Appl Intell (2017) 46:135–157
DOI 10.1007/s10489-016-0814-y

An effective synchronization clustering algorithm

Xinquan Chen[1,2]

Published online: 1 August 2016

Abstract This paper presents an Effective Synchronization Clustering (ESynC) algorithm using a linear version of Vicsek model. The development of ESynC algorithm is inspired by Synchronization Clustering (SynC) algorithm and Vicsek model. After some analysis and experimental comparison, we observe that ESynC algorithm based on the linear version of Vicsek model can get better local synchronization effect than SynC algorithm based on an extensive Kuramoto model and a similar synchronization clustering algorithm based on the original version of Vicsek model. By some simulated experiments of some artificial data sets, eight UCI data sets, and three picture data sets, we observe that ESynC algorithm not only gets better local synchronization effect but also needs less iterative times and time cost than SynC algorithm. We also introduce an Improved ESynC algorithm (IESynC algorithm) in time cost by combining multidimensional grid partitioning method and Red-Black tree structure. By some simulated experiments, we observe that IESynC algorithm can get some improvement of time cost than ESynC algorithm in some data sets. Extensive comparison experiments with some class clustering algorithms demonstrate that our two algorithms can often get acceptable clustering results in many cases. At last, it gives several solid and insightful future research suggestions.

Keywords Clustering · Synchronization · Kuramoto model · A linear version of Vicsek model · Near neighbor point set

1 Introduction

Clustering is an unsupervised learning method that tries to search some obvious clusters in unlabeled data by maximizing the similarity of the objects in a common cluster and minimizing the similarity of the objects in different clusters [20]. Clustering has been used in many areas, such as pattern classification, image processing, marketing analysis, information retrieval, bioinformatics, and so on.

Clustering algorithms have been studied for decades. We think that 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. In real-world applications, some data have complex distributions, others have diversiform types, others have great capacity, and others have many noises or isolates. So there is a continued demand for researching different kinds of clustering methods. In order 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 to develop new efficient and effective clustering algorithms.

The traditional clustering methods are usually classified into partitioning methods [3, 27], hierarchical methods [16, 23, 47], density-based methods [2, 11, 30, 31], grid-based methods [1, 45], model-based methods [41], and graph-based methods [21, 32]. Recent clustering methods have

Electronic supplementary material The online version of this article (doi:10.1007/s10489-016-0814-y) contains supplementary material, which is available to authorized users.

✉ Xinquan Chen
chenxqscut@126.com

[1] Big Data Research Center, University of Electronic Science and Technology, Sichuan, China

[2] School of Computer Science and Engineering, Chongqing Three Gorges University, Chongqing, China

quantum clustering methods [17], spectral clustering methods [26, 33], and synchronization clustering methods [4, 18, 34–39].

Recently, several original clustering algorithms, such as Affinity Propagation (AP) algorithm [13], Synchronization Clustering (SynC) algorithm [4], and clustering by fast search and find of Density Peaks (DP) algorithm [30], were published. AP algorithm is a new type of clustering algorithm published on *Science* in 2007. After AP algorithm was published, clustering based on probability graph models grew a new research direction. As we know, SynC algorithm [4] is the first clustering algorithm based on synchronization model. After SynC algorithm was presented, synchronization clustering methods attract the interest of some researchers. Some synchronization clustering methods [18, 34–39] were published from different views. DP algorithm [30] 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" [30]. 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 also 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, which is a famous synchronization clustering algorithm presented in [4], claimed that it can find the intrinsic structure of the data set without any distribution assumptions and handle outliers by dynamic synchronization [4].

This paper reveals some fundamental differences among synchronization clustering based on an extensive Kuramoto model [4], synchronization clustering based on the original version of Vicsek model, and synchronization clustering based on a linear version of Vicsek model. After explored the synchronization clustering method based on the linear version of Vicsek model, we present an Effective Synchronization Clustering (ESynC) algorithm using the linear version of Vicsek model. ESynC algorithm is inspired by SynC algorithm and Vicsek model.

The remainder of this paper is organized as follows. Section 2 lists some related work. Section 3 gives some basic knowledge. Section 4 introduces ESynC algorithm and its improved version, IESynC algorithm. Section 5 validates our two algorithms by some simulated experiments. Conclusions and future work are presented in Section 6.

2 Related work

This paper is inspired by several papers [4, 19, 42, 44].

In 1995, Vicsek et al. [42] presented a basic model of multi-agent systems that contains noise effects. This basic model can also be regarded as a special version of Reynolds model [29]. Simulation results demonstrate that some systems using Vicsek model [42] or one-dimensional model presented by Czirok et al. [10] can be synchronized when they have large population density and small noise. Naturally, we expect that this kind of model can be used to explore clusters and noises of some data sets by local synchronization. In 2003, Jadbabaie et al. [19] analyzed a simplified Vicsek model without noise effects and provided a theoretical explanation for the nearest neighbor rule that can cause all agents to eventually move in the same direction. In 2008, Liu et al. [25] provided the synchronization property of Vicsek model after given initial conditions and the model parameters. In 2009, Wang et al. [44] researched Vicsek model under noise disturbances and presented some theoretical results. In 2010, Nagy et al. [28] found a well-defined hierarchical leader-follower influential network among pigeon flocks. So they suggested that hierarchical organization of group flight might be more efficient than an egalitarian one. After that, some reports about the communication mechanism of bird flocks were published in some famous journals, such as *Nature* and its sub journals, *PNAS*, and *PRL*. In 2014, Zhang et al. [46] found that pigeon flocks adopted a mode that switches between hierarchy and egalitarian. They think the switching mechanism of pigeon flocks is promising for some industrial applications, such as multi-robot system coordination and unmanned vehicle formation control. In 2015, Chen et al. [8] found that pigeon flocks adopted a simple two-level interactive network containing one leader and some followers. They think that "the two-level organization of group flight may be more efficient than a multilevel topology for small pigeon flocks" [8].

In 2010, Böhm et al. presented a novel clustering approach, SynC algorithm [4], 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 [4]. In order to implement automatic clustering, those natural clusters can be discovered by using the Minimum Description Length (MDL) principle [15]. After SynC algorithm was presented, Shao et al. published several synchronization clustering papers from different views [34–39]. In order to detect the outliers from a real complex data set more naturally, a novel outlier detection algorithm, "Out of Synchronization" [34], was presented from a new perspective. In order to find subspace clusters of some high-dimensional sparse data sets, a novel effective and efficient subspace clustering algorithm, ORSC [35], was proposed. In order to find the intrinsic patterns of a complex graph, a novel and robust graph clustering algorithm, RSGC [37], 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 [38], was presented based on synchronization and the MDL principle. In 2013, Huang et al. [18] also presented a synchronization-based hierarchical clustering method basing on the work of [4]. In 2014, Chen [6] presented a Fast Synchronization Clustering (FSynC) algorithm basing on the work of [4]. FSynC algorithm, which is a parametric algorithm, is an improved version of SynC algorithm by combining multidimensional grid partitioning method and Red-Black tree structure to construct the near neighbor point sets of all points [6].

Recent years, some physicists also researched the explosive synchronization of some complexity networks to uncover the underlying mechanisms of the synchronization state [22, 24, 48]. In these papers, the synchronization rules of some networks were explored.

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 algorithms clearly, some concepts are presented first.

Definition 1 [5] The δ near neighbor point set $\delta(P)$ of point P is defined as:

$$\delta(P) = \{X | 0 < dis(X, P) \leq \delta, X \in S, X \neq P\}, \tag{1}$$

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

Definition 2 [4]. The extensive Kuramoto model for clustering is defined as:

Point $X = (x_1, x_2, \ldots, x_d)$ is a vector in d-dimensional Euclidean space. If point X is regarded as a phase oscillator according to Kuramoto model, with an interaction in the δ near neighbor point set $\delta(X)$, then the dynamics of the k-th dimension x_k $(k = 1, 2, \ldots, d)$ of point X over time is described by:

$$x_k(t+1) = x_k(t) + \frac{1}{|\delta(X(t))|} \sum_{Y \in \delta(X(t))} \sin(y_k(t) - x_k(t)), \tag{2}$$

where $X(t = 0) = (x_1(0), x_2(0), \ldots, x_d(0))$ represents the original phase of point X, and $x_k\ (t+1)$ describes the renewal phase value in the k-th dimension of point X at the t step evolution.

Definition 3 The t-step δ near neighbor undirected graph $G_\delta(t)$ of the data set$S = \{X_1, X_2, \ldots, X_n\}$ is defined as:

$$G_\delta(t) = (V(t), E(t)), \tag{3}$$

where $V(t = 0) = S = \{X_1, X_2, \ldots, X_n\}$ is the original vertex set, $E(t = 0) = \{(X_i, X_j) | X_j \in \delta(X_i), X_i (i = 1, 2, \ldots, n) \in S\}$ is the original edge set. $V(t) = \{X_1(t), X_2(t), \ldots, X_n(t)\}$is the t-step vertex set of the data set$S$, $E(t) = \{(X_i(t), X_j(t)) | X_j(t) \in \delta(X_i(t)), X_i(t)\ (i = 1, 2, \ldots, n) \in V(t)\}$ is the t-step edge set, and the weight computing equation of edge (X_i, X_j) is $weight(X_i, X_j) = dis(X_i, X_j)$.

Definition 4 The t-step average length of edges, $AveLen(t)$, in a t-step δ near neighbor undirected graph $G_\delta(t)$ is defined as:

$$AveLen(t) = \frac{1}{|E(t)|} \sum_{e \in E(t)} |e| \tag{4}$$

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 our two algorithms, $AveLen(t)$ can be used to characterize the degree of local synchronization.

Definition 5 [4]. The cluster order parameter $r_c$characterizing the degree of local synchronization is defined as:

$$r_c = \frac{1}{n} \sum_{i=1}^{n} \sum_{Y \in \delta(X)} e^{-dis(X,Y)}. \tag{5}$$

Definition 6 The Vicsek model [42] for clustering is defined as:

Point $X = (x_1, x_2, \ldots, x_d)$ is a vector in d-dimensional Euclidean space. If point X is regarded as an agent according to the Vicsek model, with an interaction in the δ near neighbor point set $\delta(X)$, then the dynamics of point X over time is described by:

$$X(t+1) = X(t) + \frac{X(t) + \sum_{Y \in \delta(X(t))} Y}{\left\| X(t) + \sum_{Y \in \delta(X(t))} Y \right\|} \cdot v(t) \cdot \Delta t, \tag{6}$$

where $X(t = 0) = (x_1(0), x_2(0), \ldots, x_d(0))$ represents the original location of point X, $X(t+1)$ describes the renewal location of point X at the t step evolution, $v(t)$ is the move velocity at the t step evolution, and $v(t) \cdot \Delta t$ is the move length at the t step evolution.

A special case of this original version of Vicsek model is that if the δ near neighbor point of one point is null, then this point will move along its vector direction.

In some multi-agent systems based on Vicsek model, $v(t)$ is a constant. If $v(t)$ is always a constant, maybe (6) cannot be used for clustering. In a simulation using the data set of Fig. 1, we find that the original version of Vicsek model based on (6) cannot work well for clustering when $v(t)$ is a constant. So we present another effective version of Vicsek model for clustering.

Definition 7 A linear version of Vicsek model for clustering is defined as:

Point $X = (x_1, x_2, \ldots, x_d)$ is a vector in d-dimensional Euclidean space. If point X is regarded as an agent according to 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. [19] and Wang et al. [44] is described by:

$$X(t+1) = \frac{1}{(1+|\delta(X(t))|)}\left(X(t) + \sum_{Y\in\delta(X(t))} Y\right), \quad (7)$$

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

From (7), which is similar to the searching equation of next location in Mean Shift clustering algorithm [9, 17], we can see that the renewal location of point X is the mean location of point X and other points in its δ near neighbor point set $\delta(X)$.

Equation (7) can also be rewritten by:

$$\begin{aligned} X\ (t+1) &= X(t) + \sum_{Y\in\delta(X(t))} (Y - X(t+1)) \\ &= X(t) + \frac{1}{(1+|\delta(X(t))|)}\left(\sum_{Y\in\delta(X(t))} (Y - X(t))\right). \end{aligned} \quad (8)$$

(8) has some similarity with (2) in form, but they have essential difference. The renewal model of (2) is nonlinear and the renewal model of (7) and (8) is linear.

A special case using this linear Vicsek model is that two points meet the condition that the δ near neighbor point of one point only contains another. After one time synchronization using (7), the two points will move to their middle location. So we think this linear Vicsek model for clustering is consistent with the intuition. Another special case using this linear Vicsek model is that one point meets the condition that its δ near neighbor point set is null. After one time synchronization using (7), this point is still an isolate.

Definition 8 According to [5], the Grid Cell is defined as follows:

Grid cells of the d-dimensional Euclidean space of the data set S can be constructed after partitioned the multi-dimensional Euclidean space using a kind of multidimensional grid partitioning method.

The data structure of grid cell g can be defined as:

$$\begin{aligned} \mathrm{DS(g)} = {} & (\mathrm{Grid\_Label}, \mathrm{Grid\_Coordinates}, \mathrm{Grid\_Location}, \\ & \mathrm{Grid\_Range}, \mathrm{Points\_Number}, \mathrm{Points\_Set}). \end{aligned} \quad (9)$$

In (9),

Grid_Label is the key label of the grid cell.

Grid_Coordinates is the coordinates of the grid cell. It is a $d-$ dimensional integer vector expressed by $I = (i_1, i_2, \ldots, i_d)$ that can help to construct δ near neighbor grid cell set more quickly.

Grid_Location is the center location of the grid cell. It is a $d-$dimensional vector expressed by $P = (p_1, p_2, \ldots, p_d)$.

Grid_Range records the region of the grid cell. It is a $d-$dimensional interval vector expressed by:

$$R = ([p_1 - r_1/2, p_1 + r_1/2), \ldots, [p_d - r_d/2, p_d + r_d/2)), \quad (10)$$

where $r_i (i = 1, 2, \ldots, d)$ is the interval length in the i-th dimension of the grid cell.

Points_Number records the number of points in the grid cell.

Points_Set records the labels of all points in the grid cell.

In FSynC algorithm [6] and IESynC algorithm, we use a Red-Black tree that has efficient inserting and deleting operations to record the labels of all points in the grid cell. In this paper, Grid cells and Red-Black trees are used to decrease the time cost of ESynC algorithm.

Definition 9 [5]. Suppose there is a set of N grid cells $Grid(S) = \{g_1, g_2, \ldots, g_N\}$ in the d-dimensional Euclidean space of the data set S, then the δ near neighbor grid cell set $\delta(g_i)$ of grid cell $g_i (i = 1, 2, \ldots, N)$ is defined as:

$$\begin{aligned} \delta(g_i) = {} & \{g_j\, j(\exists P)(\exists Q)(0 < dis(P, Q) \le \delta), \\ & P \in g_i, Q \in g_j, g_j \in Grid(S), g_j \ne g_i\} \end{aligned} \quad (11)$$

The construction details of δ near neighbor grid cell set are described in [5].

Definition 10 The data set $S = \{X_1, X_2, \ldots, X_n\}$ using the linear version of Vicsek model described by (7) for clustering is said to achieve local synchronization, if the final locations of all points satisfy:

$$X_i(t=T) = SL_k(T), i = 1, 2, \ldots, n, k = 1, 2, \ldots, K. \quad (12)$$

In (12), T is the times of the synchronization, K is the number of steady locations in the final synchronization step,

$SL_k(T)$ is the k-th steady location in the final synchronization step.

Usually, the final location of point X_i $(i = 1, 2, \ldots, n)$ depends on the value of parameter δ, the original location of point X_i, and the original locations of other points in the data set S.

Theorem 1 *The data set* $S = \{X_1, X_2, \ldots, X_n\}$ *using the linear version of Vicsek model described by* (7) *for clustering will achieve local synchronization, if parameter* δ *satisfies:*

$$\delta_{\min} \leq \delta \leq \delta_{\max}. \tag{13}$$

Suppose $e_{\min}$ (MST(S)), which is equal to min $\{dis(X_i, X_j) | (X_i, X_j \in S) \wedge (X_i \neq X_j)\}$, is the weight of the minimum edge in the Minimum Span Tree (MST) of the complete graph of the data set S, and $e_{\max}$(MST(S)) is the weight of the maximum edge in the MST of the complete graph of the data set S. Apparently, there is $\delta_{\min} = e_{\min}(\text{MST}(S))$. If the data set S has no isolate, then usually there is $e_{\max}(\text{MST}(S)) \leq \delta_{\max} < \max\{dis(X_i, X_j) | (X_i, X_j \in S) \wedge (X_i \neq X_j)\}$. If the data set S has isolates, we should filtrate all isolates at first.

Proof if $\delta < \delta_{\min}$, then for any point X_i $(i = 1, 2, \ldots, n)$, there is $\delta(X_i) = \emptyset$. In this case, any point in the data set S cannot synchronize with other points, so synchronization will not happen.

In another case, that is $\delta > \delta_{\max}$, then for any point X_i $(i = 1, 2, \ldots, n)$, there is $\delta(X_i(t)) = S - \{X_i(t)\}$. According to (7), there is $X_i(t+1) = \text{mean}(S)$. Here, mean ($S$) is the mean of all points in the data set S. Any point in the data set S will synchronize with all other points, so global synchronization happens. After one time synchronization, all points in the data set S will synchronize to their mean location.

Apparently, if $\delta_{\min} \leq \delta \leq \delta_{\max}$, local synchronization will happen. And the final result of synchronization is affected by the value of parameter δ and the original locations of all points in the data set S. □

Property 1 The data set $S = \{X_1, X_2, \ldots, X_n\}$ using the linear version of Vicsek model described by (7) for clustering will obtain an effective result of local synchronization with some obvious clusters or isolates, if parameter δ satisfies:

$$\begin{aligned} &\max\{longthestEdgeInMst(cluster_k) jk = 1, 2, \ldots, K\} \\ &\leq \delta < \min\{dis(cluster_i, cluster_j) ji, j = 1, 2, \ldots, K\}, \end{aligned} \tag{14}$$

where $longthestEdgeInMst(cluster_k)$ is the weight of the longest edge in the minimum spanning tree of the k-th cluster, $dis(cluster_i, cluster_j)$ is the weight of the minimum edge connecting the i-th cluster and the j-th cluster, and K 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 obvious clusters. If parameter δ is larger than or equal to $\max\{longthestEdgeInMst(cluster_k) | k = 1, 2, \ldots, K\}$, then data points in the same cluster will synchronize. If parameter δ is less than $\min\{dis(cluster_i, cluster_j) | i, j = 1, 2, \ldots, K\}$, then data points in different clusters cannot synchronize. □

4 An effective synchronization clustering algorithm based on a linear version of Vicsek model

ESynC algorithm has almost the same process as SynC algorithm except using a different dynamics clustering model, a linear version of Vicsek model represented by (7).

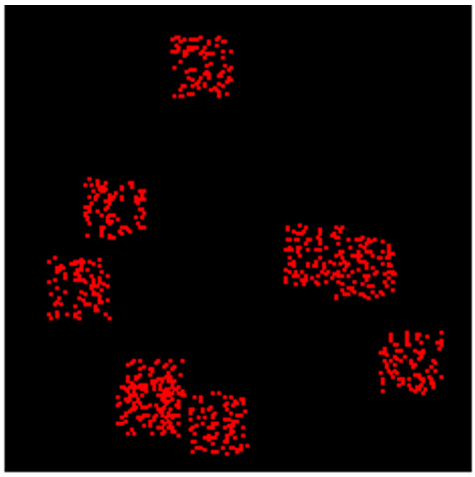
Although we use the Euclidean metric as our dissimilarity measure in this paper, the algorithm is by no means restricted to this metric and this kind of data space. If we can construct a proper dissimilarity measure in a hybrid-attribute space, the algorithm can also be used.

4.1 Compare the extensive Kuramoto model with the linear version of Vicsek model and the original version of Vicsek model

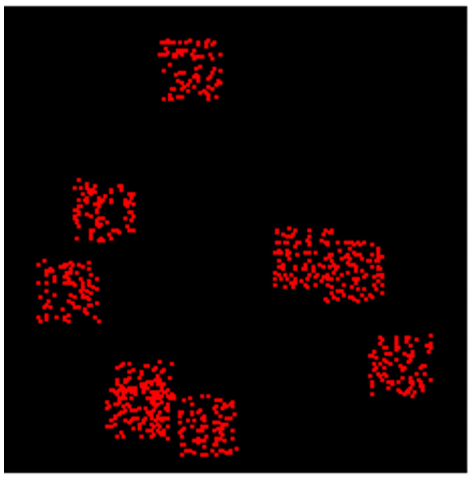
Comparing (2) with (7), we can see that the renewal model of the extensive Kuramoto model at each step evolution is nonlinear and the renewal model of the linear version of Vicsek model at each step evolution is linear.

Figure 1 compares the tracks of 800 data points in the dynamics synchronization clustering processes among the Extensive Kuramoto model (EK model), the Linear version of Vicsek model (LV model), and the Original version of Vicsek model (OV model). Figure 2a compares the cluster order parameter with t-step evolution (t: 0 - 49) among the three models. Figure 2b compares the t-step average length of edges (t: 0 - 49) among the three models. And Fig. 2c compares the relation between the final number of clusters and the value of parameter δ among the three models.

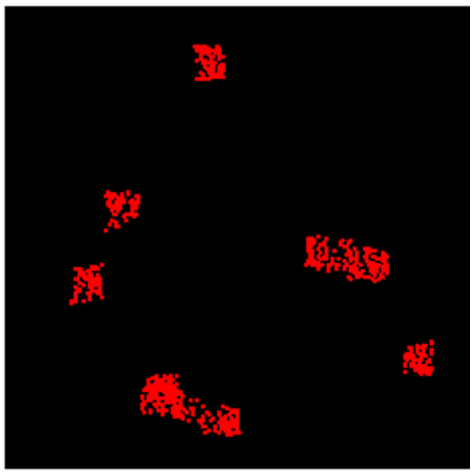
From Fig. 1, we observe that OV model cannot obtain local synchronization effect, and LV model gets better local synchronization effect than EK model. From Fig. 2a and b, we observe that the t-step average length of edges is better than the cluster order parameter with t-step evolution in measuring the synchronization results. From



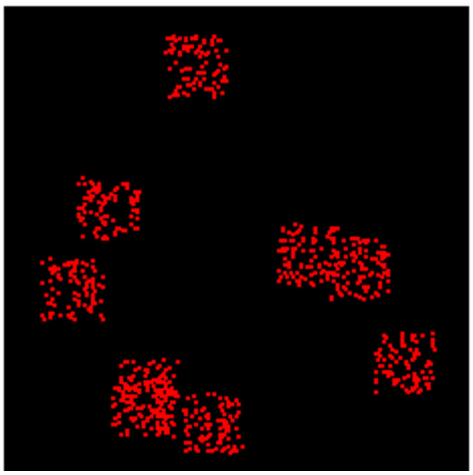
(a) $t = 0$ (The original locations of 800 data points)



(b-1) EKmodel, t=1



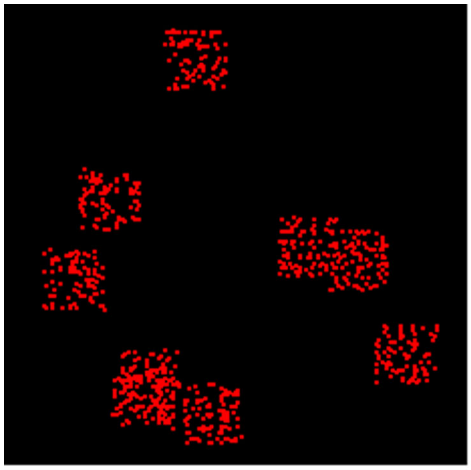
(b-2) LVmodel, $t = 1$



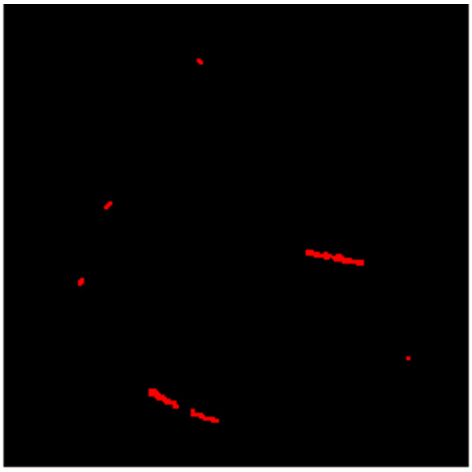
(b-3) OVmodel, $t = 1$

Fig. 1 Compare the dynamical synchronization clustering processes with time evolution among the Extensive Kuramoto model (EK model) the Linear version of Vicsek model (LV model) and the Original version of Vicsek model (OV model). From (**b**) to (**e**) of Fig. 1, parameter δ is set as 18 in the three models

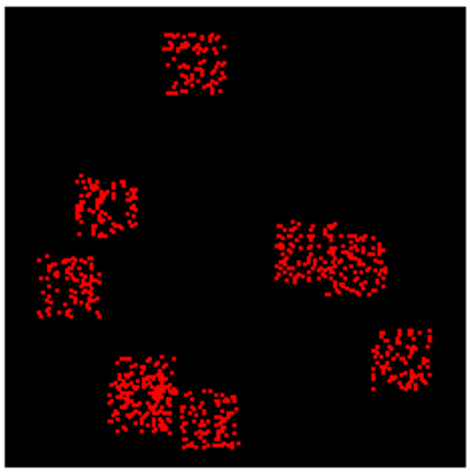
Fig. 1 (continued)



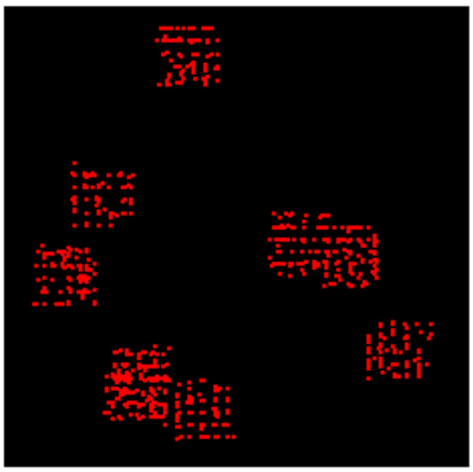
(c-1) EKmodel, t=2



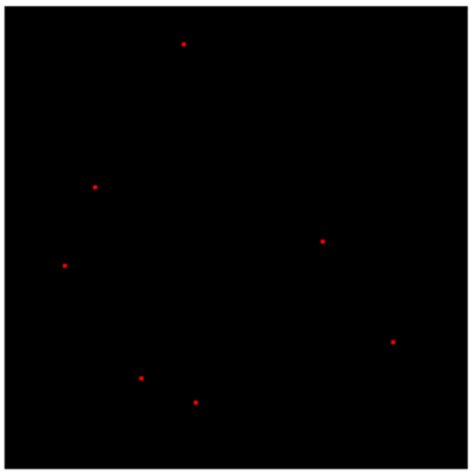
(c-2) LVmodel, t = 2



(c-3) OVmodel, t = 2

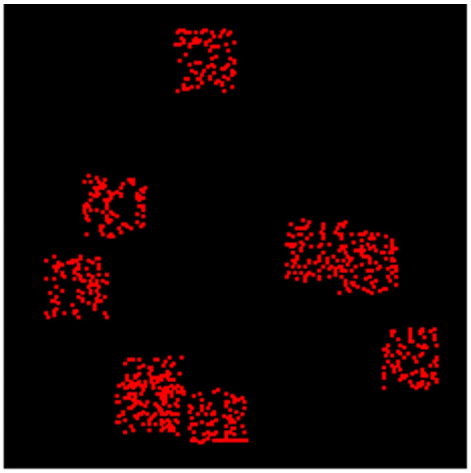


(d-1) EKmodel, t=5

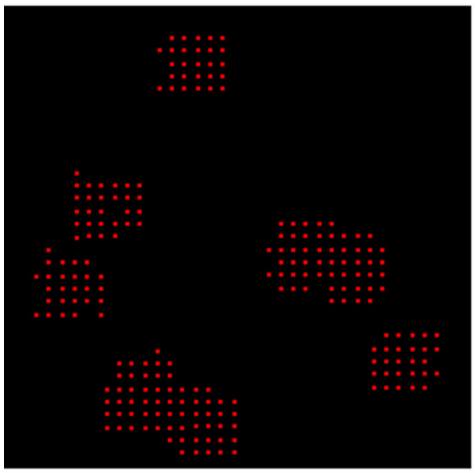


(d-2) LVmodel, t = 5

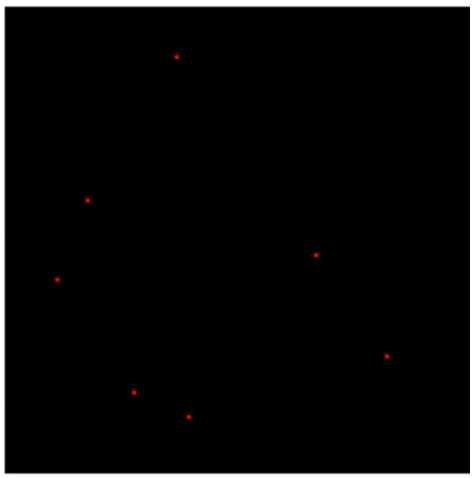
Fig. 1 (continued)



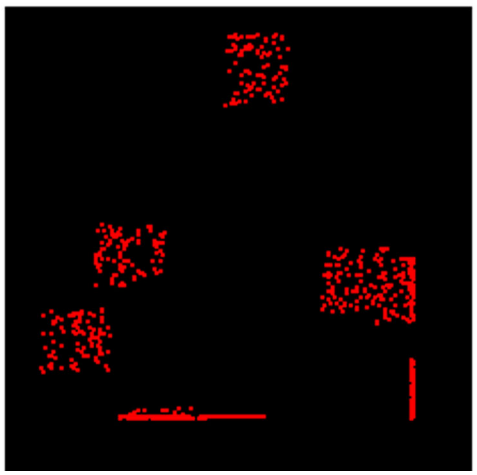
(d-3) OVmodel, $t = 5$



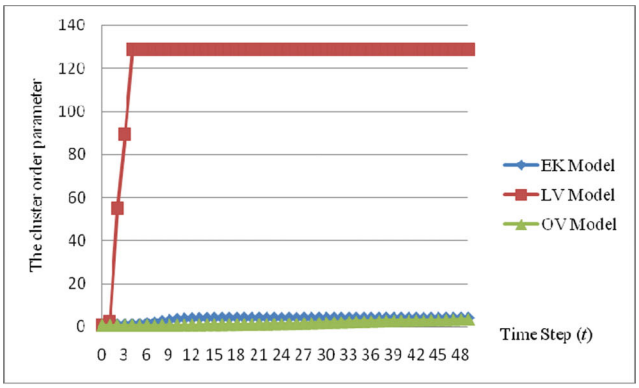
(e-1) EKmodel, t=45



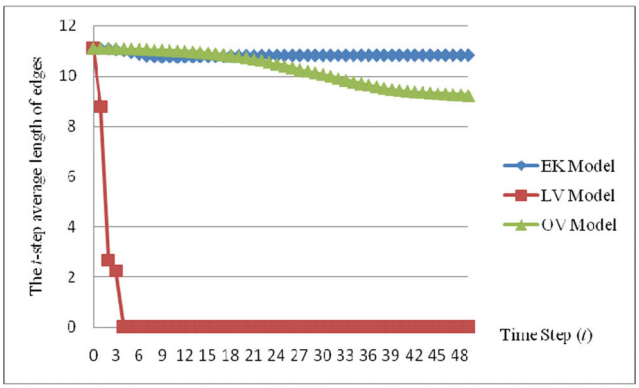
(e-2) LVmodel, $t = 45$



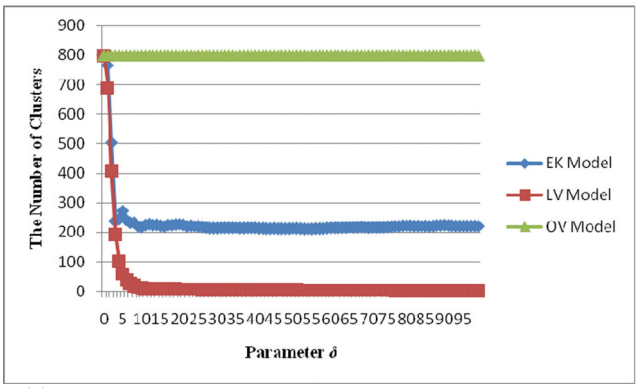
(e-3) OVmodel, $t = 45$



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



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



(c) The relation between the final number of clusters and parameterδ (δ: 0 - 99)

Fig. 2 Compare EK model, LV model, and OV model. In Fig. 2, the data set has 800 points. In Fig. 2 (**a**) and (**b**), parameter δ is set as 18 in the three models

Fig. 2c, we observe that the smaller parameter δ is set in LV model and EK model, the larger the final number of clusters is. For many data sets with obvious clusters, LV model can often get the correct final number of clusters when parameter δ chooses any value in its valid interval, and the final number of clusters using EK model is much larger than the actual number of clusters when parameter δ chooses any value in a long interval. In OV model, the final number of clusters is the same as the number of data points when parameter δ chooses any value in a long interval.

4.2 The description of SynC algorithm

The original synchronization clustering algorithm named as SynC is developed by Böhm et al. [4]. In order to show the difference between SynC algorithm and ESynC algorithm, we introduce it below using our language according to the description of [4].

Algorithm 1 Synchronization clustering (SynC) algorithm.

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 process of SynC algorithm:
1 IterateStep t is set as zero firstly, that is: $t \leftarrow 0$;
2 **for** $(i = 1; i \leq n; i++)$
3 $X_i(t) \leftarrow X_i$;
4 **while** (the dynamical clustering does not satisfy its convergent condition)
5 {
6 **for** $(i = 1; i \leq n; i++)$
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 (1) of Definition 1;
9 Compute the renewal value, $X_i(t+1)$, of $X_i(t)$ using (2) of Definition 2;
10 }
11 Compute the cluster order parameter $r_c$of all points using (5) of Definition 5;
12 IterateStep t is increased by 1, that is: $t++$;
13 **if** (r_c converges or $(t == 50)$)
14 We think the dynamical clustering reaches its convergent result, and then exit from the while repetition;
15 }
16 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 above while repetition. The final convergent set $S(T)$ reflects the natural clusters or isolates of the data set S.

4.3 The description of ESynC algorithm

ESynC algorithm can be described as follows.

Algorithm 2 An effective synchronization clustering (ESynC) algorithm.

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.
Process:
Step1. Initialization:
1 IterativeStep t is set as zero firstly, that is: $t \leftarrow 0$;
2 **for** $(i = 1; i \leq n; i++)$
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 <50))
5 {
6 **for** $(i = 1; i \leq n; i++)$
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 (1) of Definition 1;
9 Compute the renewal value, $X_i(t+1)$, of $X_i(t)$ using (7) of Definition 7;
10 }
11 Compute the t-step average length of edges of all points, $AveLen(t)$, using (4) of Definition 4;
/* We can also compute the cluster order parameter r_c using (5) of Definition 5 instead of computing $AveLen(t)$. */
12 IterativeStep t is increased by 1, that is: $t++$;
13 **if** $(AveLen(t) \rightarrow 0)$ /* $AveLen(t) \rightarrow 0$ is equivalent with $r_c \rightarrow$ the limit of r_c. */
14 We think 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.

4.4 Complexity analysis of ESynC algorithm

Step1 of ESynC algorithm needs Time $= O(n)$ and Space $= O(n)$.

In Step2, constructing the δ near neighbor point sets for all points if using a simple method needs Time $= O(dn^2)$ and Space $= O(nd)$. In constructing the δ near neighbor point sets, time cost can be decreased by using the strategy of "space exchanging time".

Step3 needs Time $= O(n)$ and Space $= O(n)$.

According to [4] and our analysis, ESynC algorithm needs Time $= O(Tdn^2)$, which is also the time complexity of SynC Algorithm. Here T is the times of the while repetition in Step2.

4.5 Setting parameter in ESynC algorithm

Parameter δ in ESynC algorithm can affect the result of clusters or isolates. In [4], parameter δ is optimized by the MDL principle [15]. In [7], two other methods were presented to estimate parameter δ. Here, we can also select a proper value for parameter δ according to Theorem 1 and Property 1.

4.6 The convergence of ESynC algorithm

In all simulations, if using ESynC algorithm to synchronize the original data set S, then $S(T) = \{X_1(T), X_2(T), \ldots, X_n(T)\}$ will stay on some locations steadily after several iterations (many simulations only need 4 - 5 times). In the final convergent set $S(T)$, a steady location that represents some points can be regarded as their cluster center, and a steady location that represents only one or several points is regarded as the final synchronization location of one or several isolates.

The renewal computing equation of ESynC algorithm can be represented by a matrix formula,

$$\boldsymbol{S}(t+1) = \boldsymbol{A}(t) \cdot \boldsymbol{S}(t). \tag{15}$$

In (15), $\boldsymbol{S}(t) = (X_1(t), X_2(t), \ldots, X_n(t))^T$ is the t-step location vector of n points, $\{X_1, X_2, \ldots, X_n\}$, and $\boldsymbol{A}(t)$ is an $n \times n$ matrix.

Suppose $\boldsymbol{S}(t = 0)$ is arranged by the final cluster order, which means that those points in the same cluster are coterminous in $\boldsymbol{S}(t = 0)$. In this case, $\boldsymbol{A}(T)$ will be a block matrix. For example, suppose the data set S has K final clusters or isolates, then $\boldsymbol{A}(T)$ of (1) is just a block matrix.

$$A(T) = \begin{pmatrix} 1/|c_1| & \ldots & 1/|c_1| & & & & & & \\ \ldots & \ldots & \ldots & & 0 & & \ldots & & 0 \\ 1/|c_1| & \ldots & 1/|c_1| & & & & & & \\ & & & 1/|c_2| & \ldots & 1/|c_2| & & & \\ 0 & & & \ldots & \ldots & \ldots & 0 & & 0 \\ & & & 1/|c_2| & \ldots & 1/|c_2| & & & \\ 0 & & & 0 & & \ldots & & 0 & \\ & & & & & & 1/|c_K| & \ldots & 1/|c_K| \\ 0 & & & 0 & & \ldots & \ldots & \ldots & \ldots \\ & & & & & & 1/|c_K| & \ldots & 1/|c_K| \end{pmatrix} \tag{16}$$

In (1), $|c_k|$ labels the number of all points in the k-th cluster or isolates.

Perhaps analyzing the synchronization process based on the linear version of Vicsek model from theory is more complex than analyzing Mean Shift clustering algorithm [7, 14] or multi-agent systems based on Vicsek model.

4.7 The improvement of ESynC algorithm

An improved version of ESynC algorithm, named as IESynC algorithm (Its description is presented in Appendix 1 of Supplementary material), is designed by combining multidimensional grid partitioning method and Red-Black tree structure to construct the near neighbor point sets of all points. The improving method that can decrease the time cost of constructing near neighbor point set is introduced in [6]. Here, we describe this method as possible as simply. Generally, we first partition the data space of the data set $S = \{X_1, X_2, \ldots, X_n\}$ using a kind of multidimensional grid partitioning method. Then we can obtain a set of all grid cells. Last design an effective index of all grid cells and constructing the δ near neighbor grid cell set for each grid cell. If every grid cell uses a Red-Black tree to index its data points in each synchronization step, then constructing the δ near neighbor point set for every point will become quicker when the number of grid cells is proper.

Another improved version of ESynC algorithm in both time cost and space cost is described in another paper.

5 Simulated experiments

5.1 Experimental design

Our experiments are finished in a personal computer (Capability Parameters: Pentium(R) Dual-Core CPU E5400 2.7GHz, 2G Memory). Experimental programs are developed using C and C++ language under Visual C++6.0 of Windows XP.

To verify the improvement in clustering effect and time cost of ESynC algorithm and IESynC algorithm, there will be some simulations of some artificial data sets, eight UCI data sets (Frank et al., 2010), and three bmp pictures in Sections 5.2, 5.3, and 5.4.

Four kinds of artificial data sets (DS1 - DS4) are produced in a 2-D region [0, 600] × [0, 600] by a program presented in Appendix 2 of Supplementary material. Eight kinds of artificial data sets (DS5 - DS12) are produced in a range [0, 600] in each dimension by a similar program. Iris et al. [12] are eight UCI data sets that used in our experiments. Three bmp pictures (named as Picture1, Picture2, and Picture3) are obtained from Internet. The description of the experimental data sets is presented in Table 1 of Appendix 3 of Supplementary material.

In Section 5.2, ESynC algorithm and IESynC algorithm will be compared with SynC algorithm and some other classic clustering algorithms (K-Means [27], FCM [3], AP [13], DBSCAN [11], Mean Shift [9, 14]) in clustering quality and time cost using some artificial data sets.

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

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

In the experiments, parameterr_i $(i = 1, 2, \ldots, d)$ used in IESynC algorithm is the interval length in the i-th dimension of grid cell [5], and parameter δ used in SynC, ESynC, IESynC, DBSCAN, and Mean Shift is the threshold of Definition 1. In DBSCAN algorithm, parameter $MinPts = 4$, and parameter Eps is the same as parameter δ.

The detailed discussion on how to construct grid cells and δ near neighbor point sets is described in [5]. How to select a proper value for parameter δ of SynC algorithm is discussed in [4]. ESynC algorithm can use the same method as SynC algorithm to select a proper value for parameter δ. In IESynC algorithm, setting different values for parameter r_i $(i = 1, 2, \ldots, d)$ will result in different number of grid cells and different time cost.

In our simulated experiments, the maximum times of synchronization moving in the while repetition of SynC algorithm, ESynC algorithm, and IESynC algorithm is set as 50.

Comparison results of these algorithms are presented by some figures (Figs. 3, 4, 5, Figs. 4–6 of Appendix 3 of Supplementary material) and some tables (Tables 1, 2, 3, 4, 5, 6). And performance of algorithms is measured by time cost (second). Clustering quality of algorithms is measured by display figures and two robust information-theoretic measures, Adjusted Mutual Information (AMI) [43] and Normalized Mutual Information (NMI) [40]. According to the opinions of [43], the higher the value of two measures gets, the better the clustering quality of algorithm is. In simulations, we use the Matlab code from [43] to compute the two clustering quality measures.

5.2 Experimental results of some artificial data sets (DS1 - DS12)

5.2.1 Comparison results among SynC algorithm, ESynC algorithm, and IESynC algorithm

Table 1 presents the comparison results of three synchronization clustering algorithms (SynC, ESynC, and IESynC)

Table 1 Compare three synchronization clustering algorithms (SynC, ESynC, and IESynC) using four kinds of artificial data sets (DS1 - DS4). In Table 1, parameter $\delta = 18$, the number of data points $n = 10000$. In IESynC of Table 1, $r_1 = r_2 = 10$

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS1	DS2	DS3	DS4
Spend time (second)	SynC	448	553	538	525
	ESynC	**56**	**70**	**107**	**81**
	IESynC	**56**	**66**	**75**	**55**
Iterative times	SynC	41	50	50	50
	ESynC, IESynC	**4**	**5**	**8**	**6**
The number of steady locations	SynC	254	379	260	431
	ESynC, IESynC	**14**	**5**	**25**	**8**
The cluster order parameter r_c	SynC	42.7595	29.1156	44.8862	25.0379
	ESynC, IESynC	**1995.40**	**1999.00**	**1352.31**	**1356.98**
AveLen(T)	SynC	11.0132	11.1481	10.6701	11.0542
	ESynC, IESynC	**0**	**0**	**0**	**0**

Note1: *AveLen*(T) is the final average length of edges in the final δ near neighbor undirected graph. Here, T is equal to the iterative times of synchronization clustering algorithm.

Note2: The bold in Table 1 marks the improved results.

using four kinds of artificial data sets (DS1 - DS4). In Table 1, by intercomparing SynC, ESynC, and IESynC, we observe that IESynC is the fastest clustering algorithm. At the same time, ESynC and IESynC can get better local synchronization results than SynC in the four data sets.

5.2.2 Comparison results among SynC algorithm, ESynC algorithm, IESynC algorithm, and some classical clustering algorithms

Table 2 presents the clustering quality of several clustering algorithms (SynC, ESynC, IESynC, and some classical clustering algorithms) using six kinds of artificial data sets (DS2, DS4, DS5, DS6, DS7, and DS8). When computing two information-theoretic measures, NMI and AMI, the predefined cluster labels of the eight artificial data sets are used in true_mem that is an input file of the MATLAB code [43]. In Table 2, by intercomparing SynC, ESynC, IESynC, and some classical clustering algorithms, we observe that ESynC and IESynC can get acceptable clustering results in the eight artificial data sets. Because the three artificial data sets (DS4 ($n = 400$), DS4 ($n = 800$), and DS5 ($n = 10000$)) have two connected clusters, ESynC and IESynC do not get the largest values of NMI and AMI.

Figures 3 and 4–6 of Appendix 3 of Supplementary material present the comparison clustering results of several clustering algorithms by some display figures that can reflect the clustering quality clearly. In Figs. 3 and 4–6 of Appendix 3 of Supplementary material, parameter $\delta = 18$ in SynC, ESynC, IESynC, DBSCAN, and Mean Shift; the number of data points $n = 400$.

From Figs. 3 and 4–6 of Appendix 3 of Supplementary material, we observe that ESynC and IESynC can get better clustering quality (obvious clusters or isolates displayed by figures) than SynC, AP, K-Means, and FCM in some artificial data sets (from DS1 - DS4). Mean Shift, DBSCAN can obtain similar clustering quality (obvious clusters displayed by figures) with ESynC and IESynC in some artificial data sets (from DS1 - DS4). Especially, SynC, ESynC, and IESynC can all easily find some isolates if setting a proper value for parameter δ.

Figure 4 presents the comparison results of several clustering algorithms in time cost. In Fig. 4, parameter $\delta = 18$ in SynC, ESynC, IESynC, DBSCAN, and Mean Shift; the number of data points $n = 20000$. In IESynC, $r_1 = r_2 = 6$.

In Fig. 4, Intercomparing ESynC, IESynC, Mean Shift, DBSCAN, FCM, and K-Means, we observe that IESynC is faster than ESynC with the same clusters, and K-Means is the fastest clustering algorithm.

5.2.3 Comparing the valid interval of parameter δ among SynC, ESynC, IESynC, DBSCAN, and Mean Shift using some artificial data sets (DS5 - DS8)

Here we compare the valid interval of parameter δ among SynC algorithm, ESynC algorithm, IESynC algorithm, DBSCAN algorithm, and Mean Shift algorithm.

Table 2 Compare the clustering quality of several clustering algorithms (SynC, ESynC, IESynC, and some classical clustering algorithms) using six kinds of artificial data sets (DS2, DS4, DS5, DS6, DS7, and DS8). In Table 2, parameter $\delta = 18$ in DS2, DS4, DS5, and DS6; parameter $\delta = 30$ in DS7 and DS8

Measure indexes of algorithms	Name of algorithms	Data sets			
		DS2 ($n = 400$)	DS4 ($n = 400$)	DS2 ($n = 800$)	DS4 ($n = 800$)
NMI	ESynC, IESynC	**1.0000**	0.9694	**1.0000**	0.9643
	SynC	0.5505	0.6324	0.5362	0.6099
	K-Means	0.8670	0.9185	0.8659	**0.9682**
	FCM	**1.0000**	0.9633	**1.0000**	0.9615
	AP	0.7966	**0.9697**	0.7355	0.8375
	DBSCAN	**1.0000**	0.9643	**1.0000**	0.9643
	Mean Shift	0.7978	0.9028	0.7799	0.9103
AMI	ESynC, IESynC	**1.0000**	0.9682	**1.0000**	0.9286
	SynC	0.1237	0.1275	0.1653	0.1785
	K-Means	0.8255	0.8980	0.8266	**0.9676**
	FCM	**1.0000**	0.9616	**1.0000**	0.9603
	AP	0.6252	**0.9684**	0.5333	0.7157
	DBSCAN	**1.0000**	0.9274	**1.0000**	0.9286
	Mean Shift	0.6251	0.8268	0.6022	0.8758
The number of clusters	ESynC, IESynC	5	9	5	8
	SynC	227	255	314	357
	K-Means	**5** (predefined)	**9** (predefined)	**5** (predefined)	**9** (predefined)
	FCM	5 (predefined)	9 (predefined)	5 (predefined)	9 (predefined)
	AP	13	**9**	20	19
	DBSCAN	**5**	**8**	**5**	**8**
	Mean Shift	15	15	17	14
		DS5 ($n = 10000$)	DS6 ($n = 10000$)	DS7 ($n = 10000$)	DS8 ($n = 10000$)
NMI	ESynC, IESynC	0.9765	**1.0000**	**1.0000**	**1.0000**
	SynC	0.6231	0.5411	0.5205	0.5194
	K-Means	0.8872	NaN (In Matlab)	0.9194	0.8437
	FCM	**0.9788**	0.5228	0.5226	0.5282
	DBSCAN	0.9765	**1.0000**	**1.0000**	**1.0000**
	Mean Shift	0.9708	**1.0000**	**1.0000**	**1.0000**
AMI	ESynC, IESynC	0.9534	**1.0000**	**1.0000**	**1.0000**
	SynC	0.3539	0.0973	0.0051	1.5118e-04
	K-Means	0.8426	NaN (Matlab)	0.8892	0.7783
	FCM	**0.9781**	0.5228	0.5226	0.2788
	DBSCAN	0.9534	**1.0000**	**1.0000**	**1.0000**
	Mean Shift	0.9534	**1.0000**	**1.0000**	**1.0000**
The number of clusters	ESynC, IESynC	**11**	**12**	**12**	**12**
	SynC	578	5577	9729	9992
	K-Means	12 (predefined)	1 (+11 null clusters)	12 (predefined)	12 (predefined)
	FCM	12 (predefined)	2 (+10 null clusters)	3 (+9 null clusters)	2 (+10 null clusters)
	DBSCAN	**11**	**12**	**12**	**12**
	Mean Shift	**12**	**12**	**12**	**12**

Note: NMI and AMI are two clustering quality measures presented in [40, 43]. In Table 2, the largest values of NMI and AMI and acceptable number of clusters in every data set are shown in bold.

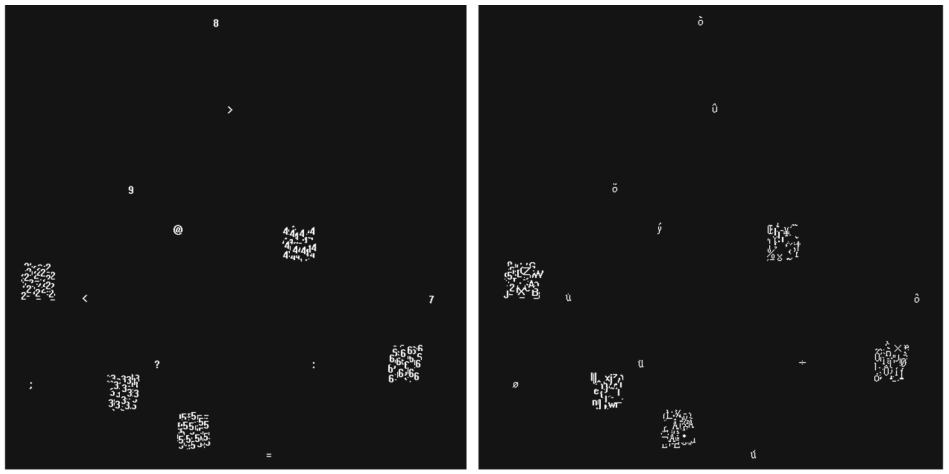
Table 3 is the comparison results among these clustering algorithms. Here, $[e_k , e_{k+1}]$ can be obtained from (8) of [7]. In Table 3, intercomparing SynC, ESynC, IESynC, DBSCAN, and Mean Shift, we observe that the valid interval of parameter δ in ESynC and IESynC is longer than that in DBSCAN, the valid interval of parameter δ in DBSCAN is consistent with $[e_k , e_{k+1}]$, and parameter δ in Mean Shift has the largest valid interval.

5.3 Experimental results of eight UCI data sets

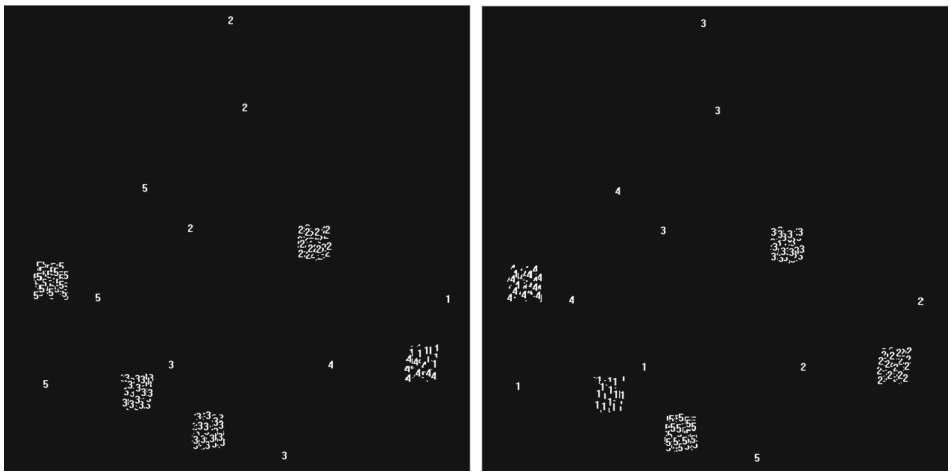
Because we do not know the true dissimilarity measure of these UCI data sets, all points of these UCI data sets are standardized into a range [0, 600] in each dimension in this experiment. When computing two information-theoretic measures (NMI and AMI), because we do not know the true cluster labels 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 [43].

5.3.1 Comparison results between SynC algorithm and ESynC algorithm

Table 4 gives the comparison results of two synchronization clustering algorithms (SynC and ESynC) using eight UCI data sets. In Table 4, by comparing SynC with ESynC, we observe that ESynC can get better local synchronization results, less synchronization times, and less clustering time than SynC in the eight UCI data sets.



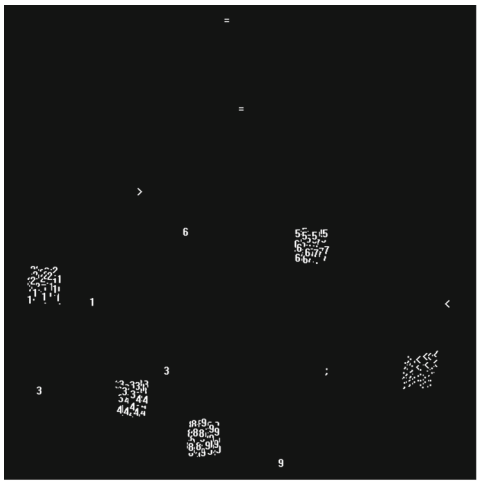
(a) Clusters identified by ESynC and IESynC (15 clusters or isolates) (b) Clusters identified by SynC (204 clusters or isolates)



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

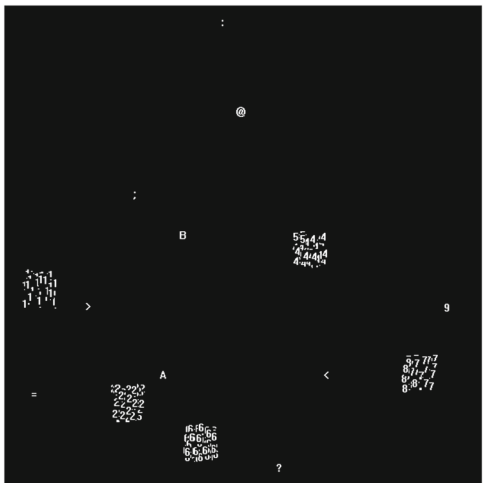
Fig. 3 Compare the clustering results of several algorithms (DS1, $n = 400$)

Fig. 3 (continued)



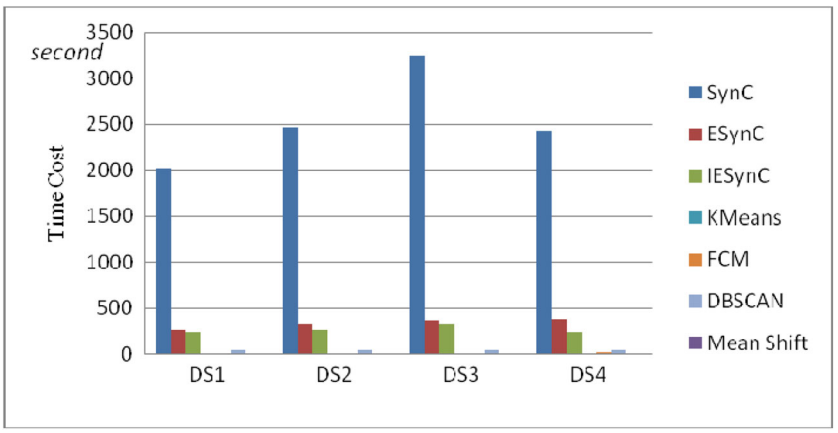
(e) Clusters identified by AP (14 clusters)

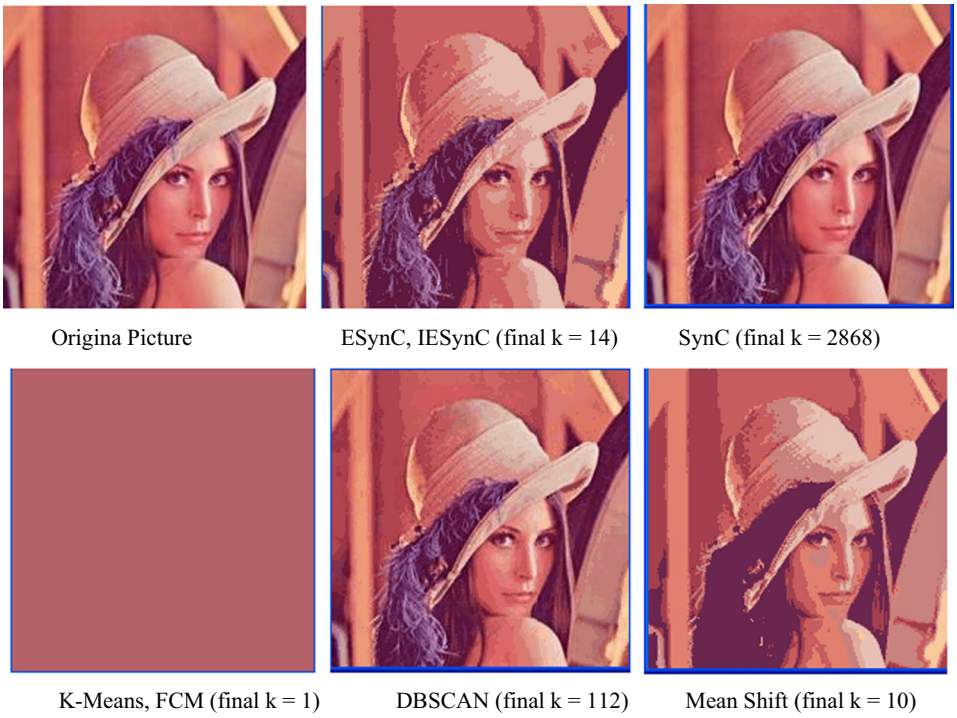
(f) Clusters identified by DBSCAN (5 clusters)



(g) Clusters identified by Mean Shift (18 clusters)

Fig. 4 Compare several algorithms in time cost using four artificial data sets (DS1 - DS4, $n = 20000$)





(a) δ = 18 in SynC, ESynC, IESynC, DBSCAN, and Mean Shift; predefined k (number of clusters) = 14 in K-Means and FCM.

(b) δ = 30 in SynC, ESynC, IESynC, DBSCAN, and Mean Shift; predefined k (number of clusters) = 6 in K-Means and FCM.

Fig. 5 Compare the original picture and several compressed pictures of Picture3. In Fig. 5, several compressed pictures based on different clustering algorithms are drawn by using the means of clusters that are obtained by clustering the 200 * 200 pixel points of Picture3 in RGB space

Table 3 Compare the valid interval of parameter δ among SynC, ESynC, IESynC, DBSCAN, and Mean Shift using four artificial data sets with different dimensions. In Table 3, $n = 10000$

		Data sets			
		DS5	DS6	DS7	DS8
The valid interval of parameter δ	SynC	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$
	ESynC, IESynC	$\delta \in [9, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	DBSCAN	$\delta \in [2, 45]$	$\delta \in [7, 147]$	$\delta \in [12, 199]$	$\delta \in [17, 281]$
	Mean Shift	$\delta \in [15, 60]$	$\delta \in [17, 176]$	$\delta \in [20, 285]$	$\delta \in [22, 396]$
$[e_k, e_{k+1}]$ in an MST of the complete graph of the data set		[2.16, 45.42]	[9.82, 147.48]	[15.29, 199.78]	[21.04, 281.19]

Table 4 Compare two synchronization clustering algorithms (SynC and ESynC) using eight UCI data sets

(a) The setting of parameter δ in two synchronization clustering algorithms

UCI data sets	Parameter δ in SynC and ESynC
Iris	120
Wine	305
Wdbc	345
Glass	148
Ionosphere	615
Letter-recognition	210
Segmentation	205
Cloud	380

(b) Comparison results of the first four UCI data sets

Measure indexes of algorithms	Name of algorithms	Data sets			
		Iris	Wine	Wdbc	Glass
Spend time (second)	SynC	0	0	15	0
	ESynC	0	0	2	0
Iterative times	SynC	50	50	50	50
	ESynC	**9**	**6**	**7**	**6**
The number of steady locations	SynC	147	178	569	213
	ESynC	**5**	**19**	**35**	**35**
The cluster order parameter r_c	SynC	0.05333	0	0	0.009346
	ESynC	**54.1067**	**47.8876**	**305.3497**	**55.1402**
AveLen(T)	SynC	83.9640	258.3664	276.6775	97.9706
	ESynC	**0**	**0**	**0**	**0**

(c) Comparison results of the next four UCI data sets

Measure indexes of algorithms	Name of algorithms	Data sets			
		Ionosphere	Letter-recognition	Segmentation	Cloud
Spend time (second)	SynC	5	6614	**1**	79
	ESynC	1	5359	**0**	**10**
Iterative times	SynC	50	50	50	50
	ESynC	**9**	**23**	**7**	**6**
The number of steady locations	SynC	350	18668	210	2043
	ESynC	**85**	**34**	**38**	**2**
The cluster order parameter r_c	SynC	0.005698	0.2596	0.000036	0.004965
	ESynC	**126.49**	**9107.0009**	**19.5905**	**1023**
AveLen(T)	SynC	401.6912	171.9401	142.6595	215.9900
	ESynC	**0**	**0**	**0**	**0**

Note: The bold in Table 4 marks the better results of ESynC algorithm.

Table 5 Compare the clustering quality of several clustering algorithms (SynC, ESynC, and some classical clustering algorithms) using eight UCI data sets

(a) The setting of parameter δ in several clustering algorithms

UCI data sets	Parameter δ in SynC, ESynC, and SSynC	Parameter δ in DBSCAN	Parameter δ in Mean Shift
Iris	120	75	150
Wine	305	242.725	305
Wdbc	345	215	345
Glass	148	80	120
Ionosphere	615	350	710
Letter-recognition	210	160	220
Segmentation	205	176	270
Cloud	380	350	350

(b) Comparison results of the first four UCI data sets

Measure indexes of algorithms	Name of algorithms	Data sets			
		Iris	Wine	Wdbc	Glass
NMI	ESynC	0.7265	0.7615	0.4655	0.4540
	SynC	0.4697	0.4578	0.3226	**0.5306**
	K-Means	0.7145	**0.8782**	**0.6232**	0.3588
	FCM	**0.7919**	0.4823	0.5947	0.4108
	AP	0.6061	0.5382	0.3594	0.4257
	DBSCAN	0.6465	0.3534	0.2904	0.2574
	Mean Shift	0.7265	0.7612	0.2797	0.4662
AMI	ESynC	0.7143	0.6057	0.3513	0.2872
	SynC	0.0050.	3.2528e-16	6.8369e-16	0.0012
	K-Means	0.7107	**0.8735**	**0.6110**	**0.3265**
	FCM	**0.7888**	0.3820	0.5887	0.2525
	AP	0.3982	0.2977	0.1453	0.2423
	DBSCAN	0.5712	0.3423	0.2496	0.2065
	Mean Shift	0.7143	0.5819	0.2086	0.2414
The number of clusters	ESynC	3 (+ 2 isolates)	3 (+ 16 isolates)	2 (+ 33 isolates)	6 (+29 isolates)
	SynC	2 (+ 145 isolates)	0 (+178 isolates)	0 (+ 569 isolates)	1 (+ 212 isolates)
	K-Means	3 (predefined)	3 (predefined)	2 (predefined)	6 (predefined)
	FCM	3 (predefined)	3 (predefined) Final: 2 (+1 null cluster)	2 (predefined)	6 (predefined) Final: 2 (+ 4 null clusters)
	AP	11	21	36 (+ 9 isolates)	12 (+ 14 isolates)
	DBSCAN	3 (+ 35 isolates)	3 (+ 75 isolates)	2 (+ 194 isolates)	6 (+ 83 isolates)
	Mean Shift	3 (+ 2 isolates)	3 (+ 18 isolates)	2 (+33 isolates +1 null clusters)	6 (+ 43 isolates)

Table 5 (continued)

(c) Comparison results of the next four UCI data sets

Measure indexes of algorithms	Name of algorithms	Data sets			
		Ionosphere	Letter-recognition	Segmentation	Cloud
NMI	ESynC	0.3106	0.3986	0.6086	**1**
	SynC	0.3339	**0.5768**	0.6033	0.3016
	K-Means	0.1299	0.3572	0.6103	0.9944
	FCM	0.1264	0.0095	0.4454	0.9944
	AP	0.2809	-	0.6781	0.4107
	DBSCAN	**0.4061**	0.1517	0.4592	**1**
	Mean Shift	0.2831	0.3649	0.6447	**1**
AMI	ESynC	0.1073	**0.3986**	0.4212	**1**
	SynC	3.5016e-04	0.0166	-1.6974e-15	2.4432e-04
	K-Means	0.1246	0.3484	**0.5286**	0.9944
	FCM	0.1211	0.0042	0.2574	0.9944
	AP	0.1002	-	0.4897	0.1653
	DBSCAN	**0.3417**	0.1517	0.4016	**1**
	Mean Shift	0.0991	0.3649	0.5048	**1**
The number of clusters	ESynC	2 (+ 83 isolates)	26 (+ 8 isolates)	7 (+ 31 isolates)	2
	SynC	0 (+ 350 isolates)	845 (+ 17823 isolates)	0 (+ 210 isolates)	5 (+ 2038 isolates)
	K-Means	2 (predefined)	26 (predefined)	7 (predefined)	2 (predefined)
	FCM	2 (predefined)	26 (predefined) Final: 2 (+ 24 null clusters)	7 (predefined) Final: 2 (+ 5 null clusters)	2 (predefined)
	AP	14 (+ 44 isolates)	-	17 (+ 7 isolates)	66 (+ 1 isolate)
	DBSCAN	2 (+ 145 isolates)	28 (+ 323 isolates)	7 (+ 51 isolates)	2
	Mean Shift	2 (+ 76 isolates)	26 (+ 3 isolates + 1 null cluster)	7 (+ 22 isolates)	2

Note1: In 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 AP algorithm column means that the time cost is too larger.

Note2: In Table 5, the largest values of NMI and AMI in every data set are shown in bold.

5.3.2 Comparison results among SynC algorithm, ESynC algorithm, and some classical clustering algorithms

Table 5 gives the comparison clustering quality of several clustering algorithms (SynC, ESynC, and some classical clustering algorithms) using eight UCI data sets. In Table 5, by intercomparing these clustering algorithms, we observe that ESynC algorithm does not get the largest values of NMI and AMI except Cloud data set. We think there are three reasons. First, we use the Euclidean metric to compute the dissimilarity measure of the eight UCI data sets without any actual knowledge on these data sets. Second, we observe that the largest values of NMI and AMI in some data sets do not mean the best clustering quality. Third, the class labels of these UCI data sets, which are not often consistent with the actual distributions of clusters, are used as the benchmark of clusters in our simulations (Because we don't have better choice). From the view of the final number of clusters of Table 5, we observe that ESynC algorithm can get better clustering results than some other clustering algorithms in some UCI data sets.

5.4 Experimental results of three bmp pictures

The value in RGB (Red, Green, and Blue) color space of pixel points is in a range [0, 255] in each dimension. In IESynC algorithm of Table 6 and Fig. 5, parameter $r_i (i = 1, 2, 3)$ is set as 6.

5.4.1 Comparison results among SynC algorithm, ESynC algorithm, and IESynC algorithm

Table 6 presents the experimental results in time cost and local synchronization results among SynC algorithm, ESynC algorithm, and IESynC algorithm. The data sets in Table 6 are pixel points of three bmp pictures. In Table 6, by intercomparing SynC algorithm, ESynC algorithm, and IESynC algorithm, we observe that ESynC and IESynC are faster than SynC in these data sets. At the same time, ESynC algorithm and IESynC algorithm can get better local synchronization results than SynC algorithm in these data sets.

5.4.2 Comparison results among SynC algorithm, ESynC algorithm, IESynC algorithm, and some classical clustering algorithms

Figure 5 lists the original picture and several compressed pictures of Picture3. The several compressed pictures are drawn by using the means of clusters that are 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 did not use it. From Fig. 5, we observe that ESynC algorithm and IESynC algorithm can get multi-level clustering compressed effect for different values of parameter δ.

5.5 Analysis and conclusions of experimental results

From the comparison experimental results of these figures and tables (Figs. 1, 2, 3, 4, 5, Figs. 4–6 of Appendix 3 of Supplementary material, and Tables 1, 2, 3, 4, 5, 6), we observe that ESynC algorithm based on the linear version of Vicsek model not only gets better local synchronization effect but also needs less iterative times and time cost than SynC algorithm based on an extensive Kuramoto model almost in all experimental data sets. We think that ESynC algorithm is superior to SynC algorithm for clustering because of its more proper synchronization model.

From the simulations, we observe that IESynC algorithm is faster than ESynC algorithm in some cases. Usually, IESynC algorithm spends less time in some kinds of data sets by selecting proper values for parameterr_i $(i = 1, 2, \ldots, d)$. The time cost of IESynC algorithm is sensitive to parameter r_i $(i = 1, 2, \ldots, d)$. If the number of grid cells is too small or too large, we find that IESynC algorithm cannot obtain obvious improvement of time cost in many cases. So we say IESynC algorithm is a parametric algorithm.

From simulations of four data sets (from DS5 - DS8), we observe that the effective interval of parameter δ in ESynC algorithm and IESynC algorithm is longer than that in SynC algorithm and DBSCAN algorithm.

In some displayed figures, by intercomparing SynC, ESynC, and IESynC, we observe that IESynC algorithm can explore the same clusters and isolates (displayed by some figures) as ESynC algorithm. In many kinds of data sets, ESynC and IESynC can explore obvious clusters or isolates if selecting a proper value for parameter δ, 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, AP, K-Means, and FCM is larger than that of ESynC and IESynC. In many data sets, ESynC, IESynC, and DBSCAN have better ability than SynC, K-Means, FCM, AP, and Mean Shift in exploring clusters and isolates. Specially, AP algorithm needs the largest time cost.

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

Because of the limited page space, we only select some typical data sets (twelve kinds of artificial data sets, eight UCI data sets, and three bmp picture data sets) used in our

Table 6 Compare three synchronization algorithms (SynC, ESynC, and IESynC) using three picture data sets. In Table 6, parameter $\delta = 18$ or 30 in SynC, ESynC, and IESynC; parameter $r_i\,(i = 1, 2, 3) = 6$ in IESynC

(a). Parameter $\delta = 18$

Measure indexes of algorithms	Name of algorithms	Data sets		
		Picture1	Picture2	Picture3
Spend time (second)	SynC	662	676	9795
	ESynC	**132**	**122**	3254
	IESynC	202	209	**2034**
Iterative times	SynC	50	50	50
	ESynC, IESynC	**10**	**9**	**16**
The number of steady locations	SynC	941	467	2868
	ESynC, IESynC	13	5	14
The cluster order parameter r_c	SynC	58.6149	118.4821	88.4415
	ESynC, IESynC	**2712.8392**	**3321.3298**	**6127.5541**
AveLen(T)	SynC	11.0537	10.5757	11.5605
	ESynC, IESynC	**0**	**0**	**0**

(b). Parameter $\delta = 30$

Measure indexes of algorithms	Name of algorithms	Data sets		
		Picture1	Picture2	Picture3
Spend time (second)	SynC	749	797	10930
	ESynC	**122**	**179**	**2139**
	IESynC	247	473	3302
Iterative times	SynC	50	50	50
	ESynC, IESynC	**9**	**13**	**10**
The number of steady locations	SynC	928	472	2896
	ESynC, IESynC	4	2	6
The cluster order parameter r_c	SynC	55.2653	106.8353	87.9900
	ESynC, IESynC	**3630.5206**	**5015.0178**	**11105.6154**
AveLen(T)	SynC	16.9417	17.5013	19.0378
	ESynC, IESynC	**0**	**0**	**0**

Note: The bold in Table 6 marks the better results of ESynC algorithm or IESynC algorithm.

experiments. For all experimental data sets, we observe that ESynC algorithm improves SynC algorithm in local synchronization effect, iterative times, and time cost. For other data sets, we think ESynC algorithm is still superior to SynC algorithm for clustering. We believe that the selection of experimental data sets is not biased. We also know that IESynC algorithm is faster than ESynC algorithm depending on the selected data. But IESynC algorithm can explore the same clusters and isolates as ESynC algorithm in any cases.

6 Conclusions

This paper presents another synchronization clustering algorithm, ESynC, which can get better clustering results than the original synchronization clustering algorithm, SynC. From the experimental results, we observe that ESynC algorithm has less iterative times, faster clustering speed, and better clustering quality than SynC algorithm in many kinds of data sets. ESynC algorithm gets better local synchronization effect than SynC algorithm because of its more proper synchronization model. ESynC algorithm can also get better or similar clustering quality or faster clustering speed than some classical clustering algorithms in some data sets.

To our knowledge, the linear version of Vicsek model used for clustering is introduced in this paper firstly. The major contributions of this paper can be summarized as follows:

(1) It presents an Effective Synchronization Clustering algorithm (ESynC), which is an improved version of SynC algorithm, using a linear version of Vicsek model.

(2) It compares the linear version of Vicsek model, the extensive Kuramoto model, and the original version of Vicsek model for exploring clusters in dynamic clustering process.
(3) It introduces an improved version of ESynC algorithm in time cost, IESynC algorithm, and validates that IESynC algorithm can get some improvement of time cost in some data sets.
(4) It presents and validates a convergent condition of dynamical clustering in synchronization clustering algorithms, the t-step average length of edges.

ESynC algorithm and SynC algorithm use a global searching strategy to construct the δ near neighbor point set for every point in each evolution, so their time complexity is $O(Tdn^2)$. IESynC algorithm uses a local searching strategy to construct the δ near neighbor point set for every point in each evolution. In some cases, the time complexity of IESynC algorithm is $O(Tdn\log n)$. In some cases, the time complexity of IESynC algorithm is near $O(Tdn^2)$. Where n is the number of all points, d is the number of dimensions, and T is the times of synchronization. So we say IESynC algorithm is a parametric algorithm.

Like DBSCAN and SynC, ESynC algorithm is also robust to outliers or isolates. In DBSCAN algorithm, Mean Shift algorithm, and ESynC algorithm, the number of clusters does not have to be fixed before clustering. Usually, parameter δ has some valid interval that can be determined by using two exploring methods presented in [7], the heuristic method described by Theorem 1 and Property 1, or the MDL-based method presented in [4]. More often, the valid interval of parameter δ in ESynC algorithm is longer than it in DBSCAN algorithm. Comparing with SynC, K-Means, FCM, and AP, ESynC algorithm can obtain better or similar clustering quality.

Although our algorithms have shown promising results, there are still some limitations. First, the time complexity of ESynC algorithm is $O(Tdn^2)$, which limits its applicability to big data. Second, IESynC algorithm is a parametric algorithm. The time complexity of IESynC algorithm is sensitive to parameter r_i $(i = 1, 2, \ldots, d)$. Third, like DBSCAN algorithm and CNNI algorithm [7], ESynC algorithm is also sensitive to parameter δ in some scatter data sets. Fourth, when many noises and few obvious clusters exist, DBSCAN algorithm and ESynC algorithm cannot generate multi-level clusters because parameter δ is fixed before clustering.

This work opens some possibilities for further improvement and investigation. First, do more comparison experiments. For example, in the process of constructing δ near neighbor point sets, comparing our improved method based on δ near neighbor grid cell set [5] and Red-Black tree with R-tree, SR-tree index structure, and other space index structures should be valuable for practical work. Second, further improve ESynC algorithm in time cost. For example, designing similarity-preserving hashing function that needs $O(1)$ time complexity is valuable in the process of constructing δ near neighbor point set. Third, extend the applicability and explore the clustering effect of our algorithms in high-dimensional data. Fourth, further explore more proper and simple methods to estimate parameter δ. Fifth, ESynC algorithm has some similarity with Mean Shift algorithm, but they have essential difference. ESynC algorithm is a dynamic synchronization clustering algorithm, and Mean Shift algorithm is a clustering algorithm based on a non-parametric model. So, it is important to explore the relation between ESynC algorithm and Mean Shift algorithm further.

Acknowledgments This work was supported by three projects (cstc2016jcyjA0521, cstc2014jcyjA40035, cstc2016jcyjA0063) from Chongqing Cutting-edge and Applied Foundation Research Program of China and a project (No. 14RC08) from Chongqing Three Gorges University of China. The author thanks the anonymous reviewers and the editors for their useful suggestions which led to the improvement of this paper.

References

1. Agrawal R, Gehrke J, Gunopolos D et al (1998) Automatic subspace clustering of high dimensional data for data mining application. In: Proceedings of ACM SIGMOD, pp 94–105
2. 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
3. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York
4. Böhm C, Plant C, Shao J, et al. (2010) Clustering by synchronization. In: Proceedings of ACM SIGKDD. USA, Washington, pp 583-592
5. Chen X (2013) Clustering based on a near neighbor graph and a grid cell graph. J Intell Inf Syst 40(3):529–554
6. Chen X (2014) A fast synchronization clustering algorithm. arXiv:1407.7449 [cs.LG].
7. Chen X (2015) A new clustering algorithm based on near neighbor influence. Expert Syst Appl 42(21):7746–7758
8. Chen Z, Zhang HT, Chen X, Chen D, Zhou T (2015) Two-level leader-follower organization in pigeon flocks. Eur phys Lett 112(2):20008
9. Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis. IEEE T Pattern Anal 24(5):603–619
10. Czirok A, Barabasi AL, Vicsek T (1999) Collective motion of self-propelled particles: kinetic phase transition in one dimension. Phys Rev Lett 82:209–212
11. 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 the 2-th International Conference on Knowledge Discovery and Data Mining, pp 226–231
12. Frank A (2010) Asuncion a. UCI Machine Learning Repository Irvine, University of California
13. Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(1):972–976

14. Fukunaga K, Hostetler L (1975) The estimation of the gradient of a density function, with applications in pattern recognition. IEEE T Inform Theory 21(1):32–40
15. Gräunwald P (2005) A tutorial introduction to the minimum description length principle. MIT Press, Cambridge
16. Guha S, Rastogi R, Shim K (2001) Cure: an efficient clustering algorithm for large databases. Inform Syst 26(1):35–58
17. Horn D, Gottlieb A (2002) Algorithm for data clustering in pattern recognition problems based on quantum mechanics. Phys Rev Lett 88(1):018702
18. Huang JB, Kang JM, Qi JJ, Sun HL (2013) A hierarchical clustering method based on a dynamic synchronization model. Sci China Ser F: Inform Sci 43:599–610
19. Jadbabaie A, Lin J, Morse AS (2003) Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE T Automat Contr 48(6):998–1001
20. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
21. Jaromczyk JW, Godfried T (1992) Relative neighborhood graphs and their relatives. In: Proceedings of the IEEE, vol 80, pp 1502-1517
22. Ji P, Peron TK, Menck PJ, Rodrigues FA, Kurths J (2013) Cluster explosive synchronization in complex networks. Phys Rev Lett 110(21):218701
23. Karypis G, Han EH, Kumar V (1999) CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Comput 32(8):68–75
24. Leyva I, Navas A, Sendiña-Nadal I et al. (2013) Explosive transitions to synchronization in networks of phase oscillators. Sci Rep-UK 3:1281
25. Liu Z, Guo L (2008) Connectivity and synchronization of Vicsek model. Sci China Ser F: Inform Sci 51(7):848–858
26. Luxburg UV (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
27. 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
28. Nagy M, Ákos Z, Biro D, Vicsek T (2010) Hierarchical group dynamics in pigeon flocks. Nature 464(7290):890–893
29. Reynolds C (1987) Flocks, birds, and schools: a distributed behavioral model. Comput Graph 21:25–34
30. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496
31. Roy S, Bhattacharyya DK (2005) An approach to find embedded clusters using density based techniques. Lect Notes Comput Sc 3816:523–535
32. Schaeffer SE (2007) Graph clustering. Comput Sci Rev 1(1):27–64
33. Schölkopf B, Smola A, Müller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10(5):1299–1319
34. Shao J, Böhm C, Yang Q, Plant C (2010) Synchronization based outlier detection. In: Proceedings of the ECML/PKDD, pp 245–260
35. Shao J, Yang Q, Böhm C, Plant C (2011) Detection of arbitrarily oriented synchronized clusters in high-dimensional data. In: Proceedings of the ICDM, pp 607–616
36. Shao J, Hahn K, Yang Q et al. (2010) Hierarchical density-based clustering of white matter tracts in the human brain. Int J Knowl Disc Bioin 1(4):1–25
37. Shao J, He X, Plant C, Yang Q, Böhm C (2013) Robust synchronization-based graph clustering. In: Proceedings of the 17-th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 249–260
38. Shao J, He X, Böhm C, Yang Q, Plant C (2013) Synchronization inspired partitioning and hierarchical clustering. IEEE T Knowl Data En 25(4):893–905
39. Shao J, Ahmadi Z, Kramer S (2014) Prototype-based learning on concept-drifting data streams. In: Proceedings of ACM SIGKDD, pp 412–421
40. Strehl A, Ghosh J (2002) Cluster ensembles - a knowledge reuse framework for combining multiple partitions. J Mach Learn Res 3:583–617
41. Theodoridis S, Koutroumbas K (2006) Pattern recognition. Academic Press
42. 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
43. 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
44. Wang L, Liu Z (2009) Robust consensus of multi-agent systems with noise. Sci China Ser F: Inform Sci 52(5):824–834
45. Wang W, Yang J, Muntz R (1997) STING: A statistical information grid approach to spatial data mining. In: Proceedings of VLDB, pp 186–195
46. Zhang HT, Chen Z, Vicsek T, Feng G, Sun L, Su R, Zhou T (2014) Route-dependent switch between hierarchical and egalitarian strategies in pigeon flocks. Sci Rep-UK 4:5805
47. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: An efficient data clustering method for very large databases. In: Proceedings of SIGMOD, pp 103–114
48. Zou Y, Pereira T, Small M, Liu Z, Kurths J (2014) Basin of attraction determines hysteresis in explosive synchronization. Phys Rev Lett 112(11):114102

Xinquan Chen received the PhD degree from South China University of Technology in China. He is now a professor of Chongqing Three Gorges University in China and a postdoctor of University of Electronic Science and Technology of China. His research interests include data mining, machine learning, clustering algorithms, and optimization method.