0

5.3.2 Comparison of the clustering quality among CESynC algorithm and some other clustering algorithms using some artificial data sets (from DS1 – DS16, $n = 2000$)

Table 6 presents the comparison results of the clustering quality among CESynC algorithm (using (*Strategy1 + Method 1*) and (*Strategy2 + Method 1*)), SynC algorithm, ESynC algorithm, DBSCAN algorithm, Mean Shift algorithm, AP algorithm, K-Means algorithm, and FCM algorithm by using sixteen kinds of artificial data sets (from DS1 – DS16).

Table 6. Comparison of the clustering quality of several clustering algorithms (CESynC (using (*Strategy1 + Method 1*) and (*Strategy2 + Method 1*)), SynC, ESynC, and several classical clustering algorithms) using sixteen artificial data sets from DS1 to DS16 ($n = 2000$). In Table 6, parameter δ, *Eps*, *h*, or *k* in these clustering algorithms gets its minimum value from its valid interval or an acceptable value. In CESynC algorithm, parameter *MinPts* is set as 2 for DS1 - DS16, and parameter σ is set according to Eq. (12).

(a) The setting of parameters in several clustering algorithms

UCI Data Sets	parameter δ in CESynC	parameter δ in SynC and ESynC	parameter k in K-Means and FCM	parameter *Eps* in DBSCAN	parameter h in Mean Shift
DS1	$\delta = 22$	$\delta = 24$	$k = 5$	*Eps* = 10	$h = 38$
DS2	$\delta = 20$	$\delta = 56$	$k = 4$	*Eps* = 12	$h = 86$
DS3	$\delta = 16$	$\delta = 19$	$k = 6$	*Eps* = 9	$h = 29$
DS4	$\delta = 11$	$\delta = 43$	$k = 5$	*Eps* = 10	$h = 69$
DS5	$\delta = 19$	$\delta = 21$	$k = 11$	*Eps* = 10	$h = 31$
DS6	$\delta = 27$	$\delta = 27$	$k = 12$	*Eps* = 27	$h = 35$
DS7	$\delta = 39$	$\delta = 39$	$k = 12$	*Eps* = 39	$h = 44$
DS8	$\delta = 53$	$\delta = 53$	$k = 12$	*Eps* = 53	$h = 53$
DS9	$\delta = 17$	$\delta = 18$	$k = 5$	*Eps* = 7	$h = 26$
DS10	$\delta = 24$	$\delta = 24$	$k = 5$	*Eps* = 24	$h = 35$
DS11	$\delta = 35$	$\delta = 35$	$k = 5$	*Eps* = 35	$h = 39$
DS12	$\delta = 46$	$\delta = 46$	$k = 5$	*Eps* = 46	$h = 47$
DS13	$\delta = 88$	$\delta = 88$	$k = 5$	*Eps* = 88	$h = 88$
DS14	$\delta = 129$	$\delta = 129$	$k = 5$	*Eps* = 139	$h = 139$
DS15	$\delta = 178$	$\delta = 178$	$k = 5$	*Eps* = 204	$h = 188$
DS16	$\delta = 201$	$\delta = 201$	$k = 5$	*Eps* = 232	$h = 223$

(b) Three robust information-theoretic measures (AMI /AVI / NMI) and the number of clusters / isolates obtained by using CESynC, SynC, ESynC, K-Means, FCM, AP, DBSCAN, and Mean Shift clustering algorithm

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS1	DS2	DS3	DS4
AMI / AVI / NMI	CESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	SynC	0.1260 / 0.2238 / 0.4977	0.0859 / 0.1583 / 0.4417	0.1619 / 0.2787 / 0.5314	0.1126 / 0.2024 / 0.4816
	ESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	0.9957 / 0.9959 / 0.9959
	K-Means	0.9785 / 0.9881 / 0.9882	**1.0000** / **1.0000** / **1.0000**	0.7039 / 0.7497 / 0.7542	0.8235 / 0.8442 / 0.8448
	FCM	0.9785 / 0.9881 / 0.9882	0.9522 / 0.9557 / 0.9557	0.9683 / 0.9839 / 0.9842	0.9864 / 0.9873 / 0.9873
	AP	0.4091 / 0.5790 / 0.6448	0.3254 / 0.4910 / 0.5766	0.4550 / 0.6223 / 0.6792	0.3844 / 0.5554 / 0.6268
	DBSCAN	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	Mean Shift	0.9725 / 0.9727 / 0.9730	**1.0000** / **1.0000** / **1.0000**	0.9996 / 0.9998 / 0.9998	0.9180 / 0.9244 / 0.9246
The number of clusters / isolates	CESynC	5 / 9	4 / 0	6 / 17	5 / 0
	SynC	889 / 0	1050 / 0	730 / 0	941 / 0
	ESynC	5 / 9	4 / 0	6 / 17	5 / 0
	K-Means	5	4 / 0	6 / 0	5 / 0
	FCM	5	4 / 0	6 / 0	5 / 0
	AP	50	56 / 0	48 / 0	53 / 0
	DBSCAN	5 / 9	4 / 0	6 / 17	5 / 0
	Mean Shift	5 / 9	4 / 0	6 / 16	5 / 0

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS5	DS6	DS7	DS8
AMI / AVI / NMI	CESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	SynC	0.1397 / 0.2452 / 0.5911	0.0012 / 0.0025 / 0.5720	0.0019 / 0.0037 / 0.5722	5.3818e-15 / 1.0764e-14 / 0.5718
	ESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	K-Means	0.7896 / 0.8212 / 0.8239	0.9057 / 0.9277 / 0.9288	0.9057 / 0.9277 / 0.9288	0.9530 / 0.9649 / 0.9654
	FCM	0.9968 / 0.9968 / 0.9968	0.2725 / 0.4283 / 0.5228	0.2725 / 0.4283 / 0.5228	0.2781 / 0.4348 / 0.5273
	AP	0.6054 / 0.7542 / 0.7862	0.6827 / 0.8114 / 0.8319	0.6827 / 0.8114 / 0.8319	0.7963 / 0.8866 / 0.8947
	DBSCAN	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	Mean Shift	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
The number of clusters / isolates	CESynC	11 / 0	12 / 0	12 / 0	12 / 0
	SynC	1023 / 0	1990 / 0	1985 / 0	2000 / 0
	ESynC	11 / 0	12 / 0	12 / 0	12 / 0
	K-Means	11 / 0	12 / 0	12 / 0	12 / 0
	FCM	11 / 0	2 / 0	2 / 0	2 / 1
	AP	47 / 0	37 / 0	37 / 0	23 / 0
	DBSCAN	11 / 0	12 / 0	12 / 0	12 / 0
	Mean Shift	11 / 0	12 / 0	12 / 0	12 / 0

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS9	DS10	DS11	DS12
AMI / AVI / NMI	CESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	SynC	0.1740 / 0.2964 / 0.5095	0.0022 / 0.0044 / 0.4607	1.2712e-14 / 2.5423e-14 / 0.4602	1.2712e-14 / 2.5423e-14 / 0.4602
	ESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	K-Means	**1.0000** / **1.0000** / **1.0000**	**1.0000** / 0.8273 / 0.8660	0.8273 / 0.8646 / 0.8659	0.6546 / 0.7308 / 0.7365
	FCM	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	0.4077 / 0.5748 / 0.6304	0.6550 / 0.7916 / 0.8096
	AP	0.4287 / 0.6002 / 0.6596	0.4361 / 0.6073 / 0.6649	0.4827 / 0.6511 / 0.6979	0.5071 / 0.6729 / 0.7147

	DBSCAN	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	Mean Shift	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
The number of clusters / isolates	CESynC	5 / 0	5 / 0	5 / 0	5 / 0
	SynC	596 / 0	1976 / 0	2000 / 0	2000 / 0
	ESynC	5 / 0	5 / 0	5 / 0	5 / 0
	K-Means	5 / 0	5 / 0	5 / 0	5 / 0
	FCM	5 / 0	5 / 0	2 / 0	3 / 0
	AP	41 / 0	39 / 0	28 / 0	24 / 0
	DBSCAN	5 / 0	5 / 0	5 / 0	5 / 0
	Mean Shift	5 / 0	5 / 0	5 / 0	5 / 0

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS13	DS14	DS15	DS16
AMI / AVI / NMI	CESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	SynC	1.2712e-14 / 2.5423e-14 / 0.4602	1.2712e-14 / 2.5423e-14 / 0.4602	1.2712e-14 / 2.5423e-14 / 0.4602	1.2712e-14 / 2.5423e-14 / 0.4602
	ESynC	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	K-Means	0.8273 / 0.8647 / 0.8659	0.4178 / 0.5894 / 0.6467	0.8273 / 0.8648 / 0.8660	0.5894 / 0.6696 / 0.6768
	FCM	0.4178 / 0.5894 / 0.6467	0.4178 / 0.5894 / 0.6467	0.4178 / 0.5894 / 0.6467	0.4182 / 0.5872 / 0.6428
	AP	0.6215 / 0.7666 / 0.7896	0.6800 / 0.8095 / 0.8255	0.6988 / 0.8227 / 0.8367	0.6998 / 0.8234 / 0.8372
	DBSCAN	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
	Mean Shift	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**	**1.0000** / **1.0000** / **1.0000**
The number of clusters / isolates	CESynC	5 / 0	5 / 0	5 / 0	5 / 0
	SynC	2000 / 0	2000 / 0	2000	2000 / 0
	ESynC	5 / 0	5 / 0	5 / 0	5 / 0
	K-Means	5 / 0	2 / 0	5 / 0	5 / 0
	FCM	2 / 0	2 / 0	2 / 0	3 / 1
	AP	14 / 0	11 / 0	10 / 0	10 / 0
	DBSCAN	5 / 0	5 / 0	5 / 0	5 / 0
	Mean Shift	5 / 0	5 / 0	5 / 0	5 / 0

Note: In Table 6, the largest values of AMI, AVI and NMI and acceptable number of clusters in every data set are shown in bold.

5.3.3 Comparison of the time cost among CESynC algorithm, ICESynC algorithm, SynC algorithm, ISynC algorithm, ESynC algorithm, and IESynC algorithm using four artificial data sets (from DS1, DS2, DS3, and DS5, n = 10000)

Table 7 presents the comparison results of time cost among CESynC algorithm ICESynC algorithm, SynC algorithm, ISynC algorithm, ESynC algorithm, and IESynC algorithm by using four artificial data sets (from DS1, DS2, DS3, and DS5).

Table 7. Comparison of time cost among three synchronization clustering algorithms and their corresponding improved versions based on *R*-tree (SynC, ISynC, ESynC, IESynC, CESynC, and ICESynC) by using four artificial data sets. In Table 7, the number of data points n = 10000, and parameter δ in these clustering algorithms gets its minimum value from its valid interval. In CESynC algorithm, parameter *MinPts* is set as 2 for DS1 – DS5, and parameter σ is set according to Eq. (12).

(a) The setting of parameter in several clustering algorithms

	Name of algorithms	Data sets			
		DS1	DS2	DS3	DS5
The value of parameter	CESynC, ICESynC	$\delta = 22$	$\delta = 20$	$\delta = 16$	$\delta = 19$
	SynC, ISynC, ESynC, IESynC	$\delta = 24$	$\delta = 56$	$\delta = 19$	$\delta = 21$

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS1	DS2	DS3	DS5
Spend time (second)	CESynC	225	364	367	152
	ICESynC	64	**76**	95	30
	SynC	367	388	365	363
	ISynC	63	112	61	48
	ESynC	126	290	165	126
	IESynC	**34**	137	**48**	**28**

Note: The bold in Table 7 marks the best results of IESynC algorithm or ICESynC algorithm.

5.3.4 Exploration of the relation between parameter σ and parameter δ in CESynC algorithm using nine artificial data sets (from DS5 - DS16, $n = 2000$)

Table 8 presents the valid interval of parameter σ among several different values of parameter δ for *MinPts* = 2, 3, 4 in CESynC algorithm (using *Strategy2 + Method 1*) using nine artificial data sets. From **Table 8**, we observe that the valid interval of parameter σ is affected by parameters δ and *MinPts*. In the valid interval of parameter σ for *MinPts* = 1, 2, 3, 4, conjoint micro-clusters will be merged, and disjoint micro-clusters can not be merged. From **Table 8**, we also observe that, if *MinPts* = 2 and without the distortion of isolates, the valid interval of parameter σ satisfies:

$$\sigma \in [\left\lceil \frac{ce_{NumMinMcbt}}{2} \right\rceil, \left\lfloor \frac{ce_{(NumMinMcbt+1)}}{2} \right\rfloor]. \quad (13)$$

When using MCBT-based method, the correct number of clusters locates in the inflexion-point of $mcbt(\boldsymbol{MCS}) = \{ce_1, ce_2, \ldots, ce_{Kclu\text{-}1}\}$, $(ce_{NumMinMcbt}/2, ce_{(NumMinMcbt+1)}/2)$. If there is distortion of isolates, the valid interval of parameter σ is near to $[\left\lceil \frac{ce_{NumMinMcbt}}{2} \right\rceil, \left\lfloor \frac{ce_{(NumMinMcbt+1)}}{2} \right\rfloor]$. When *MinPts* = 1, Eq. (13) is satisfied in many cases. When *MinPts* = 3 and *MinPts* = 4, this conclusion is no more correct.

Table 8 Exploration of the relation between parameter σ and parameter δ in CESynC algorithm (using *Strategy2 + Method 1*) using nine artificial data sets (from DS5 – DS16, $n = 2000$). In Table 8, one or several lines of parameter δ for *MinPts* = 1, 2, 3, 4 that are shown in bold in the valid interval of parameter σ means that CESynC algorithm has no isolates, and parameter $\sigma \in \Phi$ means that CESynC algorithm gets one or no micro-cluster, so there is no merging judgement.

(a) DS5

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 20	**[2, 8]**	**[2, 8]**	**[4, 9]**	**[5, 10]**
Parameter δ = 19	**[2, 8]**	**[2, 8]**	**[3, 9]**	**[5, 10]**
Parameter δ = 18	**[3, 8]**	[3, 8]	[6, 9]	[7, 10]
Parameter δ = 17	**[4, 8]**	[3, 8]	[6, 9]	[7, 10]
Parameter δ = 16	**[3, 8]**	**[3, 8]**	**[6, 9]**	**[7, 10]**
Parameter δ = 15	**[5, 8]**	[5, 8]	[7, 9]	[8, 10]
Parameter δ = 14	**[5, 8]**	[5, 8]	[7, 9]	[9, 10]
Parameter δ = 13	**[5, 11]**	[5, 12]	[7, 12]	[9, 12]
Parameter δ = 12	**[5, 8]**	**[5, 8]**	[8, 9]	[9, 10]
Parameter δ = 11	**[5, 8]**	[5, 8]	[7, 9]	[9, 10]
Parameter δ = 10	**[6, 8]**	[5, 8]	[8, 9]	[9, 10]
Parameter δ = 9	**[7, 8]**	[7, 8]	[8, 9]	{10}

(b) DS9

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 17	**[1, 17]**	**[1, 17]**	**[4, 19]**	**[4, 20]**
Parameter δ = 16	**[1, 17]**	[2, 17]	[4, 19]	[4, 20]
Parameter δ = 15	**[3, 19]**	[2, 19]	[4, 20]	[6, 20]
Parameter δ = 14	**[3, 19]**	**[3, 19]**	[5, 20]	[6, 20]
Parameter δ = 13	**[2, 17]**	**[2, 17]**	**[4, 19]**	**[5, 20]**
Parameter δ = 12	**[2, 17]**	**[2, 17]**	**[4, 19]**	**[5, 20]**
Parameter δ = 11	**[3, 17]**	**[3, 19]**	**[5, 19]**	**[5, 19]**
Parameter δ = 10	**[3, 17]**	[3, 19]	[5, 19]	[5, 19]
Parameter δ = 9	**[3, 17]**	[3, 17]	[4, 19]	[5, 20]

(c) DS10

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 24	**[1, 75]**	**[1, 75]**	**[1, 78]**	**[1, 80]**
Parameter δ = 23	**[12, 75]**	[1, 75]	[1, 78]	[1, 80]
Parameter δ = 22	**[12, 75]**	[1, 75]	[1, 78]	[1, 80]
Parameter δ = 21	**[11, 75]**	[5, 75]	[15, 78]	[16, 80]
Parameter δ = 20	**[14, 75]**	[15, 75]	[15, 78]	[16, 80]
Parameter δ = 19	**[11, 74]**	[11, 77]	[13, 77]	[19, 79]
Parameter δ = 18	**[11, 74]**	[12, 78]	[15, 77]	[17, 79]

(d) DS11

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 35	**[1, 94]**	**[1, 94]**	**[1, 99]**	**[1, 99]**
Parameter δ = 32	**[18, 94]**	[1, 94]	[1, 99]	[1, 99]
Parameter δ = 29	**[24, 97]**	[15, 94]	[1, 99]	[1, 99]
Parameter δ = 26	**[28, 93]**	[21, 90]	[23, 90]	[26, 92]
Parameter δ = 23	**[23, 95]**	[25, 92]	[29, 102]	[30, 102]

(e) DS12

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 46	**[1, 120]**	**[1, 120]**	**[1, 134]**	**[1, 135]**
Parameter δ = 42	**[23, 113]**	[1, 120]	[1, 134]	[1, 135]
Parameter δ = 38	**[31, 125]**	[20, 120]	[1, 134]	[1, 135]
Parameter δ = 34	**[30, 118]**	[20, 132]	[33, 132]	[37, 135]
Parameter δ = 30	**[27, 120]**	[28, 134]	[33, 132]	[33, 135]
Parameter δ = 26	**[24, 120]**	[22, 126]	[31, 132]	[32, 134]

(f) DS13

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 88	**[1, 400]**	**[1, 400]**	**[1, 411]**	**[1, 411]**
Parameter δ = 84	**[44, 400]**	[1, 400]	[1, 411]	[1, 411]
Parameter δ = 80	**[44, 382]**	[1, 400]	[1, 411]	[1, 411]
Parameter δ = 75	**[51, 402]**	[40, 393]	[1, 411]	[1, 411]
Parameter δ = 70	**[50, 406]**	[41, 405]	[68, 404]	[1, 411]
Parameter δ = 65	**[46, 406]**	[44, 400]	[69, 406]	[65, 409]
Parameter δ = 60	**[44, 400]**	[46, 405]	[65, 409]	[65, 410]
Parameter δ = 55	**[44, 400]**	[50, 409]	{54}	Φ
Parameter δ = 50	**[44, 400]**	[46, 408]	Φ	Φ

(g) DS14

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 129	**[1, 610]**	**[1, 610]**	**[1, 620]**	**[1, 620]**
Parameter δ = 120	**[70, 617]**	[1, 610]	[1, 620]	[1, 620]
Parameter δ = 110	**[70, 610]**	[1, 610]	[1, 620]	[1, 620]
Parameter δ = 100	**[73, 610]**	[72, 614]	[101, 620]	Φ
Parameter δ = 90	**[70, 610]**	Φ	Φ	Φ

(h) DS15

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 178	**[1, 902]**	**[1, 902]**	**[1, 918]**	**[1, 920]**
Parameter δ = 170	**[102, 906]**	[1, 902]	[1, 918]	[1, 920]
Parameter δ = 160	**[103, 902]**	[99, 102]	Φ	Φ
Parameter δ = 150	**[102, 902]**	Φ	Φ	Φ

(i) DS16

	Parameter $\sigma \in$			
	MinPts = 1	*MinPts* = 2	*MinPts* = 3	*MinPts* = 4
Parameter δ = 300	**[1, 1030]**	**[1, 1030]**	**[1, 1050]**	**[1, 1052]**
Parameter δ = 201	**[1, 1030]**	**[1, 1030]**	**[1, 1050]**	**[1, 1052]**
Parameter δ = 190	**[116, 1027]**	[1, 1030]	[1, 1050]	[1, 1052]
Parameter δ = 180	**[116, 1030]**	Φ	Φ	Φ

5.4 Experimental results of eight UCI data sets

Because we do not know the true dissimilarity measure of these UCI data sets, all points of these UCI data sets are standardized to an interval [0, 600] in each dimension in the experiments. When computing three information-theoretic measures (AMI, AVI, and NMI), the class labels of these UCI data sets are regarded as their base cluster labels.

Table 9 presents the comparison results of clustering quality among several clustering algorithms (CESynC algorithm (using *Strategy2* + *Method 1*), SynC algorithm, ESynC algorithm, DBSCAN algorithm, Mean Shift algorithm, AP algorithm, K-Means algorithm, and FCM algorithm) using eight UCI data sets. In **Table 9**, by intercomparing these clustering algorithms, we observe that CESynC algorithm gets

the largest values of AMI and AVI in two UCI data sets (Letter-recognition and Cloud) and larger (the same) values than (as) ESynC algorithm. So we can say that CESynC algorithm sometimes gets better clustering results than some clustering algorithms in some UCI data sets. From the final number of clusters in **Table 9**, we observe that CESynC algorithm and ESynC algorithm can get better local synchronization results than SynC algorithm.

Table 9 Comparison of the clustering quality among CESynC algorithm (using *Strategy2 + Method 1*), SynC algorithm, ESynC algorithm, DBSCAN algorithm, Mean Shift algorithm, AP algorithm, K-Means algorithm, and FCM algorithm by using eight UCI data sets. In Table 9, parameter δ, *Eps*, or *h* in these clustering algorithms gets an acceptable value, parameter *k* in K-Means and FCM gets the number of classes. In CESynC algorithm, parameter *MinPts* is set as 2 for Iris and Cloud, parameter *MinPts* is set as 1 for the other six UCI data sets, and parameter σ is set according to Eq. (12).

(a) The setting of parameters in several clustering algorithms

UCI Data Sets	parameters (δ, σ) in CESynC	parameter δ in SynC and ESynC	parameter *k* in K-Means and FCM	parameter *Eps* in DBSCAN	parameter *h* in Mean Shift
Iris	(120, 10)	120	3	75	150
Wine	(290, 170)	305	3	242.725	305
Wdbc	(250, 200)	325	2	212	380
Glass	(120, 100)	148	6	80	170
Ionosphere	(600, 300)	615	2	350	1200
Letter-recognition	(180, 80)	210	26	160	220
Segmentation	(180, 100)	205	7	176	270
Cloud	(320, 150)	380	2	350	350

(b) Three robust information-theoretic measures (AMI / AVI / NMI) and the number of clusters obtained by using CESynC, SynC, ESynC, K-Means, FCM, AP, DBSCAN, and Mean Shift clustering algorithm

Measure indexes of algorithms	Name of algorithms	Data sets			
		Iris	Wine	Wdbc	Glass
AMI / AVI / NMI	CESynC	0.7143 / 0.7190 / 0.7265	0.6103 / 0.7294 / 0.7634	0.3650 / 0.4712 / 0.5134	0.3053 / 0.3479 / 0.4339
	SynC	0.0050 / 0.0100 / 0.4697	3.2528e-16 / 6.5056e-16 / 0.4578	6.8369e-16 / 1.3673e-15 / 0.3226	0.0012 / 0.0025 / **0.5306**
	ESynC	0.7143 / 0.7190 / 0.7265	0.6057 / 0.7259 / 0.7615	0.3277 / 0.4205 / 0.4717	0.2872 / 0.3432 / 0.4540
	K-Means	0.7107 / 0.7109 / 0.7145	**0.8735** / **0.8769** / **0.8782**	**0.6190** / **0.6314** / **0.6320**	**0.3265** / 0.3301 / 0.3588
	FCM	**0.7888** / **0.7893** / **0.7919**	0.3820 / 0.4669 / 0.4823	0.6006 / 0.6054 / 0.6060	0.2525 / **0.3630** / 0.4108
	AP	0.3982 / 0.5468 / 0.6061	0.2977 / 0.4427 / 0.5382	0.1482 / 0.2526 / 0.3711	0.2423 / 0.3236 / 0.4257
	DBSCAN	0.3859 / 0.5406 / 0.6540	0.1517 / 0.2113 / 0.4471	0.0404 / 0.0674 / 0.2449	0.1513 / 0.2227 / 0.4671
	Mean Shift	0.7143 /	0.5819 /	0.0165 /	0.2861 /

		0.7190 / 0.7265	0.7184 / 0.7612	0.0213 / 0.0796	0.3107 / 0.4153
The number of clusters	CESynC	**3** (+ 2 isolates)	**3** (+ 15 isolates)	5 (+ 29 isolates)	**6** (+ 16 isolates)
	SynC	2 (+ 145 isolates)	0 (+178 isolates)	0 (+ 569 isolates)	1 (+ 212 isolates)
	ESynC	**3** (+ 2 isolates)	**3** (+ 16 isolates)	3 (+ 44 isolates)	**6** (+ 29 isolates)
	K-Means	3	3	2	6
	FCM	3	2 (+ 1 null cluster)	2	2 (+ 4 null clusters)
	AP	11	21	36 (+ 9 isolates)	12 (+ 14 isolates)
	DBSCAN	**3** (+ 35 isolates)	**3** (+ 75 isolates)	**2** (+ 189 isolates)	**6** (+ 81 isolates)
	Mean Shift	**3** (+ 2 isolates)	4 (+ 17 isolates + 1 null cluster)	3 (+24 isolates + 1 null cluster)	**6** (+ 24 isolates)

Measure indexes of algorithms	Name of algorithms	Data sets			
		Ionosphere	Letter-recognition	Segmentation	Cloud
AMI / AVI / NMI	CESynC	0.1092 / 0.1745 / 0.3191	**0.4282** / -0.2557 / 0.4282	0.4669 / 0.5538 / 0.6442	**1.0000** / **1.0000** / **1.0000**
	SynC	3.5016e-04 / 7.0007e-04 / 0.3339	0.0166 / 0.0317 / **0.5768**	-1.6974e-15 / -3.3948e-15 / 0.6033	2.4432e-04 / 4.8852e-04 / 0.3016
	ESynC	0.1073 / 0.1701 / 0.3106	0.3971 / **15.4680** / 0.3971	0.4212 / 0.5093 / 0.6086	**1.0000** / **1.0000** / **1.0000**
	K-Means	0.1246 / 0.1280 / 0.1299	0.3484 / 0.3540 / 0.3572	**0.5286** / 0.5843 / 0.6103	0.9944 / 1.0056 / 0.9944
	FCM	0.1211 / 0.1245 / 0.1264	0.0042 / 0.0069 / 0.0095	0.2574 / 0.3831 / 0.4454	0.9944 / 1.0056 / 0.9944
	AP	0.1002 / 0.1688 / 0.2809	-	0.4897 / **0.6068** / **0.6781**	0.1653 / 0.2838 / 0.4107
	DBSCAN	0.0949 / 0.1663 / **0.3791**	0.1736 / 2.3798 / 0.1736	0.3295 / 0.4015 / 0.5568	**1.0000** / **1.0000** / **1.0000**
	Mean Shift	**0.1445** / **0.1768** / 0.2126	0.3649 / 6.1353 / 0.3649	0.5048 / 0.5747 / 0.6447	**1.0000** / **1.0000** / **1.0000**
The number of clusters	CESynC	**2** (+ 86 isolates)	30 (+ 48 isolates)	9 (+ 30 isolates)	**2**
	SynC	0 (+ 350 isolates)	845 (+ 17823 isolates)	0 (+ 210 isolates)	5 (+ 2038 isolates)
	ESynC	**2** (+ 83 isolates)	**26** (+ 10 isolates)	**7** (+ 31 isolates)	**2**
	K-Means	2	26	7	2
	FCM	2	2 (+ 24 null clusters)	2 (+ 5 null clusters)	2
	AP	14 (+ 44 isolates)	-	17 (+ 7 isolates)	66 (+ 1 isolate)
	DBSCAN	**2** (+ 145 isolates)	28 (+ 319 isolates)	**7** (+ 51 isolates)	**2**
	Mean Shift	**2** (+ 19 isolates)	**26** (+ 3 isolates + 1 null cluster)	**7** (+ 22 isolates)	**2**

Note1: In the Letter-recognition data set, DBSCAN algorithm obtains 21 clusters and 243 isolates when parameter $\delta = 160.0001$, so we set parameter $\delta = 160$ in DBSCAN. The sign '-' in the lines of AP algorithm means that there is no results because the time cost is too larger.

Note2: In Table 9, the largest values of AMI, AVI, and NMI in every data set are shown in bold.

5.5 Analysis and conclusions of experimental results

From the comparison results of these figures and tables (Fig. 2, sfig. 4, and Tables

4 - 9), we observe that CESynC algorithm has some superiority than ESynC algorithm and SynC algorithm. We also find that ICESynC algorithm is superior to CESynC algorithm in time cost because of the use of effective spatial index structures.

From the simulations of some artificial data sets (from data0 – data6, DS0 - DS16), we observe that the valid interval of parameter δ in CESynC algorithm is longer or has some improvements than that in ESynC algorithm, DBSCAN algorithm, or Mean Shift algorithm in some cases.

From some display figures and tables, we observe that CESynC algorithm can explore the correct clusters and isolates like DBSCAN algorithm in many cases. In many kinds of data sets, CESynC algorithm, ESynC algorithm, and DBSCAN algorithm can explore obvious clusters or isolates if selecting a proper value for parameter δ or *Eps*, and SynC algorithm cannot explore obvious clusters in many data sets.

From simulations of some data sets, we observe that the iterative times of SynC algorithm, AP algorithm, K-Means algorithm, and FCM algorithm is larger than that of CESynC algorithm and ESynC algorithm. In many data sets, CESynC algorithm, ESynC algorithm, Mean Shift algorithm, and DBSCAN algorithm have better ability than SynC algorithm, K-Means algorithm, FCM algorithm, and AP algorithm in exploring clusters and isolates. Specially, AP algorithm needs the longest time.

CESynC algorithm is an improved clustering algorithm with broader clustering adjustability than ESynC algorithm and SynC algorithm almost in many cases. Usually, parameter δ and parameter σ have a long valid interval in many data sets with obvious clusters. In simulations, we observe that if parameter δ and parameter σ get some different values in their valid interval, the clustering results of CESynC algorithm are the same except the time cost.

Because of the limited page space, we only select some typical data sets used in our experiments. For all experimental data sets, we observe that CESynC algorithm improves in clustering quality or gets the same clustering results as ESynC algorithm. For other data sets, CESynC algorithm is still superior to (or the same as) ESynC algorithm in clustering quality. We believe that the selection of experimental data sets is not biased.

6. Conclusions

This paper presents an improved synchronization clustering method, CESynC algorithm. For some data sets that ESynC algorithm and SynC algorithm cannot detect

correct clusters, CESynC algorithm can obtain correct clusters. From some simulated experiments of some artificial data sets, we observe that parameter δ in CESynC algorithm has better valid interval than ESynC algorithm and SynC algorithm in some cases. From the simulated experiments of nine artificial data sets, we observe that the valid interval of parameter σ is affected by parameters δ and *MinPts*. From the simulated experiments of eight UCI data sets, we observe that CESynC algorithm gets better (or the same) clustering results than (or as) that of ESynC algorithm. From many experiments, we observe that the clustering results of CESynC algorithm and ESynC algorithm are often better than that of SynC algorithm. So we can say CESynC algorithm can often obtain better clustering quality than ESynC algorithm and SynC algorithm in some kinds of data sets. Further comparison experiments with some classical clustering algorithms demonstrate the clustering effect of CESynC algorithm.

The major contributions of the paper can be summarized as follows:

(1) In order to conquer the shortcoming of ESynC algorithm that may regard a whole irregular cluster as some micro-clusters, it develops a combined clustering algorithm based on ESynC algorithm and a merging judgement process of micro-clusters.

(2) It presents two concrete merging strategies and two judgement methods of merging micro-clusters in CESynC algorithm.

(3) It validates the improved effect of ICESynC algorithm in time cost and that of CESynC algorithm in clustering quality by some simulated experiments.

CESynC algorithm is robust to outliers and can find obvious clusters with different shapes. The number of clusters does not have to be fixed before clustering. Usually, parameter δ has a long valid interval that can be determined by using an exploring method listed in Chen (2015), the heuristic method described by Theorem 1 and Property 1 presented in Chen (2017), or using the MDL-based method presented in Böhm et al. (2010). Parameter σ also has a long valid interval that can be explored by using Eq. (12).

This work opens some possibilities for further improvement and investigation. First, further improve ICESynC algorithm in time cost. For example, designing a similarity-preserving hashing function that needs $O(1)$ time complexity is valuable and difficult in the process of constructing δ near neighbor point set. Second, explore the relation between parameter δ and parameter σ and extend the applicability of CESynC algorithm in more complex data sets. Third, implement CESynC algorithm on a cluster

with a parallel programming model such as MapReduce framework.

Acknowledgments

This work was supported by Chongqing Cutting-edge and Applied Foundation Research Program of China (grant number: cstc2016jcyjA0521) and Chongqing Three Gorges University of China (grant number: 16PY08).

Compliance with Ethical Standards:

Funding: This study was funded by Chongqing Cutting-edge and Applied Foundation Research Program of China (grant number: cstc2016jcyjA0521) and Chongqing Three Gorges University of China (grant number: 16PY08).

Conflict of Interest: Author Xinquan Chen declares that he has no conflict of interest.

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

Informed consent: Informed consent was obtained from all individual participants included in the study.

References

Agrawal, R., Gehrke, J., & Gunopolos, D., et al. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD* (pp. 94-105).

Ankerst, M., Breunig, Markus M., Kriegel, Hans-Peter., & Sander, Jörg. (1999). OPTICS: Ordering points to identify the clustering structure. In *Proceedings of ACM SIGMOD* (pp. 49-60).

Bezdek, J. C. (1981). Pattern recognition with fuzzy objective function algorithms. New York, Plenum Press.

Böhm, C., Plant, C., & Shao, J., et al. (2010). Clustering by synchronization. In *Proceedings of ACM SIGKDD* (pp. 583-592).

Chehreghani, M. H. (2017). Clustering by shift. *In Proceedings of ICDM* (pp. 793-798).

Chen, X. (2013). Clustering based on a near neighbor graph and a grid cell graph.

Journal of Intelligent Information Systems, *40*(3), 529-554.

Chen, X. (2014). Synchronization clustering based on a linearized version of Vicsek model. arXiv: 1411.0189 [cs.LG]. http://arxiv.org/abs/1411.0189.

Chen, X. (2015). A new clustering algorithm based on near neighbor influence. *Expert Systems with Applications*, 42(21), 7746-7758.

Chen, X. (2017). An effective synchronization clustering algorithm. *Applied Intelligence*, 46(1), 135 - 157.

Chen, X. (2018). Fast synchronization clustering algorithms based on spatial index structures. *Expert Systems with Applications*, 94, 276 - 290.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603-619.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communication of the ACM*, *51*(1), 107-113.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial data sets with noise. In *Proceedings of ACM SIGKDD* (pp. 226-231).

Frank, A., & Asuncion, A. (2010). UCI Machine Learning Repository Irvine, University of California.

Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, *315*(16), 972-976.

Fukunaga, K., & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, *21*(1), 32-40.

Grunwald, P. (2005). A tutorial introduction to the minimum description length principle. Cambridge, MIT Press.

Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An efficient clustering algorithm for clustering large databases. In *Proceedings of ACM SIGMOD* (pp. 73-84).

Hang, W., Choi, K., & Wang, S. (2017). Synchronization clustering based on central force optimization and its extension for large-scale datasets. *Knowledge-Based Systems*, 118, 31-44.

He, X., Gumbsch, T., Roqueiro, D., Borgwardt, K. (2017). Kernel conditional clustering. In *Proceedings of IEEE ICDM* (pp. 157-166).

Horn, D., & Gottlieb, A. (2002). Algorithm for data clustering in pattern recognition

problems based on quantum mechanics. *Physical Review Letters*, 88(1), 018702.

Huang, J. B., Kang, J. M., Qi, J. J., & Sun, H. L. (2013). A hierarchical clustering method based on a dynamic synchronization model. *Science in China Series F: Information Sciences*, *43*(5), 599-610.

Jadbabaie, A., Lin, J., & Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, *48*(6), 998-1001.

Karypis, G., Han, E. H., & Kumar, V. (1999). CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, *32*(8), 68-75.

Kaufman, L., & Rousseeuw, P. J. (1990). Finding groups in data: an introduction to cluster analysis. John Wiley & Sons.

Li, T., Zhang, Y., Li, D., Liu, X., and Peng, Y. (2017). Fast compressive spectral clustering. In *Proceedings of IEEE ICDM* (pp. 949-954).

Liu, A., Su, Y., Nie, W., & Kankanhalli, M. (2017). Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1), 102-114.

Luxburg, U. V. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395-416.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *MSP* (pp. 281-297).

Qin, J., Ma, Q., Gao, H., Shi, Y., & Kang, Y. (2017). On group synchronization for interacting clusters of heterogeneous Systems. *IEEE Transactions on Cybernetics*, *47*(12), 4122-4133.

Rodriguez, A. & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, 344(6191): 1492 - 1496.

Roy, S. & Bhattacharyya, D. K. (2005). An approach to find embedded clusters using density based techniques. *Lecture Notes in Computer Science*, 3816, 523-535.

Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299-1319.

Shao, J., He, X., Plant, C., Yang, Q., & Böhm, C. (2013a). Robust synchronization-based graph clustering. In *Proceedings of PAKDD* (PP. 249-260).

Shao, J., He, X., Böhm, C., Yang, Q., & Plant, C. (2013b). Synchronization inspired partitioning and hierarchical clustering. *IEEE Transactions on Knowledge and Data Engineering*, *25*(4), 893-905.

Shao, J., Yang, Q., Dang, H.-V., Schmidt, B., and Kramer, S. (2016). Scalable clustering by iterative partitioning and point attractor representation. *ACM Transactions on Knowledge Discovery from Data*, 11(1), 5.

Shao, J., Gao, C., Zeng, W., Song, J., & Yang, Q. (2017a). Synchronization-inspired co-clustering and its application to gene expression data. *In Proceedings of ICDM* (pp. 1075-1080).

Shao, J., Wang, X., Yang, Q., Plant, C., & Böhm, C. (2017b). Synchronization-based scalable subspace clustering of high-dimensional data. *Knowledge and Information Systems*, 52(1), 83-111.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3, 583-617.

Tan, P. N., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining*. Addison Wesley.

Theodoridis, S., & Koutroumbas, K. (2006). *Pattern recognition* (3rd edition). Academic Press.

Vicsek, T., Czirok, A., & Ben-Jacob, E., et al. (1995). Novel type of phase transitions in a system of self-driven particles. *Physics Review Letter*, *75*(6), 1226-1229.

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11, 2837-2854.

Wang, L., & Liu, Z. (2009). Robust consensus of multi-agent systems with noise. *Science in China Series F: Information Sciences*, *52*(5), 824-834.

Wang, W., Yang, J., & Muntz, R. (1997). STING: A statistical information grid approach to spatial data mining. In *Proceedings of VLDB* (pp. 186-195).

White, D. A., & Jain, R. (1996). Similarity indexing with the SS-tree. In *Proceedings of IEEE ICDE* (pp. 516-523).

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20(1), 68-86.

Zhang, T., Ramakrishnan R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD* (pp. 103-114).