



# An effective multi-level synchronization clustering method based on a linear weighted Vicsek model

Xinquan Chen<sup>1</sup> · Yirou Qiu<sup>2</sup>

Published online: 10 July 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

## Abstract

To conquer the shortcoming that general clustering methods cannot process big data in the main memory, this paper presents an effective multi-level synchronization clustering (MLSynC) method by using a framework of “divide and collect” and a linear weighted Vicsek model. We also introduce two concrete implementations of MLSynC method, a two-level framework algorithm and a recursive algorithm. MLSynC method has a different process with SynC algorithm, ESynC algorithm and SSynC algorithm. By the theoretic analysis, we find the time complexity of MLSynC method is less than SSynC. Simulation and experimental study on multi-kinds of data sets validate that MLSynC method not only gets better local synchronization effect but also needs less iterative times and time cost than SynC algorithm. Moreover, we observe that MLSynC method not only needs less time cost than ESynC and SSynC, but also almost gets the same local synchronization effect as ESynC and SSynC if the partition of the data set is proper. Further comparison experiments with some classical clustering algorithms demonstrate the clustering effect of MLSynC method.

**Keywords** Divide and collect · Kuramoto model · Shrinking synchronization clustering · Linear weighted Vicsek model · Near neighbor point set

## 1 Introduction

Clustering is an unsupervised learning method that tries to find some obvious distribution structures and patterns in unlabeled data sets by maximizing the similarity of the objects in a common cluster and minimizing the similarity of the objects in different clusters [1]. Clustering has been used in many areas such as machine learning, pattern recognition, image processing, marketing and customer analysis, agriculture, security and crime detection, information retrieval, and

bioinformatics. Cluster is often one important step in the process of data analysis.

Clustering algorithms have been studied for decades. There have been hundreds of clustering algorithms until now, but none of them is all-purpose. Almost all clustering algorithms have flaws. Some clustering algorithms are suitable for dealing with data with certain types, and others are suitable for handling data with special distribution structures. Many real data have complex distributions, diversiform types, great capacity, noises, or isolates. So there is a continuous demand for researching different kinds of clustering methods. To obtain better clustering results in real-world applications where the amount of data is often very large and the types of data are diversiform, researchers try their best to develop new efficient and effective clustering algorithms.

The traditional clustering algorithms are usually classified into partitioning methods [2, 3], hierarchical methods [4–7], density-based methods [8, 9], grid-based methods [10, 11], model-based methods [12] and graph-based methods [12–14]. Recent clustering methods have quantum clustering algorithms [15], spectral clustering algorithms [16, 17], kernel clustering algorithms [18], affinity propagation clustering algorithms [19], synchronization clustering algorithms [20–27], and so on. Partitioning methods are divided into hard

---

**Electronic supplementary material** The online version of this article (<https://doi.org/10.1007/s10489-020-01767-4>) contains supplementary material, which is available to authorized users.

---

✉ Xinquan Chen  
chenxqscut@126.com

Yirou Qiu  
qyr3366@163.com

<sup>1</sup> School of Computer & Information, Anhui Polytechnic University, Wuhu 241000, China

<sup>2</sup> School of Optics and Photonics, Beijing Institute of Technology, Beijing 100081, China

clustering and fuzzy clustering. Fuzzy clustering can cope with overlapping or cluster boundaries. Partitioning methods are sensitive to the predefined number of clusters, initial prototypes, noise and outliers. They have linear time complexity with the number of data points. K-Means [2] and FCM [3] are two famous and often used partition-based clustering algorithms.

Recently, several original clustering algorithms, such as Affinity Propagation (AP) algorithm [19], Synchronization Clustering (SynC) algorithm [20] and clustering by fast search and find of Density Peaks (DP) algorithm [28], were published. AP algorithm is a new type of clustering algorithm published in *Science*. After AP algorithm was published, clustering based on probability graph models grew a new research direction. As we know, SynC algorithm is the first synchronization clustering algorithm. After SynC algorithm was presented, synchronization clustering attracts some researchers. Some synchronization clustering methods [21–27] were published from different views. DP algorithm is a clustering algorithm 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 obtained automatically, outliers can be identified easily, and even nonspherical clusters can be explored quickly. So we think DP algorithm can lead a new research direction in clustering field.

Synchronization clustering is a kind of novel clustering approach. The original synchronization clustering algorithm (named as SynC) says that it can find the intrinsic structure of the data set without any distribution assumptions and handle outliers by dynamic synchronization [20]. This paper is inspired by several papers [20, 24, 25, 29–31] and the strategy of “divide and conquer”.

In the circumstance of big data, general clustering algorithms cannot process a large amount of data in the main memory at a time. To conquer this problem, basing on SynC algorithm [20], ESynC algorithm [25] and SSynC algorithm [24], this paper researches a Multi-Level Synchronization Clustering (MLSynC) method based on SSynC algorithm by using a framework of “divide and collect” and a linear weighted Vicsek model. MLSynC method has a different process with SynC algorithm, ESynC algorithm and SSynC algorithm. SynC algorithm [20] is based on an extensive Kuramoto model, ESynC algorithm [25] is based on a linear version of Vicsek model, SSynC algorithm [24] is based on a linear weighted Vicsek model and MLSynC method uses the strategy of “divide and conquer” and a linear weighted Vicsek model for clustering. Because the linear weighted Vicsek model has a nicer superposition characteristic for clustering, MLSynC method can be used to process big data effectively and efficiently.

The idea of “divide and collect” in MLSynC method is similar to the MapReduce framework [32] in some aspects,

although it is developed independently. MapReduce is a parallel and distributed programming model that is used to process very large data sets on a cluster. In the MapReduce framework, the Map and Reduce operations affect the clustering result very much, so they cannot be directly extended to clustering field. MLSynC method can be used for clustering on a cluster with a parallel programming model or on a personal computer with a serial programming method. If the partition of the data set is proper, MLSynC method is both efficient and effective.

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

The symbols used in this paper are summarized in Table 1.

## 2 Related works

### 2.1 Synchronization clustering methods

In 2010, Böhm et al. [20] presented a novel clustering approach, SynC algorithm, inspired by the synchronization principle. SynC algorithm can find the intrinsic structure of the data set without any distribution assumptions and handle outliers by dynamic synchronization. To implement automatic clustering, those natural clusters can be discovered by using the Minimum Description Length (MDL) principle [33].

After SynC algorithm was presented, some researchers published several synchronization clustering papers [21–27]. To find subspace clusters of some high-dimensional sparse data sets, a novel effective and efficient subspace clustering algorithm, ORSC [21], was proposed. To find the intricate patterns of a complex graph, a novel and robust graph clustering algorithm, RSGC [22], was proposed by regarding the graph clustering as a dynamic process towards synchronization. To explore meaningful levels of the hierarchical cluster structure, a novel dynamic hierarchical clustering algorithm, hSync [23], was presented based on synchronization and the MDL principle. Inspired by the work of [20] and Vicsek model [29], Chen [24] presented a Shrinking Synchronization Clustering (SSynC) algorithm by using a linear weighted Vicsek model. Inspired by the work of [20], Chen [25] proposed an Effective Synchronization Clustering (ESynC) algorithm based on a linear version of Vicsek model. Simulations validate that the linear version of Vicsek model is an effective synchronization model for clustering. Based on the metaphor of gravitational kinematics and the central force optimization method, Hang et al. [27] 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. Chen [26] presented a Fast Synchronization Clustering (FSynC) method basing on the

work of [20] and spatial index structure. FSynC algorithm, which is a parametric algorithm, is an improved version of SynC algorithm by combining a multidimensional grid partitioning method and a Red-Black tree structure to construct the near neighbor point sets of all points [26].

## 2.2 Competitive learning for clustering

Competitive learning is a type of unsupervised learning model. In simple competitive networks based on the winner-take-all activation rule, the neuron with the highest activity in response to the input wins the competition, inhibiting the other neurons to quiescence. In self-organization maps, winner and its neighborhood neurons may learn to update their weights. After the competition, the input space is separated into multiple subspaces and each neuron will output maximum value for those inputs in its responding subspace.

The famous competitive learning models have Von der Malsburg's model [34], Kohonen's self-organization map (SOM) [35, 36] and the adaptive resonance theory (ART) based on the biological motivation [37, 38]. SOM uses a set of heuristic procedures and does not use any minimization objective function. It can often preserve clustering topology well. But it is not suitable for non-vectorial data and has some shortcomings, such as forced termination, unguaranteed convergence, nonoptimized procedure and the sensitivity to the

sequence of data. ART family includes a series of unsupervised learning models [39], such as ART1, ART2, ART3, ARTMAP models and fuzzy ARTMAP. The distinctive characteristics of ART are code representation, long-term memory and corresponding geometric interpretation [40]. In supervised learning, semi-supervised learning, self-supervised and other machine learning fields, ART-based models are also researched. For example, the self-supervised ARTMAP [41] can deserve partial knowledge previously acquired from labeled data and further learn new features from unlabeled data. This model can integrate multiple knowledge from a teacher, the environment and internal model activation. Recently, Seiffert [42] extends the core ART theory and algorithms to some mixed input domains by establishing a dynamic equation model of ART in the time scales calculus.

The function of the concept root core in SSynC algorithm is similar to the concept prototype of K-Means algorithm or the concept neuron of competitive networks. The root cores after the synchronization iteration in SSynC algorithm can often be regarded as the prototypes of the clusters of the original data. In our proposed model of this paper, the location of the root core that represents some cores is regarded as their cluster center, and the location of the root core that represents only one or several cores is regarded as the final synchronization location of one or several isolates. Each neuron of competitive networks can represent one data subspace.

**Table 1** Symbols used in this paper

Symbol	Definition
$D = S_1 \cup \dots \cup S_m$	The data set $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ can be divided into $m$ data subsets $\{S_1, \dots, S_m\}$
$n$	The number of points in the original data set $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$
$\delta_1, \dots, \delta_m, \delta_{\text{merge}}$	$m + 1$ range parameters used in of Algorithm 1. They are used to compute near neighbor point set in the $m + 1$ clustering modules based on SSynC algorithm
$\varepsilon$	The deviation parameter used in SSynC algorithm
$C_1, \dots, C_m, C$	$m + 1$ core sets used in Algorithm 1
$C_{\text{Init}} = \{c_1, \dots, c_n\}$	The initial core set created from the data set $D$ in Algorithm 2
$C_{\text{Res}}$	The result core set represents the clusters and isolates of the data set $D$ in Algorithm 2
$T$	The times of synchronization iteration in SSynC algorithm
$d$	The data dimension
$n_{(t)}$	The number of active cores in the $t$ -step synchronization evolution
$n_i(t=0)$	The initial number of active cores in the $i$ -th clustering module
$n_{\text{ave}}(t=0)$	The initial average number of active cores in the first-level $m$ clustering modules
$T_{\text{max}}$	The max synchronization times in the first-level $m$ clustering modules
$ C $	The number of elements in the core set $C$
$ C _{(t)}$	The number of active cores in the $t$ -step synchronization evolution
$K_i$	The number of clusters or isolates in the data subset $S_i$ ( $i = 1, \dots, m$ )
$K$	The (predefined or final) number of clusters or isolates in the data set $D$
$\text{dist}(\cdot, \cdot)$	The dissimilarity measure in the Euclidean metric space
$\mathbf{x} = (x_1, \dots, x_d)$	The vector of point $\mathbf{x}$ in $d$ -dimensional Euclidean space
$\mathbf{c} = (c_1, \dots, c_d)$	The location vector of core $\mathbf{c}$ in $d$ -dimensional Euclidean space
$\text{MinPts}$	The density threshold parameter used in DBSCAN algorithm

Learning vector quantization (LVQ) and SOM can represent all original data using some cluster centers. The weight vectors of the SOM are often converged to the mean of input vectors. ESynC algorithm [25] uses a linear version of Vicsek model, and SSynC algorithm [24] uses a linear weighted Vicsek model. Center or mean of the input vectors from one data subspace is the common ground among SOM, ESynC algorithm and SSynC algorithm.

### 2.3 The scalability of the clustering algorithm

Dealing with data sets with a very large sample size and (or) a very high number of dimensions has always been an important challenge in clustering field because of the time-space complexity and clustering accuracy. A famous scalable version of K-Means, scaleKM, was presented by Bradley et al. [43]. Although scaleKM is much faster than K-Means for big data sets, it still requires a predefined input parameter, the correct number of clusters, which is unknown for unlabeled data. Another scalable version of K-Means for large data sets, Mini-batch K-Means (MBKM), was proposed by Sculley [44]. MBKM uses the same objective function and iteration process as K-Means, but it improves the clustering speed by selecting small random batches from the original data set.

Dimension reduction is widely used in high dimensional data sets. A simple and efficient dimension reduction method is random projection [45]. This kind of single random projection is often unstable. And ensemble-based techniques [46] was developed by combining the result of multiple lower-dimensional clustering results.

To explore those continuous data areas from massively large high dimensional data sets, an orthogonal partitioning clustering method, O-Cluster [47], was presented by combining an active random sampling strategy with an axis-parallel partitioning technique. O-Cluster can get high-quality clustering results with excellent scalability and robustness, although it needs a large buffer size and a sensitive parameter. Recently, by using fast data-space reduction and an intelligent sampling strategy, Rathore et al. [48] presented a rapid hybrid, ensemble-based clustering algorithm named as FensiVAT. FensiVAT can enhance the scalability ability of both dimensions and sample size.

## 3 An effective multi-level synchronization clustering method based on a framework of “divide and collect” and SSynC algorithm

Facing big data, general clustering methods cannot process all data in the main memory at a time. To conquer this problem, we present an effective Multi-Level Synchronization Clustering (MLSynC) method by using a framework of “divide and collect” and a linear weighted Vicsek model. The

framework of “divide and collect” is an application of the strategy of “divide and conquer” in clustering field. The linear weighted Vicsek model described by Eq.(s5) of Sdefinition 5 of Supplementary Material is used in SSynC algorithm.

Although we use the Euclidean metric as our dissimilarity measure in this paper, this method 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 method can still be used.

### 3.1 The application condition of MLSynC method

Sfig. 1 of Supplementary Material presents a figure that compares the synchronization clustering results of MLSynC method using two different partitioning methods, a random partitioning method and a direct partitioning method. From Sfig. 1, we observe that if the spatial distributions of two partitioned data subsets vary very large and the clustering structure of the original data set is dissevered by the partitioning, MLSynC method will get a different clustering result with SSynC algorithm. If the spatial distributions of two partitioned data subsets have very small deviation, or the partitioning is uniform, MLSynC method will get a similar clustering result with SSynC algorithm.

### 3.2 The description of a two-level framework algorithm of MLSynC method

MLSynC method has a different clustering process with SynC algorithm [20], ESynC algorithm [25] and SSynC algorithm [24]. Figure 1 presents a two-level framework algorithm of MLSynC method. Sfig. 2 of Supplementary Material presents a three-level framework algorithm of MLSynC method. In the two-level framework algorithm of MLSynC method, the original data set that is usually large and cannot be processed in the main memory at a time is partitioned into  $m$  subsets. Each subset is processed by a clustering model (or a clustering machine) based on SSynC algorithm. After collected all root cores from the  $m$  clustering models, a clustering model based on SSynC algorithm is used again.

In Fig. 1, if  $m$  is too large, then the two-level framework algorithm of MLSynC method should be replaced by a three-level (or four-level and above) framework algorithm. A three-level framework algorithm of MLSynC method is presented in Online Resource 3 of Supplementary Material of this paper.

From Fig. 1, we observe that MLSynC method has a natural framework based on SSynC algorithm by integrating the clustering results of all subsets. MLSynC method also has nicer incremental clustering ability. For example, in Fig. 1, if every subset in  $\{S_2, \dots, S_m\}$  only has one point, then the two-level framework algorithm becomes an incremental clustering algorithm.

Here, we present the description of the two-level framework algorithm of MLSynC method.

---

**Algorithm 1:** The two-level framework algorithm of MLSynC method

---

**Input:** Data set  $D = \{x_1, \dots, x_n\}$ , dissimilarity measure  $\text{dist}(\cdot, \cdot)$ , parameter  $m$ , range parameters:  $\{\delta_1, \dots, \delta_m, \delta_{\text{merge}}\}$  and parameter  $\varepsilon$ ;

**Output:** The final clustering result of the data set  $D$ ;

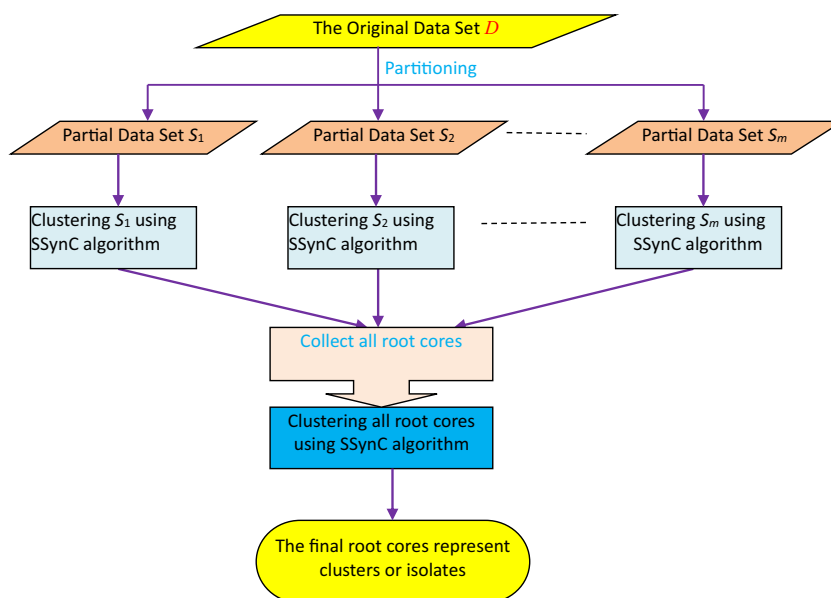
**Procedure:**

- 1: Partition the data set  $D = \{x_1, \dots, x_n\}$  that is read from a file or a data base into  $m$  sections,  $\{S_1, \dots, S_m\}$ . Usually,  $D = S_1 \cup \dots \cup S_m$  should be satisfied.
- /\* Clustering each subset  $S_i$  ( $i = 1, \dots, m$ ) from  $\{S_1, \dots, S_m\}$  using SSynC algorithm. \*/
- 2: **for**  $i = 1, \dots, m$  **do**
- 3:     Create an initial core set  $C_i$  from the subset  $S_i$ , according to Eq.(s4) of Sdefinition 4 of Supplementary Material;
- 4:     Clustering each core set  $C_i$  using SSynC algorithm. The function format of this step is:  
         $\text{NumOfClus}[i] \leftarrow \text{SSynC}(\text{CoreSet } C_i, \text{float } \delta_i, \text{float } \varepsilon)$ ; /\* Each element in array  $\text{NumOfClus}[1..m]$  is used to record the number of clusters of each subset. The initial value of the core set  $C_i$  is the input of SSynC algorithm. After finish the clustering process, core set  $C_i$  records the clustering result of subset  $S_i$ . Here,  $\text{NumOfClus}[i]$  is used to record the return value of SSynC algorithm. \*/
- 5: **end for**
- 6: Create a new core set  $C$  by collecting all root cores from the above  $m$  clustering results of  $m$  data subsets  $\{S_1, \dots, S_m\}$ .
- 7: Clustering the collected core set  $C$  using SSynC algorithm. The function format of this step is:  
         $\text{int FinNumOfClus} \leftarrow \text{SSynC}(\text{CoreSet } C, \text{float } \delta_{\text{merge}}, \text{float } \varepsilon)$ ;  
        /\*  $\text{FinNumOfClus}$  is used to record the final number of clusters of  $C$ . \*/  
        /\* After the collected core set  $C$  is operated by SSynC algorithm, the paths of some inactive cores are compressed just like the joint-set method such that the largest height of leaf cores is less than or equal to two (note: the height of root cores is one). \*/
- 8: The final root cores of the core set  $C$  operated by SSynC algorithm represent the clusters or isolates of the original data set  $D = \{x_1, \dots, x_n\}$ . Suppose the final core set  $C$  has  $K$  root cores, we think the data set  $D$  has  $K$  clusters or isolates. The location of the root core that represents some cores is regarded as their cluster center, and the location of the root core that represents only one or several cores is regarded as the final synchronization location of one or several isolates.

---

In Algorithm 1, steps 1 - 5 belong to the “divide stage”, and steps 6 and 7 belong to the “collect stage”.

**Fig. 1** A two-level framework algorithm of MLSynC method



### 3.3 The recursive algorithm of MLSynC method

Here, we present the description of the recursive algorithm of MLSynC method.

---

**Algorithm 2:** The recursive algorithm of MLSynC method

---

**Input:** Data set  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , dissimilarity measure  $\text{dist}(\cdot, \cdot)$ , parameter  $m$ , a setting set of parameter  $\delta: \{\delta_1, \dots, \delta_m, \delta_{\text{merge}}\}$  and parameter  $\varepsilon$ ;

**Output:** The final clustering result of the data set  $D$ ;

**Procedure:**

1: Create an initial core set  $C_{\text{Init}} = \{c_1, \dots, c_n\}$  from the data set  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  according to Eq.(s4) of Sdefinition 4 of Supplementary Material.

2: Call the dichotomy recursive function,  $\text{MLSynCRecur}(C_{\text{Init}}, 1, n, C_{\text{Res}})$ , in this step. The concrete implementation of the recursive function is:

*/\* MLSynCRecur is a dichotomy recursive function that can operate some huge data sets that are stored in in-memory or disk.  $C_{\text{Init}}$  is the core set used as the input of this algorithm, which represents the input data that may be loaded into in-memory step by step from disk.  $First$  is the label or index of the first record in  $C_{\text{Init}}$ , and  $Last$  is the label or index of the last record in  $C_{\text{Init}}$ .  $C_{\text{Res}}$  that represents clusters and isolates will be used to store the clustering results of  $C_{\text{Init}}$ . The return value of this function records the number of clusters and isolates. \*/*

2.1:  $\text{int MLSynCRecur}(\text{InputData } C_{\text{Init}}, \text{int } First, \text{int } Last, \text{OutputData } C_{\text{Res}})$

2.2: {

2.3:     **if**  $(Last - First) > \text{FitNum}$  **then**     */\* Parameter  $\text{FitNum}$  is a predefined threshold or a maximum that the computer system can operate the loaded data in its in-memory directly. \*/*

2.4:          $\text{int MidLoc} \leftarrow \text{DivideInitCS}(C_{\text{Init}}, First, Last)$ ; */\* The function,  $\text{DivideInitCS}$ , will partition  $C_{\text{Init}}$  into two parts. The return value of the function records the middle location of  $C_{\text{Init}}$ . \*/*

2.5:          $\text{int NumCS}_1 \leftarrow \text{MLSynCRecur}(C_{\text{Init}}, First, MidLoc, \text{OutputCS}_1)$ ;

2.6:          $\text{int NumCS}_2 \leftarrow \text{MLSynCRecur}(C_{\text{Init}}, MidLoc + 1, Last, \text{OutputCS}_2)$ ;

2.7:          $\text{InputData NewInputCS} \leftarrow \text{OutputCS}_1 \cup \text{OutputCS}_2$ ; */\*  $\text{NewInputCS}$  can be obtained by connecting  $\text{OutputCS}_1$  and  $\text{OutputCS}_2$  directly. \*/*

2.8:          $\text{int ResNumCS} \leftarrow \text{SSynC}(\text{NewInputCS}, \text{NumCS}_1 + \text{NumCS}_2, C_{\text{Res}})$ ; */\*  $\text{SSynC}$  algorithm is used to clustering  $\text{NewInputCS}$ .  $\text{CS}_{\text{Res}}$  that represents clusters and isolates will store the clustering results of  $\text{NewInputCS}$ . The return value of the function records the number of clusters and isolates. \*/*

2.9:     **else**

2.10:          $\text{int ResNumCS} \leftarrow \text{SSynC}(C_{\text{Init}}, Last - First, C_{\text{Res}})$ ; */\*  $C_{\text{Init}}$  is operated by  $\text{SSynC}$  algorithm directly. \*/*

2.11:     **end if**

2.12:     **return**  $\text{ResNumCS}$ ;

2.13: }

3: The parameter in the function  $\text{MLSynCRecur}$  of step 2,  $C_{\text{Res}}$ , represents the clusters or isolates of the data set  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ .

---

### 3.4 Time and space complexity analysis of MLSynC method

From the time complexity analysis of  $\text{SSynC}$  algorithm that presented in another paper [24], we know that  $\text{SSynC}$  algorithm needs  $\text{Time} = O\left(d \cdot \left(n_{(t=0)}^2 + \dots + n_{(t=T-1)}^2\right)\right) < O(Tdn^2)$ , which is usually less than  $\text{SynC}$  algorithm and  $\text{ESynC}$  algorithm. Here  $T$  is the times of synchronization in  $\text{SSynC}$  algorithm,  $d$  is the data dimension,  $n$  is the number of points in the original data set  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , and  $n_{(t)}$  is the number of active cores in the  $t$ -step synchronization evolution.

#### 3.4.1 Time and space complexity analysis of the two-level framework algorithm of MLSynC method

In the two-level framework algorithm of  $\text{MLSynC}$  method, step 1 needs  $\text{Time} = O(n)$  and  $\text{Space} = O(n)$ .

The time cost of step 2 is:

$$\begin{aligned} \text{Time} &= O\left(d \cdot \sum_{i=1}^m (n_i^2(t=0) + \dots + n_i^2(t=T_i-1))\right) \\ &\approx O\left(d \cdot \frac{n_{\text{ave}}^2(t=0) + \dots + n_{\text{ave}}^2(t=T_{\text{max}}-1)}{m}\right) \end{aligned} \quad (1)$$

In Eq. (1),  $T_i$  is the synchronization times in the  $i$ -th clustering module based on SSynC algorithm,  $n_i(t=0)$  is the initial number of active cores in the  $i$ -th clustering module,  $n_{ave}(t=0)$  is the initial average number of active cores in the  $m$  clustering modules, and  $T_{max}$  is the max synchronization times in the  $m$  clustering modules.

Suppose subset  $S_i$  ( $i = 1, \dots, m$ ) has  $K_i$  clusters or isolates, then step 3 needs  $\text{Time} = O(K_1 + \dots + K_m) = O(|C|)$ . Here  $|C|$  is the number of elements in the core set  $C$ .

Step 4 needs  $\text{Time} = O\left(d \cdot \left(|C|_{(t=0)}^2 + \dots + |C|_{(t=T-1)}^2\right)\right)$ . Here  $T$  is the synchronization times using SSynC algorithm in step 4 and  $|C|_{(t)}$  is the number of active cores in the  $t$ -step synchronization evolution.

Step 5 needs  $\text{Time} = O(n)$  and  $\text{Space} = O(n)$ .

### 3.4.2 Time and space complexity analysis of the recursive algorithm of MLSynC method

In the recursive algorithm of MLSynC method, step 1 needs  $\text{Time} = O(n)$  and  $\text{Space} = O(n)$ .

In many cases, the time cost of step 2 is:

$$\text{Time}(n) = \begin{cases} O\left(d \cdot \left(n_{(t=0)}^2 + \dots + n_{(t=T-1)}^2\right)\right), & \text{if } n \leq \text{FitNum}, \\ O(1) + 2 \cdot \text{Time}\left(\frac{n}{2}\right), & \text{if } n > \text{FitNum}. \end{cases} \quad (2)$$

Step 3 needs  $\text{Time} = O(n)$  and  $\text{Space} = O(n)$ .

According to our analysis, MLSynC method often needs less time than SSynC algorithm.

## 3.5 The setting of parameters $\delta$ , $\varepsilon$ and $m$ in MLSynC method

For each subset or collected core set in MLSynC method, SSynC algorithm is used for clustering with proper values of parameters  $\delta$  and  $\varepsilon$ .

### 3.5.1 The setting of parameter $\delta$ in MLSynC method

Parameter  $\delta$  can affect the clustering result of MLSynC method. There are several kinds of methods to select a proper value for parameter  $\delta$ . The first one [20] is that parameter  $\delta$  is optimized by the MDL principle. In Chen [49], two other methods were presented to estimate parameter  $\delta$ . The third one is the heuristic selection of parameter  $\delta$  in Chen [25]. The fourth one is a linear-searching exploring method of parameter  $\delta$  that is described in Chen [26]. Here, we can also select a proper value for parameter  $\delta$  according to Stheorem 1 and Sproperty 1 of Supplementary Material.

### 3.5.2 The setting of parameter $\varepsilon$ in MLSynC method

Parameter  $\varepsilon$  affects the time cost of MLSynC method slightly. Usually, parameter  $\varepsilon$  has a long valid interval. For example, if parameter  $\delta > 15$ , then the valid interval of parameter  $\varepsilon$  is about in  $(0, 10]$ . In simulations, we almost get the same results for several different values (such as 0.00001, 0.0001, 0.001, 0.01, 0.1, 1 and 10) of parameter  $\varepsilon$ .

### 3.5.3 The setting of parameter $m$ in MLSynC method

In the two-level framework algorithm of MLSynC method, parameter  $m$  affects its whole time cost. If parameter  $m$  is set too large, then the number of points in each part of the data set becomes less. So SSynC algorithm in the two-level framework algorithm of MLSynC method needs less clustering time for each subset, and the two-level framework algorithm of MLSynC method needs much divide and collect time. Usually, there is a balance between the increase of “divide and collect” time and the decrease of clustering time with the increase of parameter  $m$ . So we think that parameter  $m$  cannot be set too large.

## 3.6 The improvement of MLSynC method

In MLSynC method, one improved version of SSynC algorithm can be obtained by combining a multidimensional grid partitioning method and a Red-Black tree structure to construct the near neighbor point sets of all active cores. The improving method that can decrease its time cost is introduced in Chen [26]. Generally, we first partition the data space of the data set  $D = \{x_1, \dots, x_n\}$  by using a kind of multidimensional grid partitioning method. Then design an effective index of all grid cells and construct  $\delta$  near neighbor grid cell set for each grid cell. If every grid cell uses a Red-Black tree to index its active cores in each synchronization step, then constructing the  $\delta$  near neighbor point set for every active core will become quicker when the number of grid cells is proper.

Before synchronization iterative evolution, if we set a proper value for parameter  $\delta$  to filtrate isolates, then these isolates can be set as inactive cores that will not be operated in the next iterative evolution. This improvement of the implementation technique is often effective in some data sets.

## 3.7 The convergence of MLSynC method

In MLSynC method, SSynC algorithm is used to clustering each subset and the collected core set. So the convergence of MLSynC method is completely depended on the convergence of SSynC algorithm. According to the convergence analysis

of SSynC algorithm and our simulations, we know that MLSynC method is also convergent.

#### 4 The comparison of the dynamic clustering processes among SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method

SynC algorithm uses the extensive Kuramoto model described by Eq.(s2) of Sdefinition 2 of Supplementary Material at each step evolution that is a nonlinear renewal model. ESynC algorithm uses the linear version of Vicsek model described by Eq.(s3) of Sdefinition 3 at each step evolution that is a linear renewal model. SSynC algorithm uses the linear weighted Vicsek model described by Eq.(s5) of Sdefinition 5 at each step evolution that is a linear weighted renewal model. MLSynC method uses a framework of “divide and collect” and the linear weighted Vicsek model.

Figure 2 compares the tracks of 2000 data points from DS0 among the clustering processes of SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method. From Fig. 2, we observe that MLSynC method, ESynC algorithm and SSynC algorithm have better local synchronization effect than SynC algorithm.

Figure 3a compares a measure index of clustering result, the cluster order parameter with  $t$ -step evolution ( $t$ : 0 - 49) [20], among SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method. Figure 3b compares another measure index of clustering result, the  $t$ -step average length of edges ( $t$ : 0 - 49) [25], among SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method. And Fig. 3c compares the relationship between the final number of clusters and parameter  $\delta$  after finished clustering using the four algorithms respectively.

From Fig. 3a and b, we observe that the  $t$ -step average length of edges is better than the cluster order parameter with  $t$ -step evolution in measuring the final synchronization results. From Fig. 3c, we observed that parameter  $\delta$  has a long valid interval in ESynC algorithm, SSynC algorithm and MLSynC method. From Fig. 3c, we also observe that the smaller parameter  $\delta$  is set in SynC, ESynC, SSynC and MLSynC, the larger the final number of clusters is. In many data sets with obvious clusters, if we use a proper partitioning method to group the data set, MLSynC method can often get the correct number of clusters when parameter  $\delta$  chooses any value from its valid interval. ESynC algorithm and SSynC algorithm can often get the correct number of clusters when parameter  $\delta$  chooses any value from its valid interval. But the final number of clusters using SynC algorithm is often much larger than the actual number of clusters whenever parameter  $\delta$  gets any value in a long interval.

## 5 Simulated experiments

### 5.1 Experimental design

Our experiments are finished on a personal computer. Experimental programs are developed using C / Python / Matlab language under Windows 7.

To verify the improvement in the clustering effect and time cost of MLSynC method, there will be some simulated experiments of some artificial data sets, eight UCI data sets [50] and three bmp pictures in the next several sections.

Four kinds of artificial data sets (DS1 - DS4) are produced in a 2-D region  $[0, 600] \times [0, 600]$  by a program presented in Online Resource 4 of Supplementary Material of this paper. Other kinds of artificial data sets (DS5 - DS16) are produced in an interval  $[0, 600]$  in each dimension by a similar program. DS0 is produced in a 2-D region  $[0, 200] \times [0, 200]$  by a similar program. Iris et al. [50] are eight UCI data sets used in our experiments. Three bmp pictures (named Picture1, Picture2 and Picture3) are obtained from the Internet. Stable 2 of Supplementary Material presents the description of the experimental data sets.

In our simulated experiments, the maximum times of synchronization evolution in the while repetition of SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method is set as 50. MLSynC method is implemented using the two-level framework algorithm of MLSynC method.

The comparison results of these clustering algorithms are presented by eight figures (Figs. 4 - 5, Sfigs. 3 - 8 of Supplementary Material) and eleven tables (Stables 3 - 13 of Supplementary Material). Because of the limited pages, we only select two figures from Sfigs. 3 - 8 in the manuscript. Sfigs. 3 - 8 and Stables 3 - 13 are presented in Online Resource 6 of Supplementary Material. And the performance of algorithms is measured by time cost (second). Clustering quality of algorithms is measured by display figures of clustering results and three robust information-theoretic measures, Adjusted Mutual Information (AMI) [51], Normalized Mutual Information [52] (NMI) and Adjusted Variation Information (AVI) [51]. According to Vinh et al. [51], the higher the value of the three measures gets, the better the clustering quality of algorithms is. In simulations, we use a revised Matlab code according to [51] to compute the three clustering quality measures.

In Section 5.2, MLSynC method will be compared with SynC algorithm, ESynC algorithm, SSynC algorithm and some other classic clustering algorithms (K-Means [2], FCM [3], AP [19], DBSCAN [9], Mean Shift [53, 54], spectral clustering algorithm [16, 17], information-based clustering algorithm [49], DP clustering algorithm [28]) in clustering quality and time cost using some artificial data sets.



In Section 5.3, MLSynC method will be compared with SynC algorithm, ESynC algorithm, SSynC algorithm and some other classic clustering algorithms in clustering quality and time cost using eight UCI data sets.

In Section 5.4, MLSynC method will be compared with SynC algorithm, ESynC algorithm, SSynC algorithm and some other classic clustering algorithms in the compressed effect of clustering results, clustering quality and time cost using three bmp pictures.

In the experiments, parameter  $\delta$  used in SynC algorithm, ESynC algorithm, SSynC algorithm, MLSynC method, DBSCAN algorithm and Mean Shift algorithm is the threshold of Sdefinition 1 of Supplementary Material. In DBSCAN algorithm, parameter  $MinPts = 4$ , and parameter Eps is the same as parameter  $\delta$ .

The detailed discussion on how to select a proper value for parameter  $\delta$  in SynC algorithm is discussed in [20]. ESynC algorithm, SSynC algorithm and MLSynC method use Stheorem 1 and Sproperty 1 to select a proper value for parameter  $\delta$ . In SSynC algorithm and MLSynC method, parameter  $\varepsilon$  has a trivial effect on time cost and clustering results.

To simplify our simulation of the two-level framework algorithm of MLSynC method, we set the same value for parameter  $\delta_1, \delta_2, \dots, \delta_m$  and  $\delta_{merge}$ . So they are also named as  $\delta$ .

## 5.2 Experimental results of some artificial data sets (from DS1 - DS16)

### 5.2.1 The comparison of the clustering results among SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method (from DS1- DS4)

Stable 3 of Supplementary Material presents the comparison results of four synchronization clustering algorithms (SynC, ESynC, SSynC and MLSynC) by using four artificial data sets (from DS1 - DS4). In Stable 3, by intercomparing SynC, ESynC, SSynC and MLSynC, we observe that MLSynC is the fastest clustering algorithm. At the same time, MLSynC, SSynC and ESynC can get better local synchronization results than SynC in the four data sets.

In the two-level framework of MLSynC method, parameter  $m$  is set as 10. If we use two different partitioning methods (a near unevenly partitioning method and a near evenly partitioning method) in the four data sets, the two sequences of the number of clusters of 10 subsets in each data set are different. The comparison results are presented in Stable 4 of Supplementary Material. From Stable 4, we observe that two different partitioning methods result in different clustering distributions of 10 subsets in each data set.

### 5.2.2 The comparison of the clustering results among SynC algorithm, ESynC algorithm, SSynC algorithm, MLSynC method and some classical clustering algorithms (from DS1 - DS8)

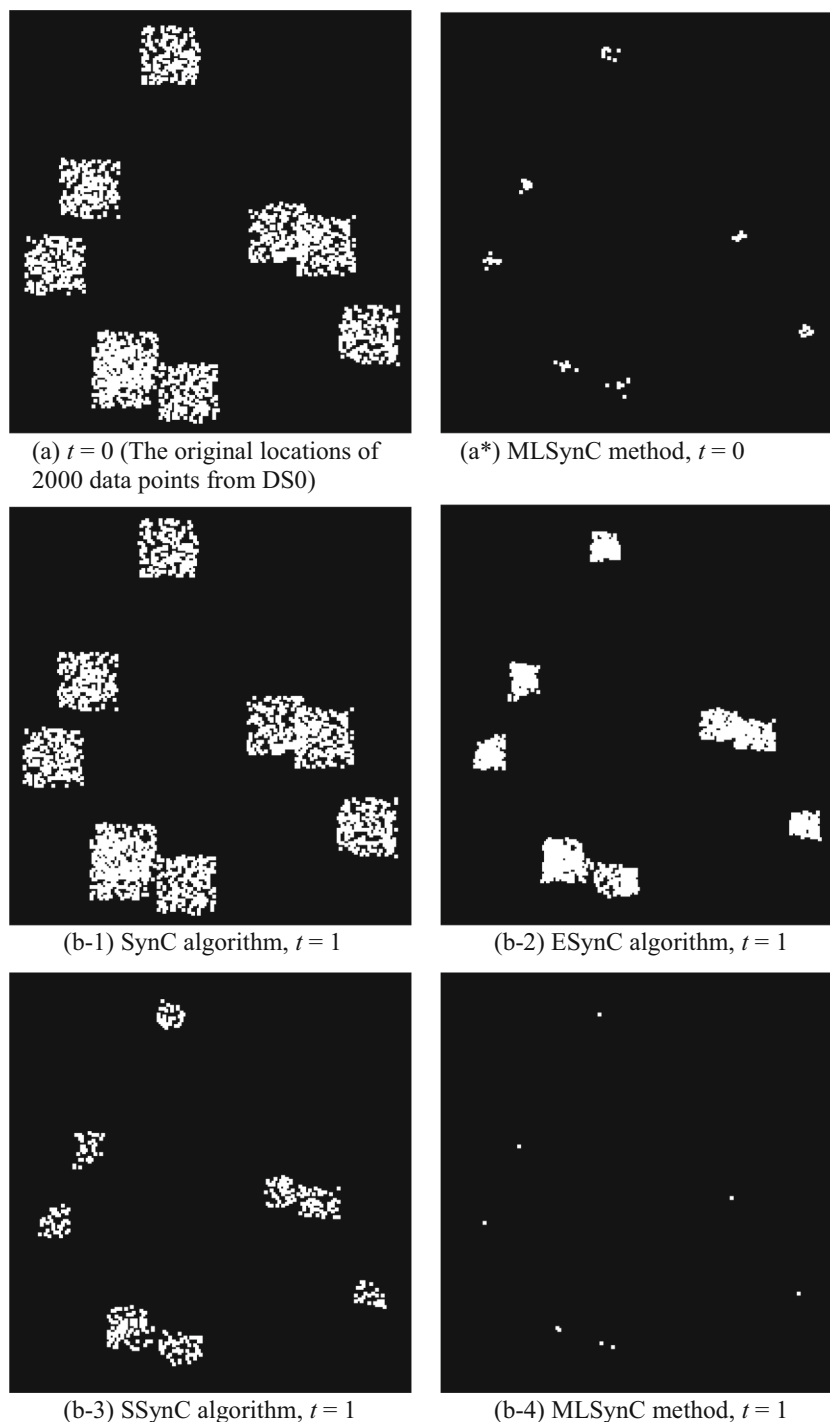
Fig. 4 and Sfigs. 4 - 6 of Supplementary Material present the comparison clustering results of several clustering algorithms by some display figures that reflect the clustering quality. From Fig. 4 and Sfigs. 4 - 6, we observe that MLSynC, SSynC and ESynC can get better clustering quality (obvious clusters or isolates displayed by figures) than SynC, AP, K-Means and FCM in the four artificial data sets (from DS1 - DS4). Mean Shift, DBSCAN can obtain similar clustering quality (obvious clusters displayed by figures) with MLSynC, SSynC and ESynC in these artificial data sets. Especially, MLSynC, SynC, ESynC and SSynC can all easily find some isolates if setting a proper value for parameter  $\delta$ , and MLSynC can get the same clustering results (the same clusters displayed by figures) with SSynC, ESynC in these data sets.

Stable 5 of Supplementary Material presents the clustering quality of several clustering algorithms (SynC, ESynC, SSynC, MLSynC and some classical clustering algorithms) by using six kinds of artificial data sets (DS2, DS4, DS5, DS6, DS7 and DS8). When computing AMI and NMI, the predefined cluster labels of the eight artificial data sets are used in the true\_mem that is an input file of the MATLAB code from [51]. In Stable 5, by intercomparing MLSynC, SynC, ESynC, SSynC and some classical clustering algorithms, we observe that MLSynC can get acceptable and similar clustering results with SSynC and ESynC in the eight data sets if the partitioning of the data set in MLSynC method is near evenly. Because the two data sets (DS4 ( $n = 1000$ ) and DS5 ( $n = 12,000$ )) have two connected clusters, MLSynC, SSynC and ESynC do not get the largest values of AMI and NMI. We also observe that the partitioning method of the data sets can affect the clustering results of MLSynC method.

Stable 6 of Supplementary Material presents the comparison results of several clustering algorithms in time cost. In Stable 6, for DS1, parameter  $\delta = 18$  in SSynC, ESynC, SynC and DBSCAN; parameter  $\delta = 25$  in MLSynC. For DS2, parameter  $\delta = 25$  in MLSynC, SSynC, ESynC, SynC and DBSCAN. For DS3, parameter  $\delta = 18$  in MLSynC, SSynC, ESynC, SynC and DBSCAN. For DS4, parameter  $\delta = 18$  in SSynC, ESynC, SynC and DBSCAN; parameter  $\delta = 20$  in MLSynC. Parameter  $\varepsilon = 1$  in SSynC and MLSynC. In MLSynC, parameter  $m$  is set as 10, and the two-level framework algorithm is used.

In Stable 6, intercomparing MLSynC, SynC, ESynC, SSynC, DBSCAN, FCM and K-Means, we observe that MLSynC is faster than SynC, ESynC, DBSCAN and SSynC. K-Means is the fastest clustering algorithm. Mean Shift and AP cannot run normally on a personal computer because the number of data points is set as 40,000.

**Fig. 2** The comparison of the dynamical clustering processes with time evolution among SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method. From **a** to **e** of Fig. 2, the data set is from DS0 with 2000 points, parameter  $\delta$  is set as 18 in the four algorithms, and parameter  $\varepsilon$  is set as 1 in SSynC algorithm and MLSynC method. In MLSynC method, parameter  $m$  is set as 10, and the two-level framework algorithm is used. **a\***, **b-4**, **c-4**, **d-4** and **e-4** are the evolution displays in the “collect stage” of the two-level framework algorithm of MLSynC method



Sfigs. 7 - 11 and Stable7 of Supplementary Material present appended experimental results in clustering quality among three famous clustering algorithms, spectral clustering algorithm [16, 17], information-based clustering algorithm [55] and DP clustering algorithm [28]. Stable 7 presents the setting of parameters for three clustering algorithms in four artificial data sets (DS1 - DS4,  $n = 400$ ). Sfig. 7 presents the clustering results and decision graphs of DP clustering algorithm.

Sfigs. 8 - 11 present the comparison of the clustering results of three famous clustering algorithms using four artificial data sets. In four artificial data sets (DS1 - DS4,  $n = 400$ ), DP clustering and MLSynC method can find correct clusters, spectral clustering and information-based clustering cannot obtain actual clustering results. DP clustering cannot explore isolates of two data sets (DS1 and DS3,  $n = 400$ ), and MLSynC method can explore isolates from them.

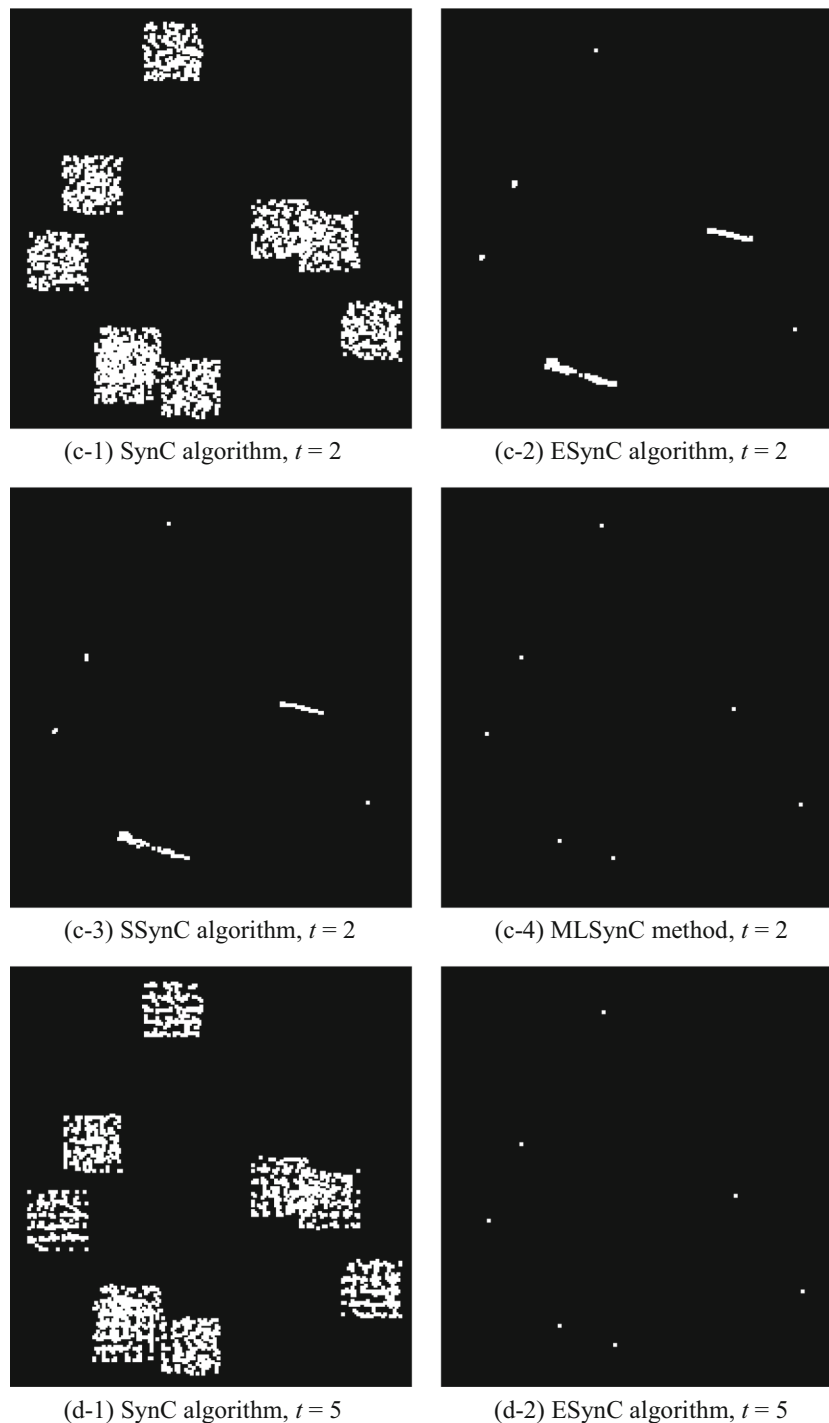


Fig. 2 (continued)

### 5.2.3 The comparison of the valid interval of parameter $\delta$ among MLSynC method, SynC algorithm, ESynC algorithm, SSynC algorithm, DBSCAN algorithm and mean shift algorithm using some artificial data sets (from DS5 - DS16)

Here we compare the valid interval of parameter  $\delta$  among MLSynC method, SynC algorithm, ESynC algorithm, SSynC algorithm, DBSCAN algorithm and Mean Shift algorithm.

Stable 8 of Supplementary Material presents the comparison results of the valid interval of parameter  $\delta$  among MLSynC, SynC, ESynC, SSynC, DBSCAN and Mean Shift. Here,  $[e_k, e_{k+1}]$  can be obtained from Eq.(8) of [49]. In Stable 8, intercomparing MLSynC, SynC, ESynC, SSynC, DBSCAN and Mean Shift, we observe that although the valid interval of parameter  $\delta$  in MLSynC is shorter than that in SSynC and ESynC, the valid interval of

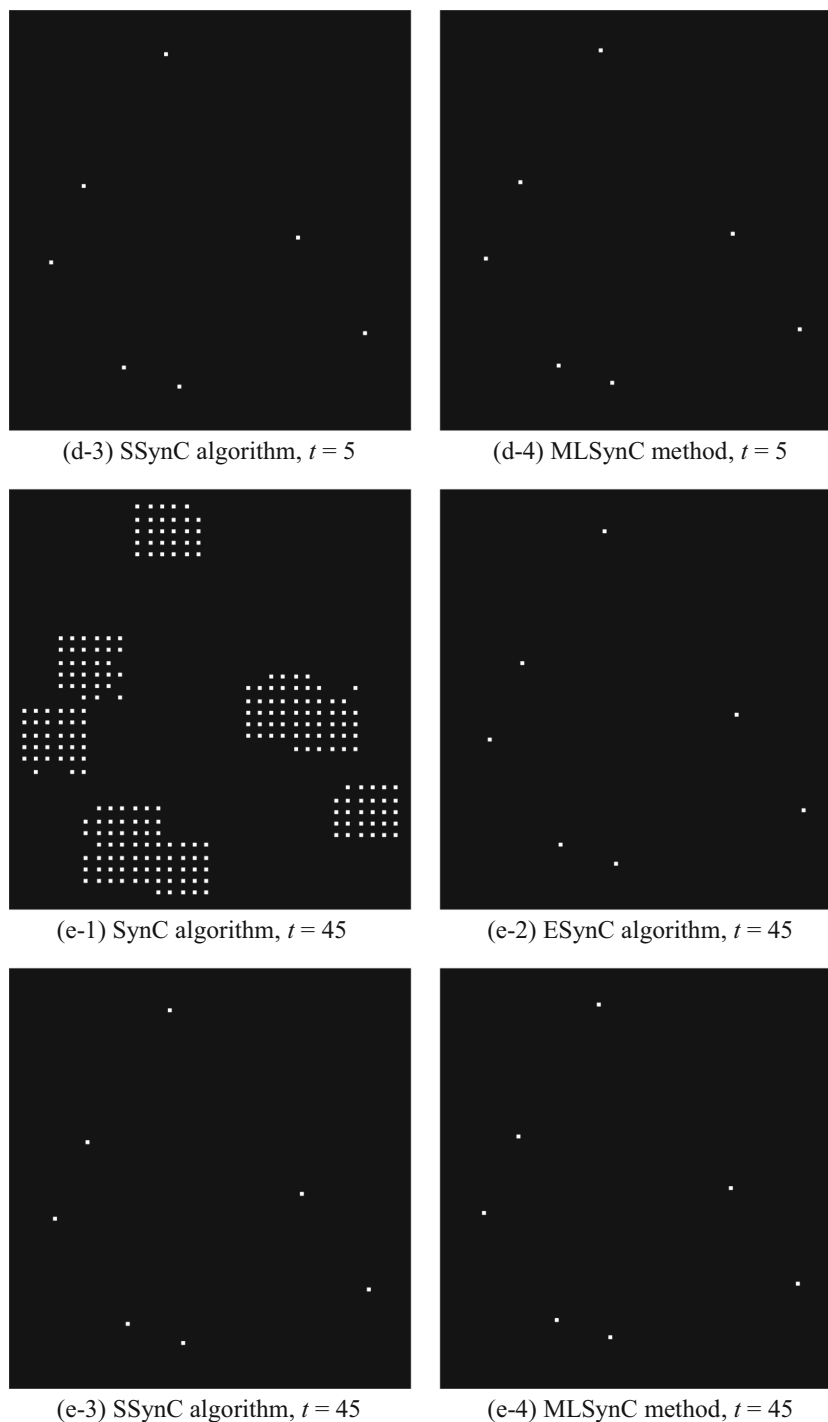


Fig. 2 (continued)

parameter  $\delta$  in MLSynC is long enough in many kinds of data sets.

Stable 9 of Supplementary Material compares the valid interval of parameter  $\delta$  in MLSynC method for several different values of parameter  $\varepsilon$  using some artificial data sets with different dimensions. In Stable 9, intercomparing several different values of parameter  $\varepsilon$ , we observe that if parameter  $\varepsilon$  is less than parameter  $\delta$ , the valid

interval of parameter  $\delta$  has a very small difference for several different values of parameter  $\varepsilon$ .

### 5.3 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 into an interval  $[0, 600]$  in each dimension in the

experiments. When computing AMI and NMI, because we do not know the true cluster labels of these UCI data sets, the class labels of these UCI data sets are used in the true\_mem that is an input file of the MATLAB code from [51].

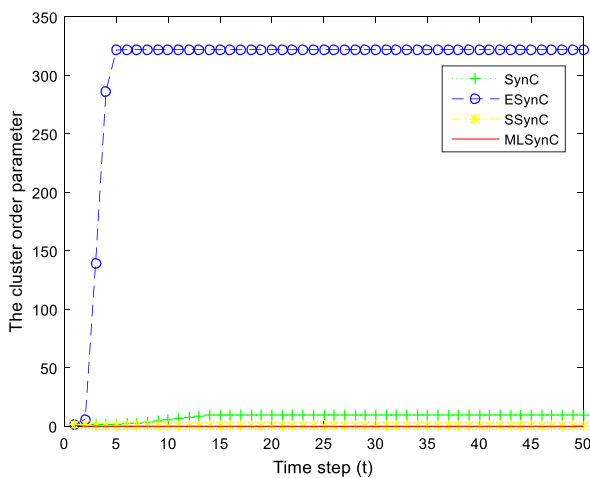
### 5.3.1 The comparison of the clustering results among MLSynC method, SynC algorithm, ESynC algorithm and SSynC algorithm

Stable 10 of Supplementary Material presents the comparison results of four synchronization clustering algorithms (MLSynC method, SynC algorithm, ESynC algorithm and SSynC algorithm) by using eight UCI data sets. In Stable 10, intercomparing MLSynC method, SynC algorithm, ESynC algorithm, SSynC algorithm, we observe that

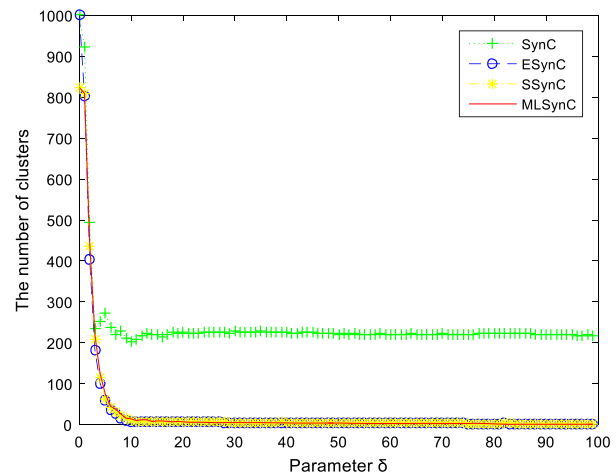
MLSynC method, ESynC algorithm and SSynC algorithm can get better local synchronization results than SynC algorithm in the eight UCI data sets, and MLSynC method is the fastest algorithm. From this simulated experiment, we also find that if the number of points in the data set is small and we use an uneven partition method, the difference of clustering effect between MLSynC method and SSynC algorithm is large.

### 5.3.2 The comparison of the clustering quality among MLSynC method, SynC algorithm, ESynC algorithm, SSynC algorithm and some classical clustering algorithms

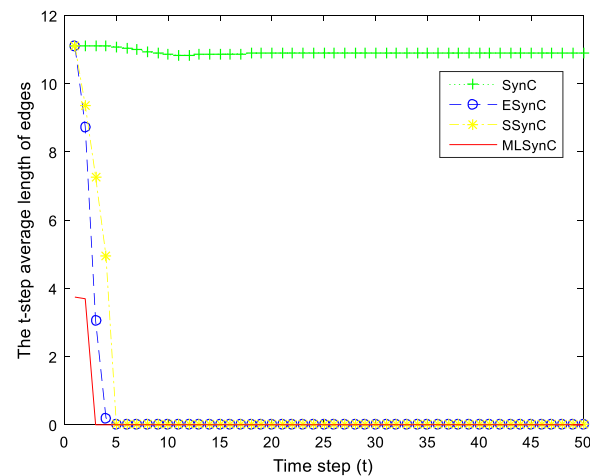
Stable 11 of Supplementary Material presents the comparison clustering quality of several clustering algorithms (MLSynC



(a) The cluster order parameter with  $t$ -step evolution ( $t: 0 - 49$ )



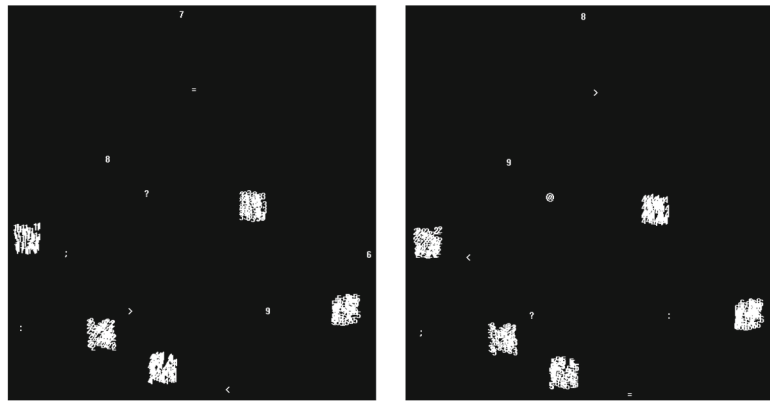
(c) The relation between the final number of clusters and parameter  $\delta$  ( $\delta: 0 - 99$ ) among four synchronization algorithms.



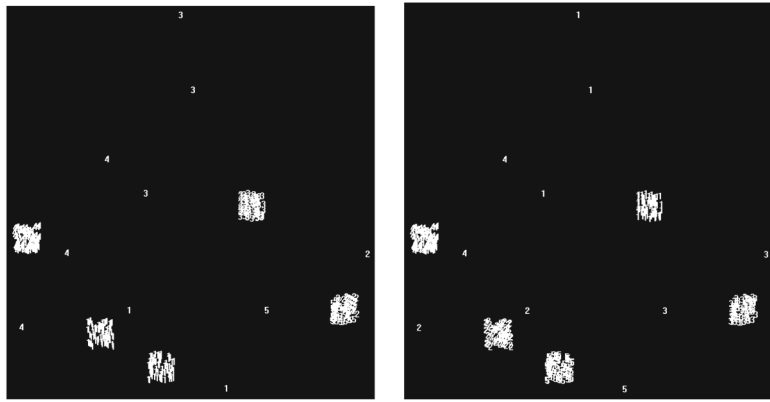
(b) The  $t$ -step average length of edges ( $t: 0 - 49$ )

**Fig. 3** The comparison of SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method in two measure indexes of clustering result and the relation between the final number of clusters and parameter  $\delta$ . In Fig. 3, the data set is from DS0 with 2000 points, and parameter  $\varepsilon$  is set as 1 in SSynC algorithm and MLSynC method. In MLSynC method, parameter  $m$  is set as 10, the two-level framework

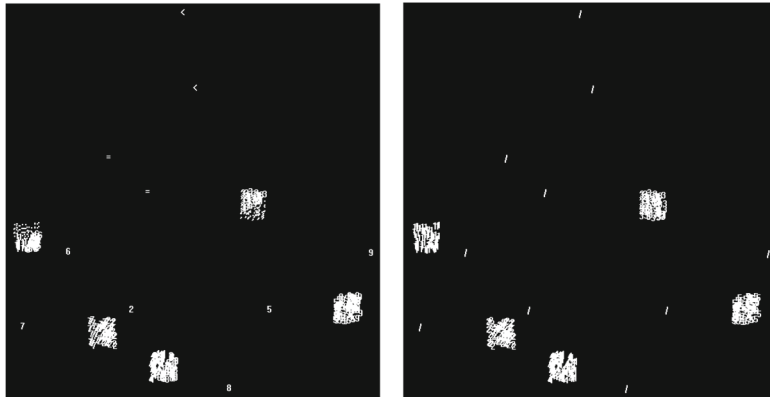
algorithm is used. In **a** and **b**, parameter  $\delta$  is set as 18 in the four algorithms. In MLSynC method of **a** and **b**, two indexes (The cluster order parameter with  $t$ -step evolution and the  $t$ -step average length of edges) are computed in the “collect stage” of the two-level framework algorithm of MLSynC method



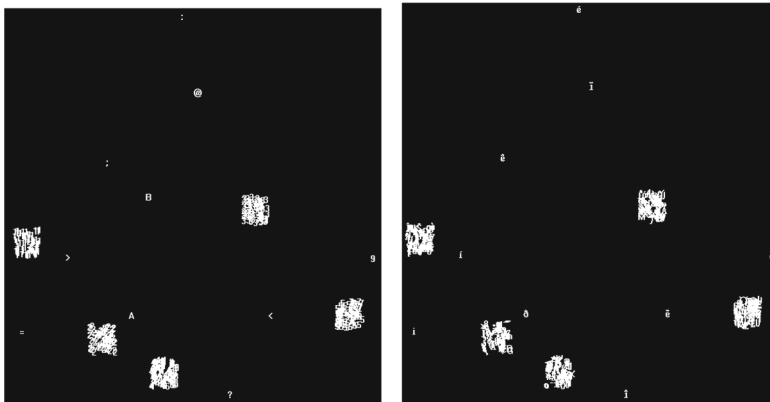
(a) Clusters identified by MLSynC (15 clusters or isolates) (b) Clusters identified by ESynC and SSynC (15 clusters or isolates)



(c) Clusters identified by K-Means (predefined 5 clusters) (d) Clusters identified by FCM (predefined 5 clusters)



(e) Clusters identified by AP (13 clusters) (f) Clusters identified by DBSCAN (5 clusters)



(g) Clusters identified by Mean Shift (18 clusters) (h) Clusters identified by SynC (192 clusters or isolates)

◀ **Fig. 4** The comparison of the clustering results of several algorithms (DS1,  $n = 400$ ). In Fig. 4, parameter  $\delta = 18$  in SynC, ESynC, SSynC, DBSCAN, Mean Shift and MLSynC method; the number of data points  $n = 400$ ; parameter  $\varepsilon = 1$  in SSynC algorithm and MLSynC method. In MLSynC method, parameter  $m$  is set as 2, and the two-level framework algorithm is used

method, SynC algorithm, ESynC algorithm, SSynC algorithm and some classical clustering algorithms) using eight UCI data sets. In Stable 11, by intercomparing these clustering algorithms, we observe that MLSynC method gets the largest value of AMI and NMI in five UCI data sets (Iris, Wdbc, Glass, Segmentation and Cloud) if it uses a sequential and uneven partitioning method. So we can say that MLSynC method often gets better clustering results than some clustering algorithms in some UCI data sets. From the final number of clusters in Stable 11, we observe that MLSynC method, SSynC algorithm and ESynC algorithm can get better local synchronization results than SynC algorithm.

Stable 12 presents the comparison of clustering quality among three famous clustering algorithms (spectral clustering algorithm [16, 17], information-based clustering algorithm [55] and DP clustering algorithm [28]) using eight UCI data sets. Stable 13 presents the comparison of clustering quality among MLSynC method, spectral clustering algorithm, information-based clustering algorithm and DP clustering algorithm. In eight UCI data sets, spectral clustering and DP clustering get better clustering results than information-based clustering. MLSynC method obtains the best clustering quality in seven UCI data sets, and DP clustering obtains the best clustering quality in one UCI data set.

## 5.4 Experimental results of three bmp pictures

The value in RGB (Red, Green and Blue) color space of pixel points is in an interval  $[0, 255]$  in each dimension. In Stable 14 and Sfig. 13 of Supplementary Material, parameter  $\varepsilon$  is set as 1 in MLSynC method and SSynC algorithm. In MLSynC method of Stable 14 and Sfig. 13, parameter  $m$  is set as 10, and the two-level framework algorithm is used.

### 5.4.1 The comparison of the clustering results among SynC algorithm, ESynC algorithm, SSynC algorithm and MLSynC method

Stable 14 presents the comparison results of four synchronization clustering algorithms (SynC, ESynC, SSynC and MLSynC) using three pixel-point data sets from RGB color space of three bmp pictures. In Stable 14, by intercomparing SynC, ESynC, SSynC and MLSynC, we observe that MLSynC is the fastest clustering algorithm. At the same time, MLSynC, SSynC and ESynC can get better local synchronization results than SynC in these pixel-point data sets.

### 5.4.2 The comparison of the clustering compressed effect among MLSynC method, SynC algorithm, ESynC algorithm, SSynC algorithm and some classical clustering algorithms

Sfig. 12 lists Picture3 and its RGB space distribution of  $200 * 200$  pixel points. Sfig. 13 of Supplementary Material lists the original picture and several compressed pictures of Picture3. In Sfig. 13, several compressed pictures are drawn by using the means of clusters obtained by clustering the  $200 * 200$  pixel points of Picture3 in RGB color space using different algorithms. Because AP algorithm needs too much time and space for Picture3, this experiment does not use it. Although the queue of DBSCAN algorithm stops over, it still gets a compressed picture basing on the clustering results. From Sfig. 13, we observe that MLSynC method, ESynC algorithm and SSynC algorithm can get multi-level clustering compressed effects for different values of parameter  $\delta$ .

## 5.5 Analysis and conclusions of experimental results

From the comparison experimental results of these tables (Stable 3, Stable 6, Stable 10 and Stable 14 of Supplementary Material), we observe that MLSynC method is faster than SSynC algorithm, ESynC algorithm and SynC algorithm. We think that MLSynC method is superior to SSynC algorithm, ESynC algorithm and SynC algorithm in time cost because of its framework of “divide and collect”.

From the simulations of some artificial data sets (from DS5 - DS16), we observe that the effective interval of parameter  $\delta$  in MLSynC method is enough long just like Mean Shift algorithm, DBSCAN algorithm, SSynC algorithm and ESynC algorithm. In some cases, the effective interval of parameter  $\delta$  in MLSynC method is longer than that in DBSCAN algorithm.

In some display figures (Sfigs. 3 – 6 of Supplementary Material), by intercomparing MLSynC method, SSynC algorithm, ESynC algorithm and SynC algorithm, we observe that MLSynC method can explore the same clusters and isolates (displayed by some figures) with ESynC algorithm and SSynC algorithm if the partition in MLSynC method is near well-proportioned or unaided (each subset is independent with each other in the space). In many kinds of data sets, MLSynC method, SSynC algorithm and ESynC algorithm can explore obvious clusters or isolates if selecting a proper value for parameter  $\delta$ , 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 MLSynC method, SSynC algorithm and ESynC algorithm. In many data sets, MLSynC method, ESynC algorithm, SSynC algorithm, DP algorithm, Mean Shift algorithm and DBSCAN algorithm have a better ability than SynC algorithm, K-Means algorithm, FCM algorithm, spectral clustering

algorithm, information-based clustering algorithm and AP algorithm in exploring clusters and isolates. Especially, AP algorithm needs the longest time. From the simulation, we also find that DP clustering algorithm cannot explore isolates and determine the number of clusters automatically.

MLSynC method is an improved clustering algorithm with faster clustering speed than SSynC algorithm and ESynC algorithm almost in all cases. Usually, parameter  $\varepsilon$  has a long effective interval (for example, the effective interval of parameter  $\varepsilon$  is about  $(0, 10]$  if parameter  $\delta > 15$ ). In simulations, we observe that if parameter  $\varepsilon$  gets some different values in its effective interval, the clustering results of MLSynC method is almost the same except the time cost.

Because the values in RGB space, the pixel points of Picture3 are almost continuous and have no obvious clusters. In this case, MLSynC method, SSynC algorithm and ESynC algorithm can get more obvious multi-level compressed effects than some other clustering algorithms, such as K-Means algorithm and FCM algorithm. In simulations, we also observe that DBSCAN algorithm needs more space than MLSynC method, SSynC algorithm and ESynC algorithm because of its recursive process.

Because of the limited page space, we only select some typical data sets (sixteen kinds of artificial data sets, eight UCI data sets and three bmp pictures) used in our experiments. For all experimental data sets, we observe that MLSynC method improves ESynC algorithm in time cost. For other data sets, we think MLSynC method is still superior to SynC algorithm in time cost. We believe that the selection of experimental data sets is not biased.

From Fig. 1 and some other simulated experimental results, we conclude the application condition of MLSynC method. In any one of the following two cases, MLSynC method can get a similar clustering effect with SSynC algorithm and ESynC algorithm.

**Case 1:** The spatial distribution of any partitioned data subset is almost the same as that of the original data set.

**Case 2:** Any two partitioned data subsets cannot intersect, cannot be joined, or cannot be much near (less than or equal to parameter  $\delta$ ). In this case, any unabridged cluster of the original data set will not be partitioned into multiple near (larger than parameter  $\delta$ ) small clusters after the partition process of MLSynC method.

When the partition of the original data set does not satisfy any case above, MLSynC method will often get different clustering effects with SSynC algorithm and ESynC algorithm.

## 6 Conclusions

This paper presents an improved synchronization clustering method, MLSynC, which often gets better clustering results than the original synchronization clustering algorithm, SynC. From the experimental results, we observe that MLSynC method can often obtain less iterative times, faster clustering speed and better clustering quality than SynC algorithm in multi-kinds of data sets.

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

- (1) It develops an effective framework of “divide and collect” in clustering field by using a linear weighted Vicsek model.
- (2) It presents two concrete implementations of MLSynC method, a two-level framework algorithm and a recursive algorithm.
- (3) It validates the improved effect of MLSynC method in time cost and clustering quality by some simulated experiments.

MLSynC method is also 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  $\delta$  has some valid interval that can be determined by using an exploring method presented in [49], the heuristic method described by Stheorem 1 and Sproperty 1, or using the MDL-based method presented in [20].

Usually, some clustering algorithms will get a large deviation when they are transplanted on the MapReduce framework. Only some specially designed clustering methods based on MapReduce framework and ensemble clustering methods can obtain the same, similar, or better clustering results. MLSynC method has some similarities with the MapReduce framework. So we can say it is an application example of the MapReduce framework in clustering field. In MLSynC method, the relation between the distribution deviation of partitioned data subsets and the clustering results should be investigated. How the partitioning strategies of the original data set affect the clustering results can also be further researched.

This work opens some possibilities for further improvement and investigation. First, improve MLSynC method in time cost further. For example, designing a similarity-preserving hashing function that needs  $O(1)$  time complexity is valuable in the process of constructing  $\delta$  near neighbor point set. Second, extend the applicability and explore the clustering effect of our algorithms in high-dimensional data. Third, implement MLSynC method on a cluster with a parallel programming model or the MapReduce framework.



**Acknowledgments** This work was supported by the projects from Natural Science Research in Colleges and Universities of Anhui Province of China (grant number: KJ2019ZD15, KJ2019A0158), the University Synergy Innovation Program of Anhui Province (grant number: GXXT-2019-002), Anhui Polytechnic University (grant number: 2018YQQ031), Chongqing Cutting-edge and Applied Foundation Research Program (grant number: cstc2016jcyjA0521), Chongqing Three Gorges University (grant number: 16PY08) and National Natural Science Foundation of China (grant number: 61976005). The authors thank the editors and the anonymous reviewers for their useful suggestions that help us to improve this paper.

## Compliance with ethical standards

**Conflict of interest** None.

## References

- Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. *ACM Comput Surv* 31(3):264–323
- MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: *Proceedings of the 5-th MSP*. University of California Press, Berkeley, pp 281–297
- Bezdek JC (1981) *Pattern recognition with fuzzy objective function algorithms*. Plenum Press, New York
- Bouguettaya A, Yu Q, Liu X et al (2015) Efficient agglomerative hierarchical clustering. *Expert Syst Appl* 42(5):2785–2797
- Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithm for clustering large databases. In: *Proceedings of ACM SIGMOD*, pp 73–84
- Karypis G, Han EH, Kumar V (1999) CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. *IEEE Comput* 32(8):68–75
- 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
- Ankerst M, Breunig MM, Kriegel HP, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: *Proceedings of ACM SIGMOD*, pp 49–60
- Ester M, Kriegel HP, 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
- Agrawal R, Gehrke J, Gunopulos D et al (1998) Automatic subspace clustering of high dimensional data for data mining application. In: *Proceedings of ACM SIGMOD*, pp 94–105
- Wang W, Yang J, Muntz R (1997) STING: a statistical information grid approach to spatial data mining. In: *Proceedings of VLDB*, pp 186–195
- Theodoridis S, Koutroumbas K (2006) *Pattern recognition*. Academic, New York
- Tan PN, Steinbach M, Kumar V (2005) *Introduction to data mining*. Addison Wesley, Boston
- Zahn CT (1971) Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans Comput* C-20(1):68–86
- Horn D, Gottlieb A (2002) Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Phys Rev Lett* 88(1):018702
- Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: analysis and an algorithm. In: *Proceedings of NIPS*, pp 849–856
- Luxburg UV (2007) A tutorial on spectral clustering. *Stat Comput* 17(4):395–416
- Girolami M (2002) Mercer kernel-based clustering in feature space. *IEEE Trans Neural Netw* 13:780–784
- Frey BJ, Dueck D (2007) Clustering by passing messages between data points. *Science* 315(16):972–976
- Böhm C, Plant C, Shao J et al (2010) Clustering by synchronization. In: *Proceedings of ACM SIGKDD*, Washington, USA, pp 583–592
- Shao J, Yang Q, Böhm C, Plant C (2011) Detection of arbitrarily oriented synchronized clusters in high-dimensional data. In: *Proceedings of ICDM*, pp 607–616
- 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 Trans Knowl Data Eng* 25(4):893–905
- Chen X (2014) A fast synchronization clustering algorithm. arXiv: 1407.7449 [cs.LG]. <http://arxiv.org/abs/1407.7449>
- Chen X (2017) An effective synchronization clustering algorithm. *Appl Intell* 46(1):135–157
- Chen X (2018) Fast synchronization clustering algorithms based on spatial index structures. *Expert Syst Appl* 94:276–290
- Hang W, Choi K, Wang S (2017) Synchronization clustering based on central force optimization and its extension for large-scale datasets. *Knowl-Based Syst* 118:31–44
- Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. *Science* 344(6191):1492–1496
- Vicsek T, Czirok A, Ben-Jacob E et al (1995) Novel type of phase transitions in a system of self-driven particles. *Phys Rev Lett* 75(6):1226–1229
- Jadbabaie A, Lin J, Morse AS (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans Autom Control* 48(6):998–1001
- Wang L, Liu Z (2009) Robust consensus of multi-agent systems with noise. *Sci China Ser F: Inform Sci* 52(5):824–834
- Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. *Commun ACM* 51(1):107–113
- GrÅunwald P (2005) *A tutorial introduction to the minimum description length principle*. MIT Press, Cambridge
- Von der Malsburg C (1973) Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik* 14:85–100
- Kohonen T (1982) Self-organized formation of topologically correct feature maps. *Biol Cybern* 43(1):59–69
- Kohonen T (1989) On the significance of internal representations in neural networks. In: *Proceedings of ICANN*, pp 158–162
- Grossberg S (1976a) Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biol Cybern* 23(3):121–134
- Grossberg S (1976b) Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions. *Biol Cybern* 23(4):187–202
- Du KL (2010) Clustering: a neural network approach. *Neural Netw* 23(1):89–107
- Brito da Silva LE, Elnabarawy I, Wunsch DC II (2019) A survey of adaptive resonance theory neural network models for engineering applications. *Neural Netw* 120:167–203
- Amis GP, Carpenter GA (2010) Self-supervised ARTMAP. *Neural Netw* 2:265–282
- Seiffert J (2019) Adaptive resonance theory in the time scales calculus. *Neural Netw* 120:32–39
- Bradley PS, Fayyad UM, Reina C et al (1998) Scaling clustering algorithms to large databases. In: *Proceedings of ACM SIGKDD*, pp 9–15
- Sculley D (2010) Web-scale k-means clustering. In: *Proceedings of WWW*, pp 1177–1178
- Urruty T, Djeraba C, Simovici DA (2007) Clustering by random projections. In: *Proceedings of ICDM*, pp 107–119

46. Fern XZ, Brodley CE (2003) Random projection for high dimensional data clustering: a cluster ensemble approach. In: Proceedings of ICML, pp 186–193
47. Milenova BL, Campos MM (2002) O-cluster: scalable clustering of large high dimensional data sets. In: Proceedings of ICDM, pp 290–297
48. Rathore P, Kumar D, Bezdek JC, Rajasegarar S, Palaniswami M (2019) A rapid hybrid clustering algorithm for large volumes of high dimensional data. *IEEE Trans Knowl Data Eng* 31(4):641–654
49. Chen X (2015) A new clustering algorithm based on near neighbor influence. *Expert Syst Appl* 42(21):7746–7758
50. Dua D, Graff C (2019) UCI machine learning repository. University of California, School of Information and Computer Science, Irvine. <http://archive.ics.uci.edu/ml>
51. Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J Mach Learn Res* 11:2837–2854
52. Strehl A, Ghosh J (2002) Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *J Mach Learn Res* 3: 583–617
53. Fukunaga K, Hostetler L (1975) The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans Inf Theory* 21(1):32–40
54. Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis. *IEEE Trans Pattern Anal* 24(5):603–619
55. Slonim N, Atwal GS, Tkacik G, Bialek W (2005) Information-based clustering. *Proc Natl Acad Sci U S A* 102(51):18297–18302

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Xinquan Chen** is a professor of the School of Computer & Information, Anhui Polytechnic University, Wuhu, Anhui, PR China. He received the Ph.D degree from South China University of Technology in 2007. His research interests include data mining, machine learning, clustering algorithms, and optimization method. The corresponding author of this paper.



**Yirou Qiu** is a student of the School of Optics and Photonics, Beijing Institute of Technology, Beijing, PR China. Her research interests include Image processing, data analysis and optimization method.