A shrinking synchronization clustering algorithm based on a linear weighted Vicsek model

Xinquan Chen

August 27, 2019

Title Page

A shrinking synchronization clustering algorithm based on a linear weighted Vicsek model

Xinquan Chen[1, 2]

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

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

chenxqscut@126.com

Author: Xinquan Chen

Corresponding Author: Xinquan Chen

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

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

Post Address:

Xinquan Chen

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

Conflicts of interest: None

A shrinking synchronization clustering algorithm based on a linear weighted Vicsek model

Abstract: Clustering is an unsupervised learning method that tries to find some distributions and patterns in unlabeled data sets. Although clustering algorithms have been studied for many years, none of them is all purpose. This paper presents a Shrinking Synchronization Clustering (SSynC) algorithm by using a linear weighted Vicsek model. It is inspired by Synchronization Clustering (SynC) algorithm and Vicsek model. After some analysis and comparison, we find that SSynC algorithm based on the linear weighted Vicsek model has better synchronization effect than SynC algorithm based on an extensive Kuramoto model and has similar synchronization effect with Effective Synchronization Clustering (ESynC) algorithm based on another linear version of Vicsek model. In the simulations, several clustering algorithms (SynC, ESynC, K-Means, FCM, AP, DBSCAN, and Mean Shift) are used as comparative algorithms. By some simulated experiments of some artificial data sets, several real data sets, and three picture data sets, we observe that SSynC algorithm not only gets better local synchronization effect but also needs less iterative times and time cost than SynC algorithm. Moreover, SSynC algorithm needs less time cost than ESynC algorithm and almost get the same local synchronization effect and the same iterative times. Extensive comparison experiments with some class clustering algorithms demonstrate the effectiveness of our algorithm. At last, it gives some research expectations to popularize this algorithm.

Keywords: Data mining; Clustering; SynC algorithm; Kuramoto model; Shrinking synchronization; A linear weighted Vicsek model; Near neighbor points

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. Clustering has been used in many areas such as machine learning, pattern recognition, image processing, marketing and costumer analysis, agriculture, security and crime detection, information retrieval, and bioinformatics.

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 some 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 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, some researchers try their best to develop new efficient and effective clustering algorithms.

The traditional clustering algorithms are usually categorized into partitioning methods (Bezdek, 1981; MacQueen, 1967), hierarchical methods (Guha et al., 1998; Karypis et al., 1999; Zhang et al., 1996), density-based methods (Ankerst et al., 1999; Ester et al., 1996; Roy et al., 2005; Rodriguez et al., 2014), grid-based methods (Agrawal et al. 1998; Wang et al., 1997), model-based methods (Theodoridis et al., 2006), and graph-based methods (Jaromczyk and Godfried, 1992; Schaeffer, 2007). Recent clustering methods have quantum clustering algorithms (Horn et al., 2002), spectral clustering algorithms (Luxburg, 2007; Schölkopf et al., 1998), and synchronization clustering algorithms (Böhm et al., 2010; Huang et al., 2013; Shao et al., 2013a, 2013b, 2014; Chen, 2017).

Recently, several original clustering algorithms, such as Affinity Propagation (AP) algorithm (Frey et al., 2007) and Synchronization Clustering (SynC) algorithm (Böhm et al., 2010), and clustering by fast search and find of Density Peaks (DP) algorithm (Rodriguez et al., 2014), were published. AP 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 (Böhm et al., 2010) is the first synchronization clustering algorithm. After Böhm et al. (2010) presented SynC algorithm, synchronization clustering attracts some researchers, and some synchronization clustering methods (Huang et al., 2013; Shao et al., 2013a, 2013b, 2014; Chen, 2016) were published from different views. DP (Rodriguez et al., 2014) 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 (Rodriguez et al., 2014), 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 create 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 Böhm et al. (2010), claimed that it can find the intrinsic structure of the data set without any distribution assumptions and handle outliers by dynamic synchronization (Böhm et al., 2010).

This paper researches another synchronization clustering method based on the linear weighted Vicsek model comparing to SynC algorithm, and presents a Shrinking Synchronization Clustering (SSynC) algorithm by using the linear weighted Vicsek model. It is inspired by SynC algorithm and Vicsek model.

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

2. Related work

This paper is inspired by several papers (Vicsek et al., 1995; Jadbabaie et al., 2003; Wang et al., 2009; Böhm et al., 2010).

In 1995, Vicsek et al. (1995) 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 (Reynolds, 1987). Simulation results demonstrate that some systems using Vicsek model (Vicsek et al., 1995) or one-dimensional models presented by Czirok et al. (1999) 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. (2003) 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. (2008) provided the synchronization property of Vicsek model after given initial conditions and the model parameters. In 2009, Wang et al. (2009) researched Vicsek model under noise disturbances and presented some theoretical results. In 2010, Nagy, M. et al. (2010) 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, H. T. et al. (2014) 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, Z. et al. (2015) found that pigeon flocks adopted a simple two-level interactive network containing one leader and some followers. And they think that “the two-level organization of group flight may be more efficient than a multilevel topology for small pigeon flocks”.

In 2010, Böhm et al. 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. In order to implement automatic clustering, those natural clusters can be discovered by using the Minimum Description Length principle (MDL) (GrÄunwald, 2005). After SynC algorithm was presented, Shao et al. published several synchronization clustering papers from several views (Shao et al., 2010, 2011, 2013a, 2013b, 2014). In order to find subspace clusters of some high-dimensional sparse data sets, a novel effective and efficient subspace clustering algorithm, ORSC (Shao et al., 2011), was proposed. In order to detect the outliers from a real complex data set more naturally, a novel outlier detection algorithm was presented from a new perspective, "Out of Synchronization" (Shao et al., 2010). In order to find the intrinsic patterns of a complex graph, a novel and robust graph clustering algorithm, RSGC (Shao et al., 2013a), was proposed by regarding the graph clustering as a dynamic process towards synchronization. In order to explore meaningful levels of the hierarchical cluster structure, a novel dynamic hierarchical clustering algorithm, hSync (Shao et al., 2013b), was presented based on synchronization and the MDL principle. In 2013, Huang et al. (2013) also presented a synchronization-based hierarchical clustering method basing on the work of Böhm et al. (2010). In 2014, Chen (2014) presented a Fast Synchronization Clustering (FSynC) algorithm basing on the work of Böhm et al. (2010). In 2017, Chen (2017) presented an Effective Synchronization Clustering (ESynC) algorithm based on a linear version of Vicsek model.

Recent years, some physicists also researched the explosive synchronization in some complexity networks to uncover the underlying mechanisms of the synchronization state (Ji et al., 2013; Leyva et al., 2013; Zou et al., 2014). 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. The δ near neighbor point set $\delta(P)$ of point P is defined as:

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

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 (Böhm et al., 2010)**.** 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 each 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 (Chen, 2017). 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, $E(t = 0) = \{(X_i, X_j) \mid X_j \in \delta(X_i), X_i\ (i = 1, 2, \ldots, n) \in S\}$ 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, $E(t) = \{(X_i(t), X_j(t)) \mid X_j(t) \in \delta\ (X_i(t)), X_i(t)\ (i = 1, 2, \ldots, n) \in V(t)\}$ 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 $AveLen(t) \rightarrow 0$, as more δ near neighbor points synchronize together with time evolution. In our algorithm, $AveLen(t)$ can be used to characterize the degree of local synchronization.

Definition 5 (Böhm et al., 2010). 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 (Chen, 2017)**.** 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 each 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. (2003) and Wang et al. (2009) is described by:

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

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.

Definition 7. A core is defined as:

In our Shrinking Synchronization Clustering (SSynC) algorithm, point X can be regarded as an active core C if and only if:

(a). Point X is active in the current synchronization step.

(b). Point X is not labeled as an attributive point of another core.

At this time, the points in the ε near neighbor point set $\varepsilon(C)$ of core C should be labeled as attributive points of core C, where parameter ε is a small real number that is less than parameter δ.

The data structure of core C can be defined as:

DS(C) = (Core_Id, Core_Location, Parent_CoreId, Number_ContainingPoints). (7)

In Eq.(7),

Core_Id is the identification number of core C in the original data set.

Core_Location is the current location of core C. It is a d-dimensional vector expressed by $C = (c_1, c_2, \ldots, c_d)$.

Parent_CoreId is the identification number of the parent of core C in the original data set. At the original step of dynamic clustering, the Parent_CoreId of core C is itself. At the middle or final of dynamic clustering, the Parent_CoreId of core C is the Core_Id of the attributive core of core C.

Number_ContainingPoints is the number of points that are represented or contained by the core C.

The main purpose of introducing the concept of core is to record the clustering information in SSynC algorithm.

Definition 8. A synchronization model for clustering a core set is defined as:

Core $C = (c_1, c_2, \ldots, c_d)$ is a vector in a d-dimensional Euclidean space. If each core C is regarded as an agent according to an extended linear version of Vicsek model (this model is also named as: the linear weighted Vicsek model), with an interaction in the δ near neighbor point set $\delta(C)$, then the dynamics of core C over time is described by:

$$C(t+1) = \frac{1}{(count(C(t)) + \sum_{Y \in \delta(C(t))} count(Y))} \left(count(C(t)) \cdot C(t) + \sum_{Y \in \delta(C(t))} (count(Y) \cdot Y) \right), \quad (8)$$

where $C(t = 0) = (c_1(0), c_2(0), \ldots, c_d(0))$ represents the original phase of core C, $C(t+1)$ describes the renewal phase value of core C at the t step evolution, and $count(C)$ represents the value of the Number_ContainingPoints of core C.

In the dynamical clustering, if the Parent_CoreId of core C is itself and the value of the Number_ContainingPoints of core C is equal to 1, then Eq.(8) is equivalent with Eq.(6). Actually, in the dynamical clustering, if core C is represented by its parent core (which means that the value of the Number_ContainingPoints of the parent core is added by $count(C)$), then Eq.(8) can be used for saving time and space in SSynC algorithm.

Definition 9. The data set $S = \{X_1, X_2, \ldots, X_n\}$ using the linear weighted Vicsek model described by Eq.(8) for clustering is said to achieve local synchronization if the final locations of all points satisfy:

$$X_i(t = T) = RC_k(T),\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, K, \quad (9)$$

where T is the times of the final synchronization, K is the number of the root cores in the final synchronization step, $RC_k(T)$ is the k-th root core in the final synchronization step.

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

Definition 10. The data set $S = \{X_1, X_2, \ldots, X_n\}$ uses the linear weighted Vicsek model described by Eq.(8) for synchronization clustering. In each evolution step of synchronization clustering, all cores become some trees with synchronization action. When the number of root cores in the t-step evolution is equal to that in the $(t+1)$-step evolution, an average difference between the root cores in the *t*-step evolution and the root cores in the (*t*+1)-step evolution is defined as:

$$differInRootCores(t, t+1) = \frac{1}{n_t}\sum_{k=1}^{n_t} dis(RC_k(t).Core\_Location, RC_k(t+1).Core\_Location),\ k = 1, 2, \cdots, n_t, \tag{10}$$

where n_t is the number of the root cores in the *t*-step evolution, $RC_k(t)$.Core_Location is the location of the *k*-th root core in the *t*-step evolution, and $dis(RC_k(t)$.Core_Location, $RC_k(t+1)$.Core_Location) is the dissimilarity between the location of the *k*-th root core in the *t*-step evolution and the location of the *k*-th root core in the (*t*+1)-step evolution.

Apparently, if the average difference between the root cores in the *t*-step evolution and the root cores in the (*t*+1)-step evolution computed by Eq.(10) is less than a predefined threshold, we think SSynC algorithm can exit.

Theorem 1. The data set $S = \{X_1, X_2, \ldots, X_n\}$ using the linear weighted Vicsek model described by Eq.(8) for clustering will achieve local synchronization, if parameter δ satisfies:

$$\delta_{\min} \le \delta \le \delta_{\max}, \tag{11}$$

Suppose $e_{\min}$ (MST(S)), which is also 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 $\delta_{\min} = e_{\min}$(MST(S)). If the data set *S* has no isolate, then usually there is $e_{\max}(\text{MST}(S)) \le \delta_{\max} \le \max\{dis(X_i, X_j)|\ (X_i, X_j \in S) \wedge (X_i \neq X_j)\}$. 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 Eq.(8), there is $X_i(t+1)$ = 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 $\delta_{\min} \le \delta \le \delta_{\max}$, 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 weighted Vicsek model described by Eq.(8) for clustering will obtain an effective result of local synchronization with some obvious clusters or isolates, if parameter δ satisfies:

$$\max\{longestEdgeInMst(cluster_k)\ |\ k = 1, 2, \ldots, K\} < \delta < \min\{dis(cluster_i, cluster_j)\ |\ i \neq j,\ i, j = 1, 2, \ldots, K\}, \tag{12}$$

where *longestEdgeInMst*($cluster_k$) 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 $\max\{longestEdgeInMst(cluster_k) \mid k = 1, 2, \ldots, K\}$, then data points in the same cluster will synchronize. If parameter δ is less than $\min\{dis(cluster_i, cluster_j) \mid i, j = 1, 2, \ldots, K\}$, then data points in different obvious clusters cannot synchronize.

4. A shrinking synchronization clustering algorithm based on a linear weighted Vicsek model

SSynC algorithm has similar process with SynC algorithm (Böhm et al., 2010) and ESynC algorithm except using a different dynamical synchronization clustering model. The synchronization model represented by Eq.(8) can be used for clustering a core set.

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 The description of SynC algorithm

The original synchronization clustering algorithm named as SynC is developed by Böhm et al. (Böhm et al., 2010). In order to make a difference between SynC algorithm and our algorithm, we introduce it below using our language according to the description of (Böhm et al., 2010).

Algorithm name: Synchronization Clustering algorithm (SynC).

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 is described as follows:

```
1   IterateStep $t$ is set as zero firstly, that is: $t \leftarrow 0$;
2   for ($i = 1; i \le n; i$++)
3       $X_i(t) \leftarrow X_i$;
4   while (the dynamical clustering does not satisfy its convergent condition)
5   {
6       for ($i = 1; i \le n; i$++)
7       {
8           Construct the $\delta$ near neighbor point set $\delta(X_i(t))$ for each point $X_i(t)$ ($i = 1, 2, \ldots, n$) using Eq.(1) of Definition 1;
9           Compute the renewal value, $X_i(t+1)$, of $X_i(t)$ using Eq.(2) of Definition 2;
10      }
11      Compute the cluster order parameter $r_c$ of all points using Eq.(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 $S(T) = \{X_1(T), X_2(T), \dots, X_n(T)\}$, 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 $S$.
```

4.2 The description of ESynC algorithm

Effective Synchronization Clustering algorithm (ESynC) is developed by Chen (Chen, 2017). In order to make a difference between ESynC algorithm and our algorithm, we introduce it simply below.

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

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

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

Procedure:

Step1. Initialization:

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

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

```
4   while ((the dynamical clustering does not satisfy its convergent condition) and ($t < 50$))
5   {
6       for ($i = 1$; $i \le n$; $i$++)
7       {
8           Construct the $\delta$ near neighbor point set $\delta(X_i(t))$ for each point $X_i(t)$ ($i = 1, 2, \dots, n$) using Eq.(1) of Definition 1;
9           Compute the renewal value, $X_i(t+1)$, of $X_i(t)$ using Eq.(6) of Definition 6;
10      }
11      Compute the $t$-step average length of edges of all points, $AveLen(t)$, using Eq.(4) of Definition 4;
        /* We can also compute the cluster order parameter $r_c$ using Eq.(5) of Definition 5 instead of computing $AveLen(t)$. */
12      IterativeStep $t$ is increased by 1, that is: $t$++;
13      if ($AveLen(t) \to 0$)    /* $AveLen(t) \to 0$ is equivalent with $r_c \to$ 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 $S(T) = \{X_1(T), X_2(T), \dots, X_n(T)\}$, 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.

4.3 The description of SSynC algorithm

Algorithm name: a Shrinking Synchronization Clustering algorithm (SSynC).

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

Output: The final core set $CS(T) = \{C_1(T), C_2(T), \ldots, C_n(T)\}$.

Procedure:

Step1. Initialization:

```
1    IterateStep $t$ is set as zero firstly, that is: $t \leftarrow 0$;
     /* Create initial core set $C(t = 0) = \{C_1(t = 0), C_2(t = 0), \ldots, C_n(t = 0)\}$. */
2    for ($i = 1$; $i \leq n$; $i$++)
3    {
4        $C_i(t = 0)$.Core_Id $\leftarrow i$;
5        $C_i(t = 0)$.Core_Location $\leftarrow X_i$;
6        $C_i(t = 0)$.Parent_CoreId $\leftarrow i$;
7        $C_i(t = 0)$.Number_ContainingPoints $\leftarrow 1$;
8    } // for
     /* Create initial active point set $AP(t = 0)$. */
9    $AP(t = 0) \leftarrow \{X_1, X_2, \ldots, X_n\}$;
10   NumberOfAP($t = 0$) $\leftarrow n$;/* NumberOfAP($t = 0$) is used to record the number of points in the active point set $AP(t = 0)$. */
```

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

```
11   while ((the dynamical clustering does not satisfy its convergent condition) and ($t < 50$))
12   {
13       for (each point $Y(t)$ in the active point set $AP(t)$)
14       {
15           According to Definition 1, in the active point set $AP(t)$ construct the $\delta$ near neighbor point set $\delta(Y(t))$ for point $Y(t)$;
16           Compute the renewal value, $Y(t+1)$, of $Y(t)$ using Eq.(8) of Definition 8;
17       } // for
         /* After the above for repetition, we get a point set $AP(t+1)$ that is composed of the renewal value $Y(t+1)$ of each point $Y(t)$ in the active point set $AP(t)$. */
18       for (each unlabeled point $Y(t+1)$ in the point set $AP(t+1)$)
19       {
20           The member "Core_Location" of the corresponding core of point $Y(t+1)$ is updated by the value of $Y(t+1)$;
21           According to Definition 1, in the point set $AP(t+1)$ construct the $\varepsilon$ near neighbor point set $\varepsilon(Y(t+1))$ for point $Y(t+1)$;
22           for (each unlabeled point $Z(t+1)$ in the $\varepsilon$ near neighbor point set $\varepsilon(Y(t+1))$ of point $Y(t+1)$)
23           {
24               Point $Z(t+1)$ is labeled as inactive point;
25               The member "Parent_CoreId" of the corresponding core of point $Z(t+1)$ is assigned by the member "Core_Id" of the corresponding core of point $Y(t+1)$;
26               The member "Number_ContainingPoints" of the corresponding core of point $Z(t+1)$ is added into the member "Number_ContainingPoints" of the corresponding core of point $Y(t+1)$;
27           } // for
28       } // for
29       Delete all labeled inactive points from $AP(t+1)$;        /* After this deleting process,
$AP(t+1)$ only contains those active points, which are also the root nodes in its disjoint-set forest. */
30          NumberOfAP($t$+1) is assigned by the current number of unlabeled points of the renewal active point set $AP(t+1)$;
31          IterateStep $t$ is increased by 1, that is: $t$++;
32          if (NumberOfAP($t$+1) == NumberOfAP($t$) and (the difference between $AP(t+1)$ and $AP(t)$ is very small))  /* NumberOfAP(t+1) == NumberOfAP(t) means the number of points in the renewal active point set $AP(t+1)$ is equal to the number of points in the active point set $AP(t)$ , and the difference between $AP(t+1)$ and $AP(t)$ can be computed by Eq.(10). */
33                We think the dynamical clustering reaches its convergent result, and then exit from the while repetition;
34    } // while
35    Compress the paths of some inactive cores in the core set $CS(t)$ just like the joint-set method such that the largest height of leaf cores is less than or equal to 2 (Note: the height of root cores is 1).
```

Step3. Finally we get a core set $CS(T) = \{C_1(T), C_2(T), \ldots, C_n(T)\}$, where T is the times of the while repetition in Step2. The final set $CS(T)$ reflects the natural clusters or isolates of the data set S.

For example, if $C_i(T)$.Core_Id is equal to $C_i(T)$.Parent_CoreId and $C_i(T)$.Number_ContainingPoints is equal to 1 are satisfied, we can think the i-th point is an isolate; if $C_i(T)$.Core_Id is equal to $C_i(T)$.Parent_CoreId and $C_i(T)$.Number_ContainingPoints >> 1 are satisfied, we can think the i-th point is a cluster core that represents some other points.

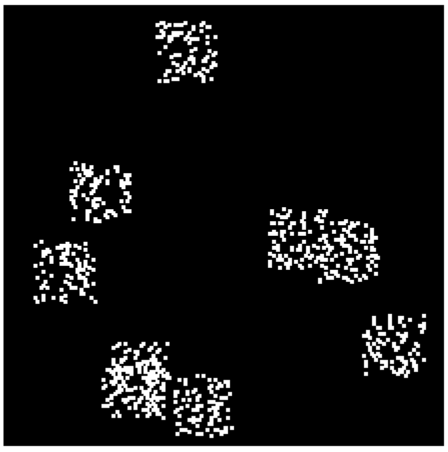
Note: Parameter ε that is less than parameter δ is a very small real number. Usually, if the distance of two points is less than ε, then they should always be in the same cluster.

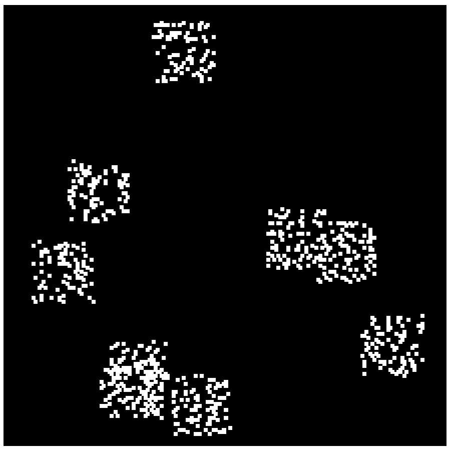
4.4 Compare the dynamic clustering processes of SynC algorithm, ESynC algorithm, and SSynC algorithm

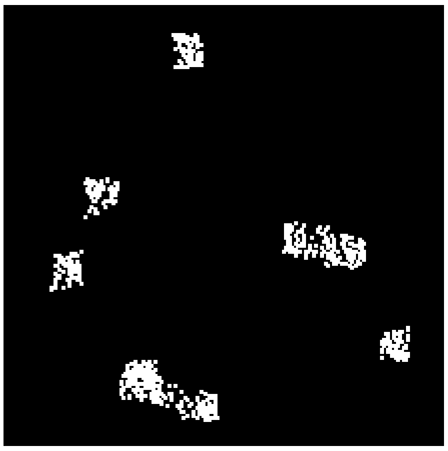
SynC algorithm uses the extensive Kuramoto model described by Eq.(2) that is a nonlinear renewal model at each step evolution. ESynC algorithm uses the linear version of Vicsek model described by Eq.(6) that is a linear renewal model at each step evolution. And SSynC algorithm uses the synchronization model described by Eq.(8) that is a linear weighted renewal model at each step evolution.

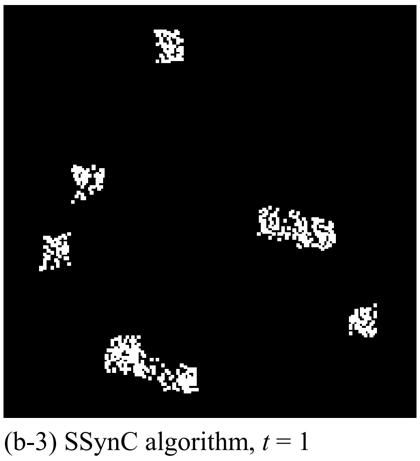
Fig. 1 uses 800 data points from DS0 to compare the tracks of the clustering processes of SynC algorithm, ESynC algorithm, and SSynC algorithm. Fig. 2 (a) compares the cluster order parameter with t-step evolution (t: 0 - 49) among SynC, ESynC, and SSynC. Fig. 2 (b) compares the t-step average length of edges (t: 0 - 49) among SynC, ESynC, and SSynC. And Fig. 2 (c) compares the relation between the final number of clusters and the value of parameter δ among the three algorithms.

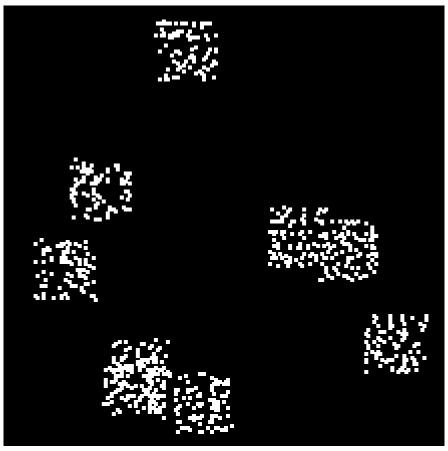
From Fig. 1, we observe that ESynC and SSynC have better local synchronization effect than SynC. From Fig. 2 (a) and (b), we observe that the t-step average length of edges is better than the cluster order parameter with t-step evolution in measuring the final synchronization results. From Fig. 2 (c), we observe that the smaller parameter δ is set in SynC, ESynC, and SSynC, the larger the final number of clusters is. For many data sets with obvious clusters, ESynC and SSynC 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 SynC algorithm is much larger than the actual number of clusters when parameter δ chooses any value in a long interval.

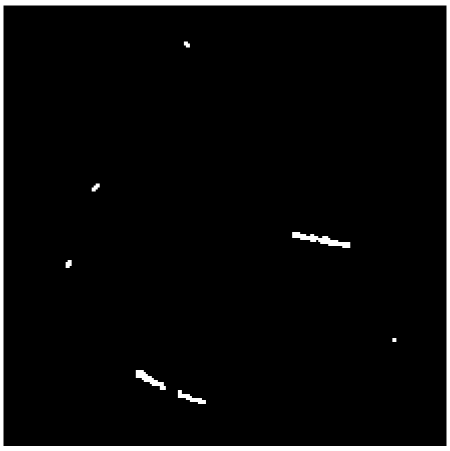

(a) $t = 0$ (The original locations of 800 data points from DS0)

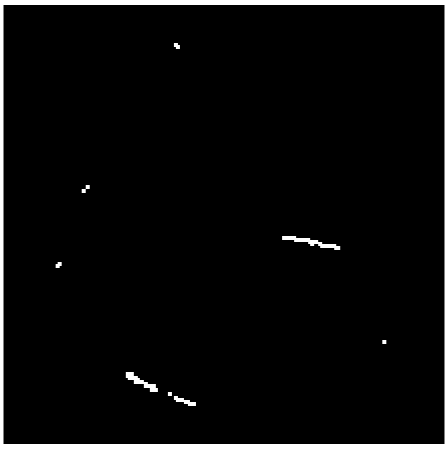

(b-1) SynC algorithm, $t = 1$

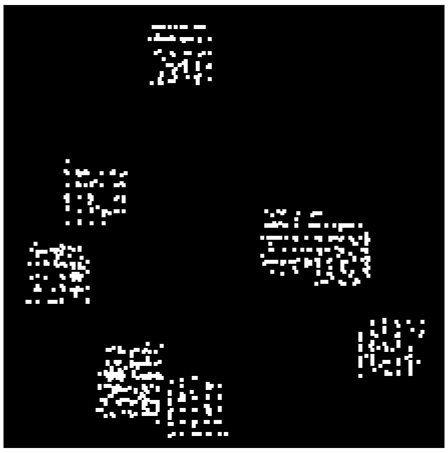

(b-2) ESynC algorithm, $t = 1$

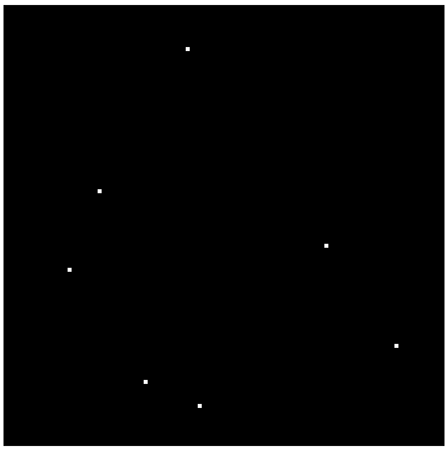

(b-3) SSynC algorithm, $t = 1$

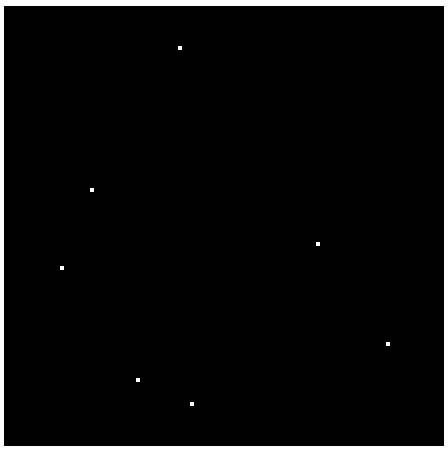

(c-1) SynC algorithm, *t* = 2

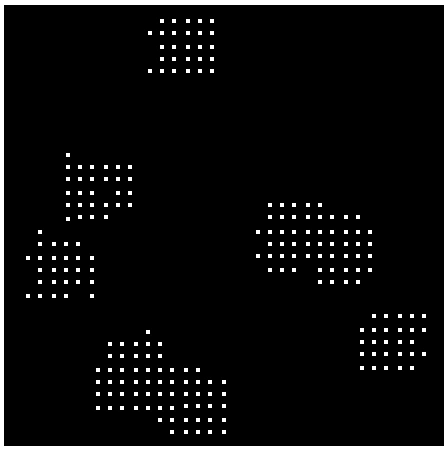

(c-2) ESynC algorithm, *t* = 2

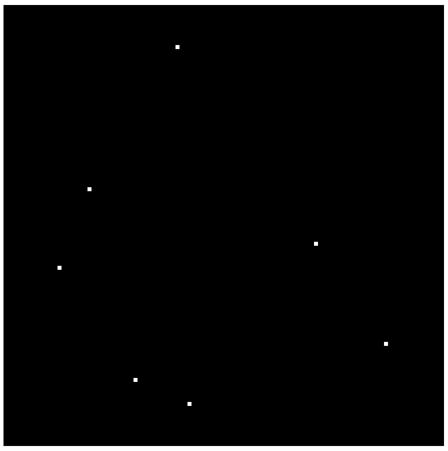

(c-3) SSynC algorithm, *t* = 2


(d-1) SynC algorithm, *t* = 5


(d-2) ESynC algorithm, *t* = 5


(d-2) SSynC algorithm, *t* = 5


(e-1) SynC algorithm, *t* = 45


(e-2) ESynC algorithm, *t* = 45

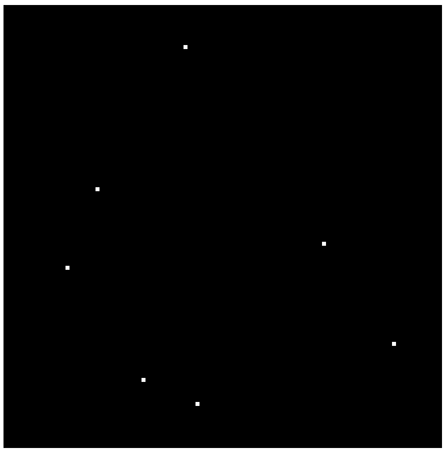
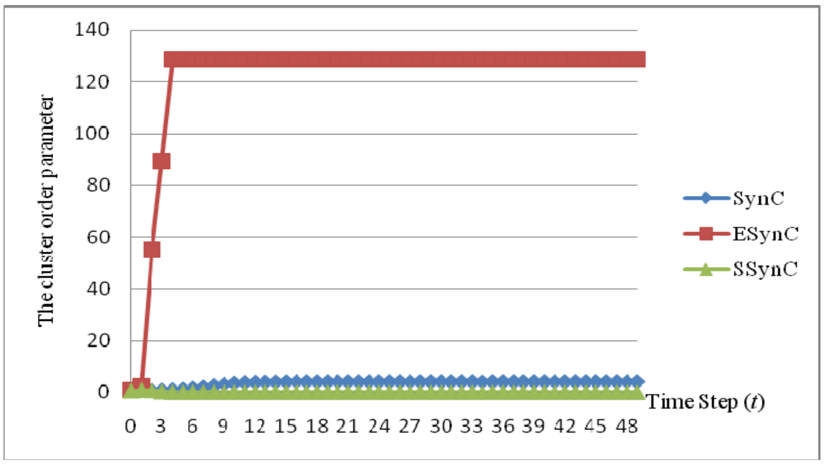
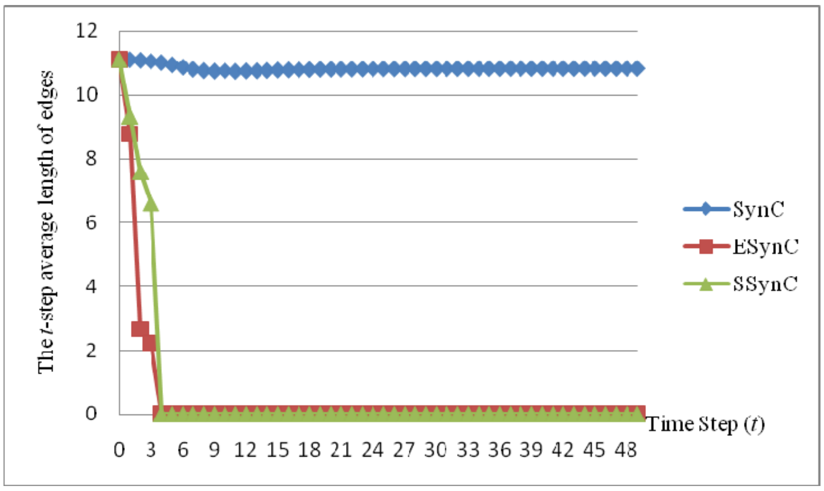

(e-3) SSynC algorithm, *t* = 45

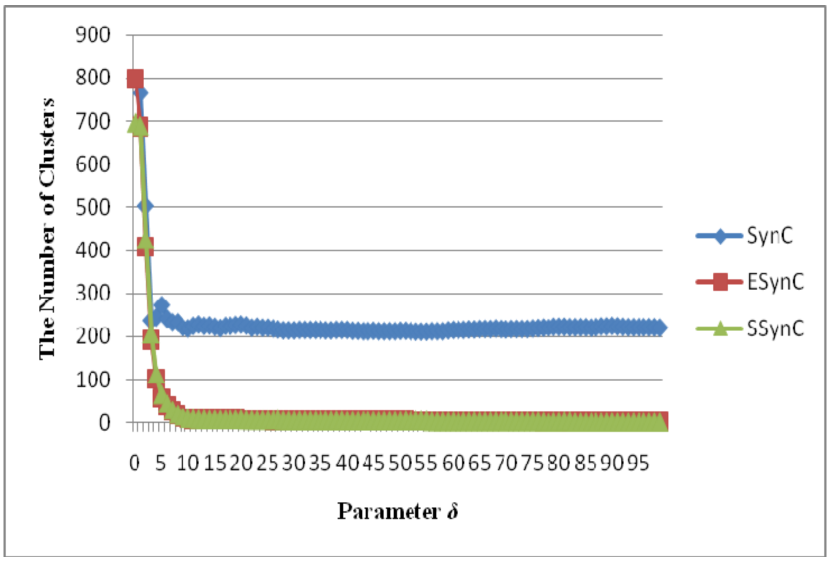
Fig. 1. Compare the dynamical synchronization clustering processes with time evolution among SynC algorithm, ESynC algorithm, and SSynC algorithm. From (a) to (e) of Fig. 1, the data set is 800 points from DS0, parameter δ is set as 18 in the three algorithms, and parameter ε is set as 1 in SSynC algorithm.



(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 parameter δ (δ: 0 - 99).

Fig. 2. Compare SynC algorithm, ESynC algorithm, and SSynC algorithm. In Fig. 2, the data set is 800 points from DS0, and parameter ε is set as 1 in SSynC algorithm. In Fig.2 (a) and (b), parameter δ is set as 18 in the three algorithms.

4.5 Time and space complexity analysis of SSynC algorithm

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

In the first synchronization process of Step2, constructing the δ near neighbor point sets for all points needs Time = $O(dn^2)$ and Space = $O(nd)$ if using a simple method needs. In the t-step synchronization process of Step2, constructing the δ near neighbor point sets for all points needs Time = $O(dn_{(t)}^2)$ and Space = $O(n_{(t)}d)$ if using a simple method, where $n_{(t)}$ is the number of active cores in the t-step synchronization process. In constructing the δ near neighbor point sets, the time cost can be decreased by using the strategy of "space exchanges time".

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

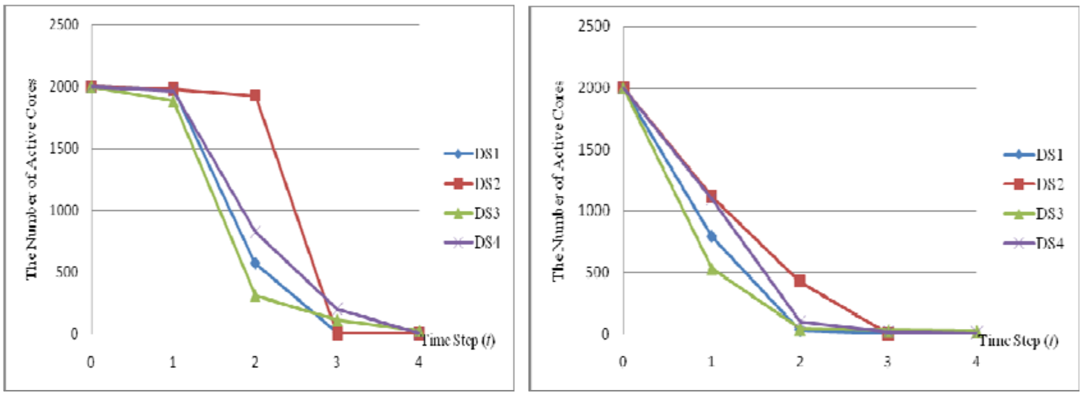
According to Böhm et al. (2010) and our analysis, SSynC algorithm needs Time = $O(d\cdot(n_{(t=0)}^2 + n_{(t=1)}^2 + \ldots + n_{(t=T\text{-}1)}^2)) < O(Tdn^2)$, which is usually less than SynC algorithm and ESynC algorithm. Here T is the times of the while repetition in Step2.

4.6 Setting parameters in SSynC algorithm

Parameter δ in SSynC algorithm that affects the clustering results is the same as SynC algorithm and ESynC algorithm. In Böhm et al. (2010), parameter δ is optimized by the MDL principle. In Chen (2015), 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.

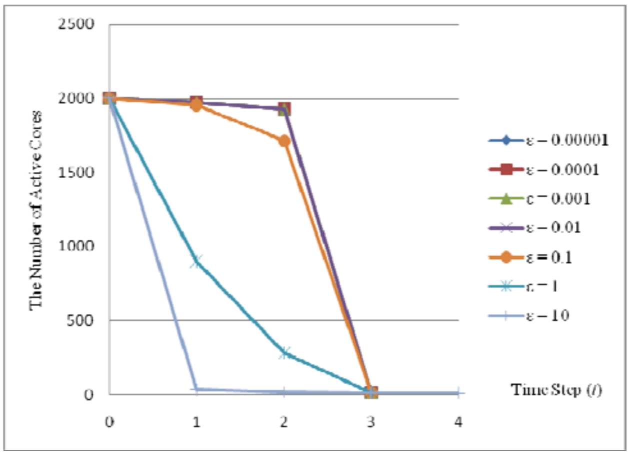
Parameter ε affects the time cost of SSynC algorithm slightly. In simulations, we get the same clustering results except time cost for several different values (such as 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, and 10) of parameter ε. Usually, the larger parameter ε is set, the less time cost SSynC algorithm needs.

Fig. 3 describes the number of active cores with time evolution based on SSynC algorithm using four different data sets. From Fig. 3, we observe that different data sets have different number of active cores with time evolution.



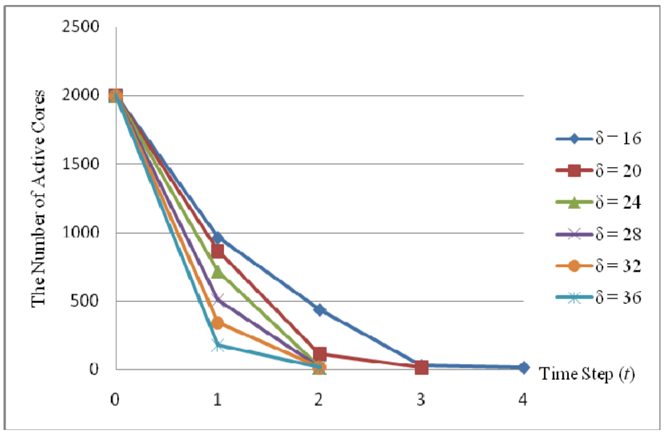
(a) Parameter $\varepsilon = 0.00001$ (b) Parameter $\varepsilon = 1$

Fig. 3. The number of active cores with time evolution based on SSynC algorithm using four different data sets. In Fig. 3, parameter δ is set as 22, parameter ε is set as 0.00001 and 1, and the numbers of points in the four data sets are all set as 2000.



(a) DS1

Fig. 4. The number of active cores with time evolution based on SSynC algorithm using several different values of parameter ε. In Fig. 4, parameter δ = 18 in DS1 and DS3, δ = 20 in DS2, δ = 22 in DS4; parameter ε is set as seven different value (0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10) respectively; and the number of points in the four data sets is all set as 2000. Fig. 4 (b), (c), and (d) are presented in Online Resource of Supplementary Material.



(a) DS1

Fig. 5. The number of active cores with time evolution based on SSynC algorithm using several different values of parameter δ. In Fig. 5, parameter δ is set as six different values (16, 20, 24, 28, 32, 36) respectively, parameter ε is set as 1, and the numbers of points in the four data sets are all set as 2000. Fig. 5 (b), (c), and (d) are presented in Online Resource of Supplementary Material.

Fig. 4 describes the number of active cores with time evolution using SSynC algorithm for several different values of parameter ε. From Fig. 4, we observe that parameter ε can affect the number of active cores with time evolution, which affects the time cost of SSynC algorithm.

Fig. 5 describes the number of active cores with time evolution using SSynC algorithm for different values of parameter δ. From Fig. 5, we observe that parameter δ can affect the number of active cores with time evolution, which affects the time cost of SSynC algorithm.

4.7 The improvement of SSynC algorithm

One improved version of SSynC algorithm can be designed by combining multidimensional grid partitioning method and Red-Black tree structure to construct the near neighbor point sets of all active cores. The improving method that can decrease the time cost of constructing near neighbor point set is introduced in Chen (2014). Here, we describe this method as possible as simply. Generally, we first partition the data space of the data set $S = \{X_1, X_2, \ldots, X_n\}$ by 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 active cores in each synchronization step, then constructing the δ near neighbor point set for every active core will become quicker when the number of grid cells is proper.

Before iterative evolution, if we set a proper value for parameter δ 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 implementation technique is often effective for some data sets.

Another improvement in time cost of SSynC algorithm is described in another paper.

4.8 The convergence of SSynC algorithm

The renewal computing of SSynC algorithm is similar to ESynC algorithm. In all simulations, if using SSynC algorithm to synchronize the original data set S, then all root nodes of its final core set $C(T) = \{C_1(T), C_2(T), \ldots, C_n(T)\}$ will stay on some steady locations after several iterations (many simulations only need 4 - 5 times). In the final convergent core set $C(T)$, those root cores that represent some points can be regarded as their cluster centers, and some root cores that represent only one or several points are regarded as the final synchronization locations of isolates.

5. Simulated experiments

5.1 Experimental design

Our experiments are finished in a personal computer (Capability Parameters: Pentium(R) Dual-Core CPU E5400 2.7 GHz, 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 our algorithm, there will be some simulated experiments of some artificial data sets, several UCI data sets (Frank et al., 2010), and three bmp pictures in the next sections.

DS0 is produced in a 2-D region [0, 200] × [0, 200] by a program. Four kinds of artificial data sets (DS1 - DS4) are produced in a 2-D region [0, 600] × [0, 600] by a program. Other kinds of artificial data sets (DS5 - DS16) are produced in a range [0, 600] in each dimension by a similar program. Iris et al. are several UCI data sets (Frank A, et al., 2010) 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 1 of Supplementary Material.

In section 5.2, SSynC algorithm will be compared with SynC algorithm, ESynC algorithm, and some other classic clustering algorithms (K-Means (MacQueen, 1967), FCM (Bezdek, 1981), AP (Frey et al., 2007), DBSCAN (Ester et al., 1996), Mean Shift (Fukunaga et al., 1975; Comaniciu et al., 2002)) in clustering quality and time cost using some artificial data sets.

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

In section 5.4, SSynC algorithm will be compared with SynC algorithm, ESynC algorithm, and some other classic clustering algorithms in compress effective of clustering results, clustering quality, and time cost using three bmp pictures.

In the experiments, parameter δ used in SynC, ESynC, SSynC, DBSCAN, and Mean Shift is the threshold of Definition 1. In DBSCAN algorithm, parameter *MinPts* is set as 4, and parameter Eps is the same as parameter δ.

The detailed discussion on how to construct δ near neighbor point sets is described in Chen. (2013). How to select a proper value for parameter δ for SynC algorithm is discussed in Böhm et al. (2010). SSynC algorithm and ESynC algorithm can use the same method as SynC algorithm to select a proper value for parameter δ. In SSynC algorithm, parameter ε has trivial effect in time cost and clustering results.

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

Comparison results of these clustering algorithms are presented by some figures (Figs. 6 - 8 and Figs. 7 - 9 of Appendix 1 of Supplementary Material) and some tables (Tables 1 - 7). And performance of algorithms is measured by time cost (second). Clustering quality of algorithms is measured by two robust information-theoretic measures, Adjusted Mutual Information (AMI) (Vinh et al., 2010) and Normalized Mutual Information (NMI) (Strehl et al., 2002), which are presented in Appendix 2 of Supplementary Material. According to the opinions of Vinh et al. (2010), the higher the value of the two measures gets, the better the clustering quality of algorithm is. In simulations, we use the Matlab code from Vinh et al. (2010) to compute the two clustering quality measures.

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

5.2.1 Compare the clustering results among SynC algorithm, ESynC algorithm, and SSynC algorithm

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

Table 1. Compare three different synchronization clustering algorithms (SynC, ESynC, and SSynC) by using four artificial data sets (DS1 - DS4). In Table 1, parameter δ = 18, the number of data points n = 10000, and parameter ε = 0.00001 in SSynC algorithm.

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
	SSynC	**52**	**69**	**34**	**52**
Iterative times	SynC	41	50	50	50
	ESynC, SSynC	**4**	**5**	**8**	**6**
The number of steady locations	SynC	254	379	260	431
	ESynC, SSynC	**14**	**5**	**25**	**8**

Note: The bold in Table 1 marks the better results of SSynC algorithm or ESynC algorithm.

5.2.2 Compare the clustering results among SynC algorithm, ESynC algorithm, SSynC algorithm, and some classical clustering algorithms

Table 2. Compare the clustering quality of several clustering algorithms (SynC, ESynC, SSynC, and some classical clustering algorithms) using six kinds of artificial data sets (DS2, DS4, DS5, DS6, DS7, and DS8). In Table 2, parameter δ = 18 in DS2, DS4, DS5, and DS6; parameter δ = 30 in DS7 and DS8; parameter ε = 0.00001 in SSynC algorithm.

(a)

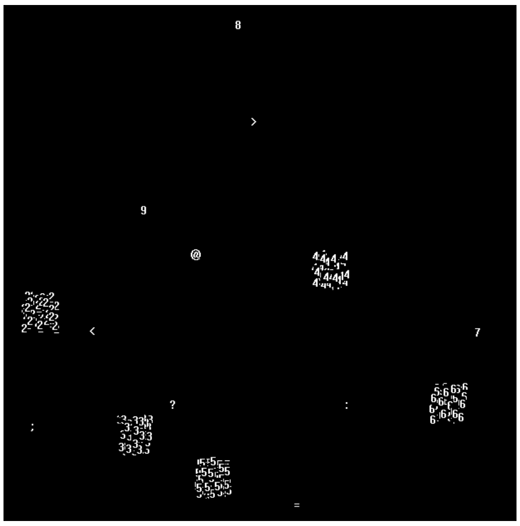
Measure indexes of algorithms	Name of algorithms	Data sets			
		DS2 (n = 400)	DS4 (n = 400)	DS2 (n = 800)	DS4 (n = 800)
NMI	SSynC, ESynC	**1.0000**	0.9694	**1.0000**	0.9643
	SynC	0.5505	0.6324	0.5362	0.6099
	K-Means	0.8670	0.9185	0.8659	**0.9682**
	FCM	**1.0000**	0.9633	**1.0000**	0.9615
	AP	0.7966	**0.9697**	0.7355	0.8375
	DBSCAN	**1.0000**	0.9643	**1.0000**	0.9643
	Mean Shift	0.7978	0.9028	0.7799	0.9103
AMI	SSynC, ESynC	**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	SSynC, ESynC	**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

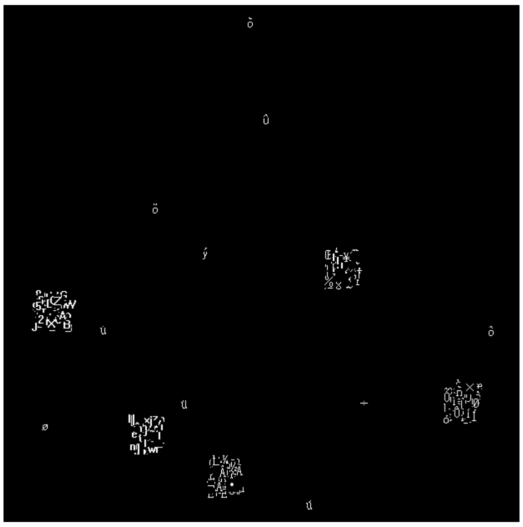
(b)

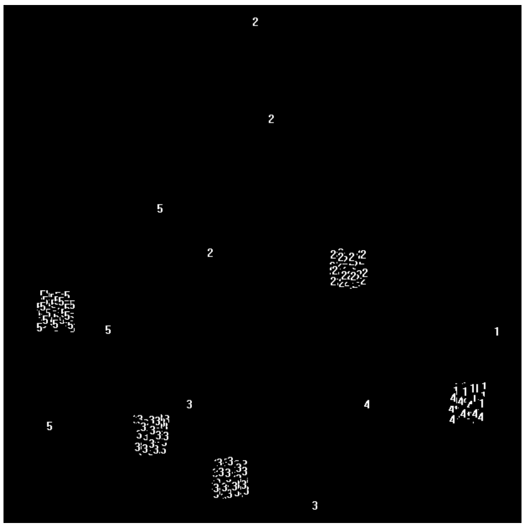
Measure indexes of algorithms	Name of algorithms	Data sets			
		DS5 (n = 10000)	DS6 (n = 10000)	DS7 (n = 10000)	DS8 (n = 10000)
NMI	SSynC, ESynC	0.9765	**1.0000**	**1.0000**	**1.0000**
	SynC	0.6231	0.5411	0.5205	0.5194
	K-Means	0.8872	NaN (Matlab)	0.9194	0.8437
	FCM	**0.9788**	0.5228	0.5226	0.5282
	DBSCAN	0.9765	**1.0000**	**1.0000**	**1.0000**
	Mean Shift	0.9708	**1.0000**	**1.0000**	**1.0000**
AMI	SSynC, ESynC	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	SSynC, ESynC	**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**

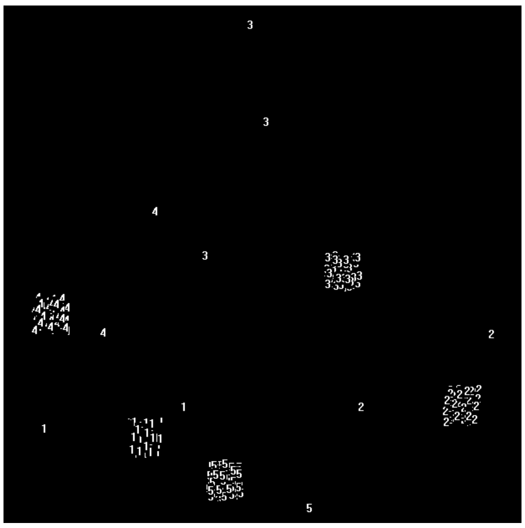
Note: NMI and AMI are two clustering quality measures presented in Strehl et al. (2002) and Vinh et al. (2010). In Table 2, the largest values of NMI and AMI and acceptable number of clusters in every data set are shown in bold.

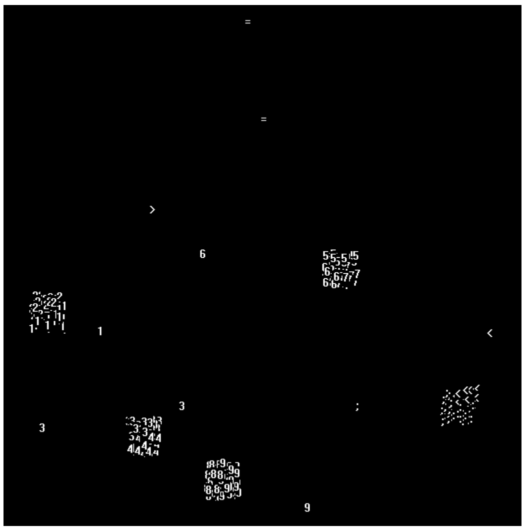
Table 2 presents the clustering quality of several clustering algorithms (SynC, ESynC, SSynC, and some classical clustering algorithms) by using six kinds of artificial data sets (DS2, DS4, DS5, DS6, DS7, and DS8). When computing the two information-theoretic measures (NMI and AMI), the predefined cluster labels of the eight artificial data sets are used in true_mem that is an input file of the MATLAB code (Vinh et al., 2010). In Table 2, by intercomparing SynC, ESynC, SSynC, and some classical clustering algorithms, we observe that SSynC and ESynC can get acceptable clustering results in the eight data sets. Because the three data sets (DS4 (n = 400), DS4 (n = 800), and DS5 (n = 10000)) have two connected clusters, SSynC and ESynC do not get the largest values of NMI and AMI.

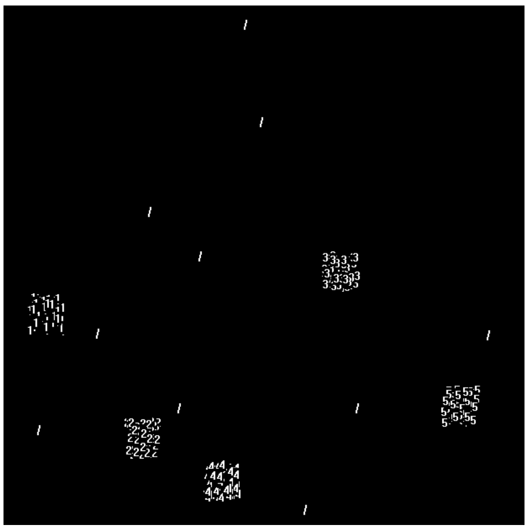

(a) Clusters identified by ESynC (15 clusters or isolates)

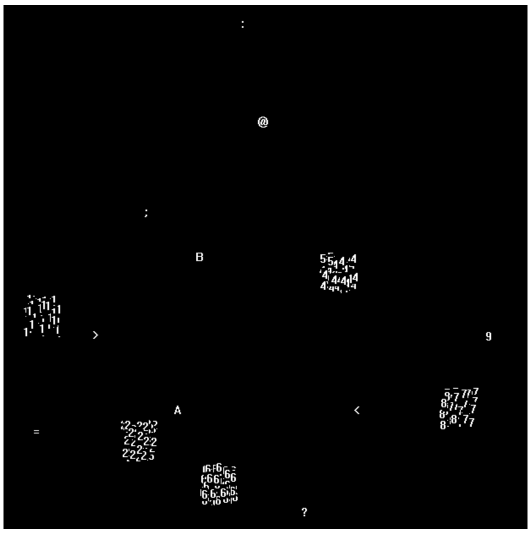
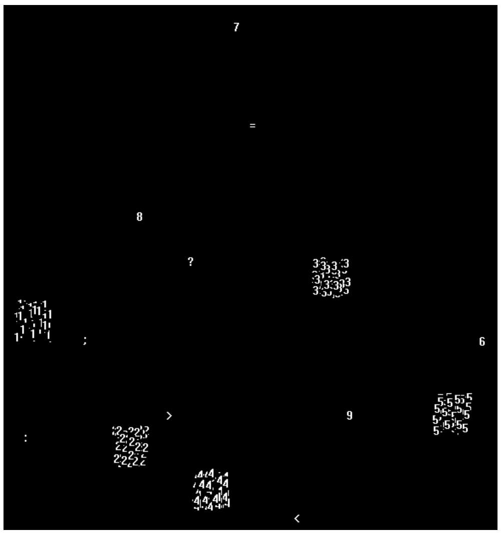

(b) Clusters identified by SynC (204 clusters or isolates)


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


(d) Clusters identified by FCM (predefined 5 clusters)


(e) Clusters identified by AP (14 clusters)


(f) Clusters identified by DBSCAN (5 clusters)

(g) Clusters identified by Mean Shift (18 clusters) (f) Clusters identified by SSynC (15 clusters or isolates)

Fig. 6. Compare the clustering results of several algorithms (DS1, n = 400)

Figs. 6 and Figs. 7 - 9 of Appendix 1 of Supplementary Material present the comparison clustering results of several clustering algorithms by some display figures that can reflect the clustering quality clearly. In Fig. 6 and Figs. 7 - 9 of Appendix 1 of Supplementary Material, parameter δ = 18 in SynC, ESynC, SSynC, DBSCAN, and Mean Shift; the number of data points n = 400; parameter ε = 0.00001 in SSynC algorithm.

From Fig. 6 and Figs. 7 - 9 of Appendix 1 of Supplementary Material, we observe that SSynC and ESynC 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 SSynC and ESynC for some artificial data sets (from DS1 - DS4). Especially, SynC, ESynC, and SSynC can all easily find some isolates if setting a proper value for parameter δ, and SSynC gets the same clustering results (the same clusters displayed by figures) with ESynC in these data sets.

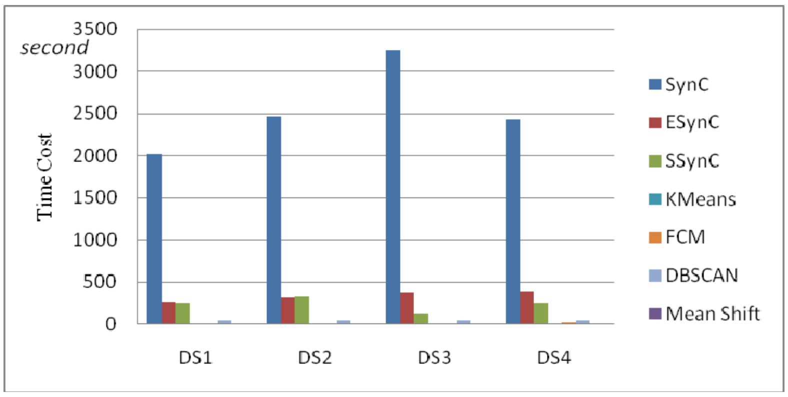


Fig. 7. Compare several clustering algorithms in time cost by using four artificial data sets (DS1 - DS4, n = 20000).

Fig. 7 presents the comparison results of several clustering algorithms in time cost. In Fig. 7, parameter δ = 18 in SynC, ESynC, SSynC, DBSCAN, and Mean Shift;

the number of data points $n = 20000$; parameter $\varepsilon = 0.00001$ in SSynC algorithm.

In Fig. 7, intercomparing SynC, ESynC, SSynC, Mean Shift, DBSCAN, FCM, and K-Means, we observe that SSynC is faster than SynC and ESynC in many cases, and K-Means is the fastest clustering algorithm.

5.2.3 Compare the valid interval of parameter δ among SynC, ESynC, SSynC, DBSCAN, and Mean Shift using some artificial data sets (DS5 - DS16)

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

Table 3 gives the comparison results among these clustering algorithms. Here, $[e_k, e_{k+1}]$ can be obtained from Eq.(8) of Chen (2015). In Table 3, intercomparing SynC, ESynC, SSynC, DBSCAN, and Mean Shift, we observe that the valid interval of parameter δ in SSynC and ESynC is longer than it in DBSCAN in these data sets, the valid interval of parameter δ in DBSCAN is consistent with $[e_k, e_{k+1}]$, and parameter δ in Mean Shift has the longest and largest valid interval.

Table 4 compares the valid interval of parameter δ in SSynC algorithm for several different value of parameter ε using some artificial data sets with different dimensions. In Table 4, intercomparing several different value of parameter ε, we observe that the valid interval of parameter δ has very small difference for several different value of parameter ε if parameter ε is less than parameter δ.

Table 3. Compare the valid interval of parameter δ among SynC, ESynC, SSynC, DBSCAN, and Mean Shift using some artificial data sets with different dimensions. In Table 3, $n = 10000$, parameter $\varepsilon = 0.00001$ in SSynC algorithm.

(a)

		Data sets			
		DS5	DS6	DS7	DS8
The valid interval of parameter δ	SynC	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$
	SSynC, ESynC	$\delta \in [9, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	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 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]

(b)

		Data sets			
		DS9	DS10	DS11	DS12
The valid interval of parameter δ	SynC	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$
	SSynC, ESynC	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	DBSCAN	$\delta \in [2, 68]$	$\delta \in [6, 193]$	$\delta \in [11, 232]$	$\delta \in [15, 279]$
	Mean Shift	$\delta \in [14, 89]$	$\delta \in [15, 219]$	$\delta \in [19, 261]$	$\delta \in [21, 312]$
$[e_k, e_{k+1}]$ In MST of the complete graph of the data set		[1.36, 68.69]	[6.89, 193.04]	[6.89, 193.04]	[18.47, 279.44]

(c)

		Data sets			
		DS13	DS14	DS15	DS16
The valid interval of parameter δ	SynC	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$	$\delta \in \emptyset$
	SSynC, ESynC	$\delta \in [40, 854]$	$\delta \in [63, 1271]$	$\delta \in [87, 1850]$	$\delta \in [123, 2917]$
	DBSCAN	$\delta \in [34, 841]$	$\delta \in [57, 1257]$	$\delta \in [90, 1841]$	$\delta \in [135, 2908]$
	Mean Shift	$\delta \in [40, 872]$	$\delta \in [65, 1283]$	$\delta \in [92, 1864]$	$\delta \in [136, 2935]$
$[e_k , e_{k+1}]$ In MST of the complete graph of the data set		[39.69, 841.37]	[64.05, 1257.35]	[97.34, 1841.97]	[142.44, 2908.82]

Table 4. Compare the valid interval of parameter δ in SSynC algorithm for several different value of parameter ε using some artificial data sets with different dimensions. In Table 4, n = 10000, parameter ε is set as several different value respectively in SSynC algorithm.

(a) DS5 - DS8

		Data sets			
		DS5	DS6	DS7	DS8
The valid interval of parameter δ in SSynC algorithm for several different value of parameter ε	$\varepsilon = 0.00001$	$\delta \in [9, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	$\varepsilon = 0.0001$	$\delta \in [9, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	$\varepsilon = 0.001$	$\delta \in [9, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	$\varepsilon = 0.01$	$\delta \in [9, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	$\varepsilon = 0.1$	$\delta \in [13, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 214]$	$\delta \in [22, 298]$
	$\varepsilon = 1$	$\delta \in [12, 58]$	$\delta \in [11, 164]$	$\delta \in [16, 215]$	$\delta \in [22, 298]$
	$\varepsilon = 10$	$\delta \in [14, 22]$	$\delta \in [16, 161]$	$\delta \in [17, 215]$	$\delta \in [22, 298]$

(b) DS9 - DS12

		Data sets			
		DS9	DS10	DS11	DS12
The valid interval of parameter δ in SSynC algorithm for several different value of parameter ε	$\varepsilon = 0.00001$	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	$\varepsilon = 0.0001$	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	$\varepsilon = 0.001$	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	$\varepsilon = 0.01$	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	$\varepsilon = 0.1$	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	$\varepsilon = 1$	$\delta \in [9, 83]$	$\delta \in [10, 208]$	$\delta \in [13, 248]$	$\delta \in [19, 297]$
	$\varepsilon = 10$	$\delta \in [14, 83]$	$\delta \in [16, 208]$	$\delta \in [16, 249]$	$\delta \in [19, 297]$

(c) DS13 - DS16

		Data sets			
		DS13	DS14	DS15	DS16
The valid interval of parameter δ in SSynC algorithm for several different value of parameter ε	$\varepsilon = 0.01$	$\delta \in [40, 854]$	$\delta \in [63, 1271]$	$\delta \in [87, 1850]$	$\delta \in [123, 2917]$
	$\varepsilon = 0.1$	$\delta \in [40, 854]$	$\delta \in [63, 1271]$	$\delta \in [87, 1850]$	$\delta \in [123, 2917]$
	$\varepsilon = 1$	$\delta \in [40, 854]$	$\delta \in [63, 1271]$	$\delta \in [87, 1851]$	$\delta \in [123, 2917]$
	$\varepsilon = 10$	$\delta \in [40, 854]$	$\delta \in [63, 1271]$	$\delta \in [87, 1851]$	$\delta \in [123, 2917]$
	$\varepsilon = 20$	$\delta \in [40, 854]$	$\delta \in [63, 1271]$	$\delta \in [87, 1851]$	$\delta \in [123, 2918]$
	$\varepsilon = 30$	$\delta \in [41, 854]$	$\delta \in [64, 1271]$	$\delta \in [87, 1851]$	$\delta \in [125, 2919]$
	$\varepsilon = 40$	$\delta \in [42, 854]$	$\delta \in [65, 1271]$	$\delta \in [89, 1851]$	$\delta \in [125, 2917]$

Note: SSynC algorithm gets 12 clusters when parameter δ in its valid interval**.** In the DS5 (n = 10000) data set, there are two clusters that are almost connected to become one cluster, so parameter ε affects the final number of clusters very much. For other data sets, parameter ε affects the final number of clusters little.

5.3 Experimental results of several 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 the 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 true_mem that is an input file of the MATLAB code (Vinh et al., 2010).

5.3.1 Compare the clustering results among SynC algorithm, ESynC algorithm, and SSynC algorithm

Table 5 gives the comparison results of three synchronization clustering algorithms (SynC, ESynC, and SSynC) by using several UCI data sets. In Table 5, by comparing SynC, ESynC, and SSynC, we observe that SSynC and ESynC can get better local synchronization results than SynC in the eight UCI data sets, and SSynC is the fastest algorithm.

Table 5. Compare three synchronization clustering algorithms (SynC, ESynC, and SSynC) by using several UCI data sets. In Table 5, parameter $\varepsilon = 1$ in SSynC algorithm.

(a) The setting of parameter δ in three synchronization clustering algorithms for several UCI data sets

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
	SSynC	0	0	**1**	0
Iterative times	SynC	50	50	50	50
	SSynC, ESynC	**9**	**6**	**7**	**6**
The number of steady locations	SynC	147	178	569	213
	SSynC, 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**
	SSynC	**0**	**0**	**0**	**0**
AveLen(*T*)	SynC	83.9640	258.3664	276.6775	97.9706
	SSynC, 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	4186	1	79
	ESynC	**1**	2270	**0**	10
	SSynC	**1**	**394**	**0**	**4**
Iterative times	SynC	50	50	50	50
	SSynC, ESynC	**9**	**23**	**7**	**6**
The number of steady locations	SynC	350	18668	210	2043
	SSynC, 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**
	SSynC	**0**	**0**	**0**	**0**
$AveLen(T)$	SynC	401.6912	171.9401	142.6595	215.9900
	SSynC, ESynC	**0**	**0**	**0**	**0**

Note: The bold in Table 5 marks the better results of SSynC algorithm or ESynC algorithm.

5.3.2 Compare the clustering results among SynC algorithm, ESynC algorithm, SSynC algorithm, and some classical clustering algorithms

Table 6. Compare the clustering quality of several clustering algorithms (SynC, ESynC, SSynC, and some classical clustering algorithms) by using several UCI data sets. In Table 6, parameter $\varepsilon = 1$ in SSynC algorithm.

(a) The setting of parameter δ in several clustering algorithms for several UCI data sets

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	SSynC, 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	SSynC, 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	SSynC, 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)

(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	SSynC, 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	SSynC, 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	SSynC, 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 the Letter-recognition data set, DBSCAN algorithm obtains 21 clusters and 243 isolates when parameter δ = 160.0001, so we set parameter δ = 160 in DBSCAN. The sign '-' in AP column means that the time cost is too larger.

Note2: In Table 6, the largest values of NMI and AMI in every data set are shown in bold.

Table 6 gives the comparison clustering quality of several clustering algorithms (SynC, ESynC, SSynC, and some classical clustering algorithms) by using several UCI data sets. In Table 6, by intercomparing these clustering algorithms, we observe that SSynC and ESynC do 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 for the eight UCI data sets without any actual knowledge on these data sets. Second, we observe that the largest values of NMI and AMI do not mean the best clustering quality for some data sets. 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 have not better choice). From the final number of clusters of Table 6, we observe that SSynC and ESynC can get better local synchronization results than some other clustering algorithms for 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 are in a range [0, 255] in each dimension. In Table 7 and Fig. 8, parameter ε = 1 in SSynC algorithm.

5.4.1 Compare the clustering results among SynC algorithm, ESynC algorithm, and SSynC algorithm

Table 7. Compare three different synchronization algorithms (SynC, ESynC, and SSynC) by using three picture data sets. In Table 7, parameter δ = 18 or 30 in SynC, ESynC, and SSynC; parameter ε = 1 in SSynC algorithm.

(a). parameter δ = 18

Measure indexes of algorithms	Name of algorithms	Data sets		
		Picture1	Picture2	Picture3
Spend time (second)	SynC	662	676	9795
	ESynC	132	122	3254
	SSynC	**18**	**16**	**297**
Iterative times	SynC	50	50	50
	SSynC, ESynC	**10**	**9**	**16**
The number of steady locations	SynC	941	467	2868
	SSynC, ESynC	**13**	**5**	**14**
The cluster order parameter r_c	SynC	58.6149	118.4821	88.4415
	ESynC	**2712.8392**	**3321.3298**	**6127.5541**
	SSynC	**0**	**0**	**0**
AveLen(*T*)	SynC	11.0537	10.5757	11.5605
	SSynC, ESynC	**0**	**0**	**0**

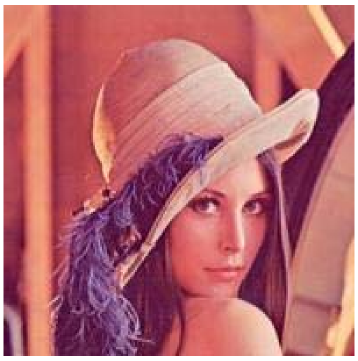
(b). parameter δ = 30

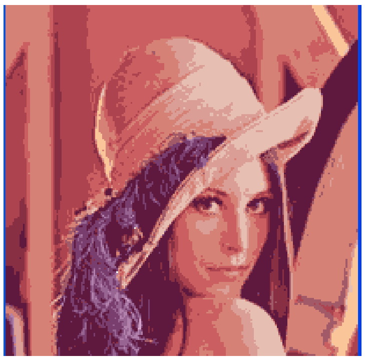
Measure indexes of algorithms	Name of algorithms	Data sets		
		Picture1	Picture2	Picture3
Spend time (second)	SynC	749	797	10930
	ESynC	122	179	2139
	SSynC	**16**	**16**	**274**
Iterative times	SynC	50	50	50
	SSynC, ESynC	**9**	**13**	**10**
The number of steady locations	SynC	928	472	2896
	SSynC, ESynC	**4**	**2**	**6**
The cluster order parameter r_c	SynC	55.2653	106.8353	87.9900
	ESynC	**3630.5206**	**5015.0178**	**11105.6154**
	SSynC	**0**	**0**	**0**
AveLen(*T*)	SynC	16.9417	17.5013	19.0378
	SSynC, ESynC	**0**	**0**	**0**

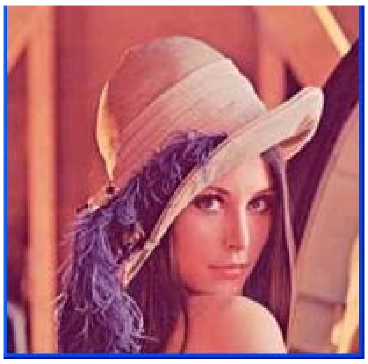
Note: The bold in Table 7 marks the better results of SSynC algorithm or ESynC algorithm.

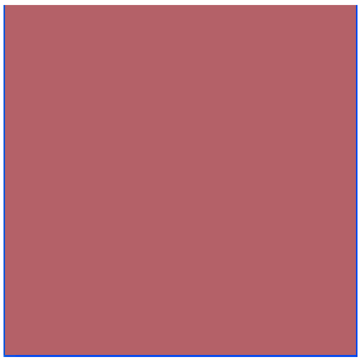
Table 7 is the experimental results in time cost and local synchronization results among SynC, ESynC, and SSynC by clustering pixel points of three bmp picture in RGB color space. In Table 7, by comparing SynC, ESynC, and SSynC, we observe that SSynC and ESynC are faster than SynC for these data sets. At the same time, SSynC and ESynC can get better local synchronization results than SynC in these data sets.

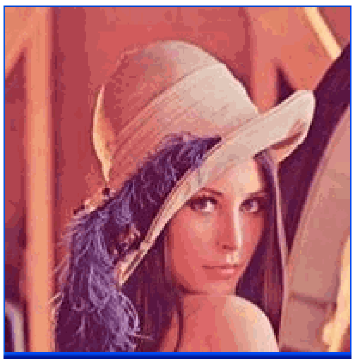
5.4.2 Compare the clustering compress results among SynC algorithm, ESynC algorithm, SSynC algorithm, and some classical clustering algorithms

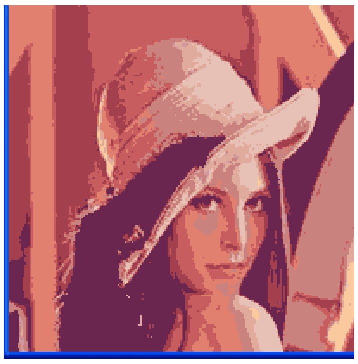

Origina Picture


SSynC, ESynC (final k = 14)

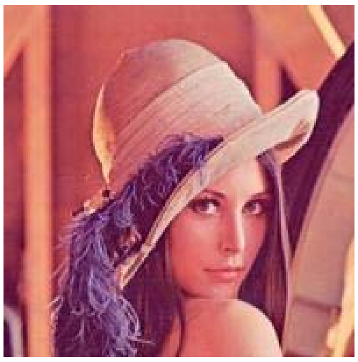

SynC (final k = 2868)

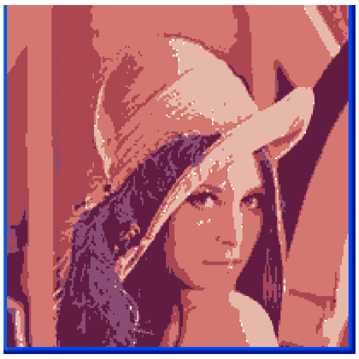

K-Means, FCM (final k = 1)

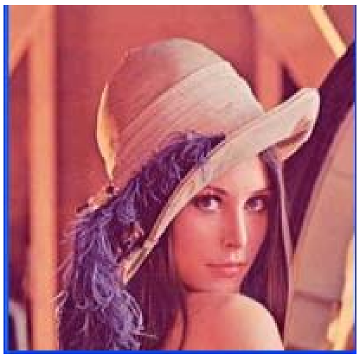

DBSCAN (final k = 112)

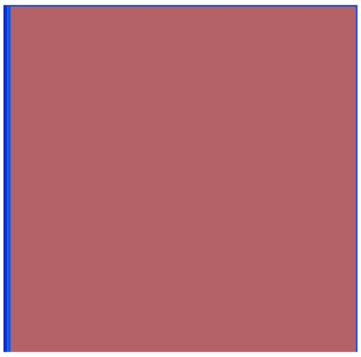

Mean Shift (final k = 10)

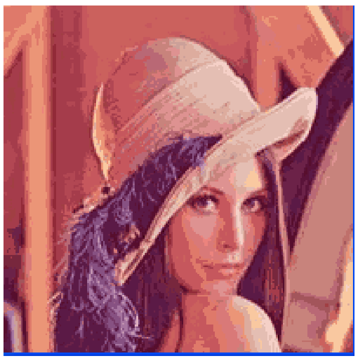
(a) δ = 18 for SynC, ESynC, SSynC, DBSCAN, and Mean Shift; predefined k (number of clusters) = 14 for K-Means and FCM.

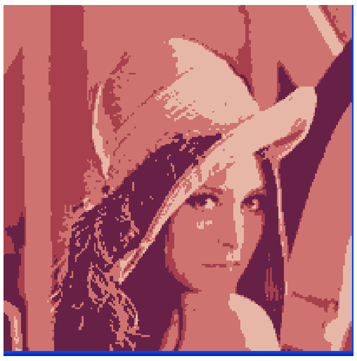

Origina Picture


SSynC, ESynC (final k = 6)


SynC (final k = 2896)


K-Means, FCM (final k = 1)


DBSCAN (final k = 35)


Mean Shift (final k = 4)

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

Fig. 8. Compare the original picture and several compressed pictures of Picture3 by clustering pixel

points of Picture3 in RGB color space using several algorithms. In Fig. 8, several compressed pictures are drawn using the means of clusters obtained by clustering 200 * 200 pixel points of Picture3 in RGB space.

Fig. 8 lists the original picture and several compressed pictures of Picture3. The several compressed pictures are drawn using the means of clusters obtained by clustering 200 * 200 pixel points of Picture3 in RGB color space using different algorithms. Because AP needs too much time and space for Picture3, this experiment does not use it. From Fig. 8, we observe that SSynC and ESynC 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 - 8, Figs. 7 - 9 of Appendix 1 of Supplementary Material, and Tables 1 - 7), we observe that SSynC algorithm is faster than ESynC algorithm and SynC algorithm almost in all cases. In simulations of some artificial data sets (from DS5 to DS16), we observe that the effective interval of parameter δ in SSynC and ESynC has a long range, and in many cases it is longer than the effective interval of parameter δ (or Eps) in DBSCAN.

In some displayed figures, by intercomparing SynC, ESynC, and SSynC, we observe that SSynC can explore the same clusters and isolates (displayed by some figures) with ESynC. For many kinds of data sets, SSynC and ESynC can explore obvious clusters or isolates if setting a proper value of parameter δ, and SynC cannot explore obvious clusters of many data sets.

In simulations of some data sets, we observe that the iterative times of SynC, AP, K-Means, and FCM is larger than the iterative times of SSynC and ESynC. For many data sets, ESynC, SSynC, 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, SSynC algorithm and ESynC algorithm can get more obvious multi-level compress effect than some other algorithms, such as K-Means and FCM. In simulations, we also observe that DBSCAN algorithm needs more space than SSynC algorithm and ESynC algorithm because of its recursion procedure.

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

6. Conclusions

This paper presents an improved synchronization clustering algorithm, SSynC, which often gets better clustering results than the original synchronization clustering algorithm, SynC. From the experimental results, we observe that SSynC algorithm can often obtain less iterative times, faster clustering speed, and better clustering quality than SynC algorithm for many kinds of data sets. SSynC algorithm can also get better or similar clustering results or faster clustering speed than some classical clustering algorithms for some data sets.

To our knowledge, the linear weighted Vicsek model and shrinking synchronization clustering are introduced firstly. The major contributions of this paper can be summarized as follows:

(1). It presents a Shrinking Synchronization Clustering (SSynC) algorithm, which is an improved version of SynC algorithm, by using a linear weighted Vicsek model.

(2). It validates the improved effect of SSynC algorithm in time cost and clustering quality by the simulated experiments of several different kinds of data sets.

(3). It presents and validates our convergent condition of dynamical clustering in SSynC algorithm, the t-step average length of edges, by the simulated experiments of several different kinds of data sets.

SSynC algorithm uses a global searching strategy to construct the δ near neighbor point set for every point in each evolution, so its time complexity is $O(d\cdot(n_{(t=0)}^2 + n_{(t=1)}^2) + \ldots + n_{(t=T-1)}^2))$, which is less than $O(Tdn^2)$ that is the time complexity of SynC algorithm. Where n is the number of all points, $n_{(t)}$ is the number of synchronized points in the t-step synchronization, d is the number of dimensions, and T is the times of synchronization.

Like DBSCAN, SynC, and ESynC, SSynC algorithm is also robust to outliers or isolates. Like DBSCAN and ESynC, SSynC can find obvious clusters with different shapes. For DBSCAN, Mean Shift, ESynC, and SSynC, 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 an exploring method listed in Chen (2015) or using the MDL-based method presented in Böhm et al. (2010). More often, the valid interval of parameter δ in ESynC and SSynC is longer than it in DBSCAN. Comparing with SynC, K-Means, FCM, and AP, ESynC and SSynC can obtain better or similar clustering quality.

Although our algorithm has shown promising results, there are still some limitations. First, the computational complexity of SSynC is $O(d\cdot(n_{(t=0)}^2 + n_{(t=1)}^2) + \ldots + n_{(t=T-1)}^2))$, which limits its applicability to big data. Second, like DBSCAN, ESynC, and CNNI (Chen, 2015), SSynC is also sensitive to parameter δ for some scatter data sets. When many noises and few obvious clusters exist, DBSCAN, ESynC, and SSynC cannot generate clusters with different levels of scatter because parameter δ is fixed before clustering.

This work opens some possibilities for further improvement and investigation. First, do more comparative experiments. For example, in the process of constructing δ near neighbor point sets, comparing the simplest search method with our improved method based on δ near neighbor grid cell set (Chen, 2013) and Red-Black tree, R-tree index structure (Guttman, 1984; Manolopoulos et al., 2006) method, and SR-tree index structure (Katayama et al., 1997) method should be valuable for practical work. Second, further improve SSynC algorithm in time cost. For example, designing similarity preserving Hash function that needs $O(1)$ time complexity is valuable in the process of constructing δ near neighbor point sets. Third, extend the applicability and explore the clustering effect of our algorithm in high-dimensional data. Fourth, further explore more proper and simple methods to estimate parameter δ. Fifth, SSynC algorithm is a dynamic synchronization clustering algorithm, and Mean shift algorithm (Fukunaga et al., 1975; Comaniciu et al., 2002) is a clustering algorithm based on a non-parametric modeling method. Although SSynC algorithm and Mean shift algorithm have essential difference, they still have some similarity. So, it is important to explore the relation between SSynC algorithm and Mean shift algorithm further.

Acknowledgments

This work was supported by three projects (grant No: cstc2016jcyjA0521, cstc2014jcyjA40035, cstc2016jcyjA0063) from Chongqing Cutting-edge and Applied Foundation Research Program of China and a project (grant No: 16PY08) from Chongqing Three Gorges University of China.

References

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

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

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

Böhm, C., Plant, C., Shao, J., et al. (2010). Clustering by synchronization. Proceedings of ACM SIGKDD. Washington, USA: 583–592.

Chen, X. (2013). Clustering based on a near neighbor graph and a grid cell graph. Journal of Intelligent Information Systems, 40(3): 529-554.

Chen, X. (2014). A fast synchronization clustering algorithm. arXiv:1407.7449 [cs.LG]. http://arxiv.org/abs/1407.7449.

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

Chen, X. (2017). An effective synchronization clustering algorithm. Applied Intelligence, DOI: 10.1007/s10489-016-0814-y.

Chen, Z., Zhang, H. T., Chen, X., Chen, D.,& Zhou, T. (2015). Two-level leader-follower organization in pigeon flocks. Europe physics Letters,112(2), 20008.

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

Czirok, A., Barabasi, A. L., & Vicsek, T. (1999). Collective motion of self-propelled particles: kinetic phase transition in one dimension. Physical Review Letters, 82: 209–212.

Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial data sets with noise. The 2-th International Conference on Knowledge Discovery and Data Mining: 226-231.

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

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

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

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

Guha, S., Rastogi, R., and Shim, K. (1998). CURE: An efficient clustering algorithm for clustering large databases, in: Proceedings of the Symposium on Management of Data (SIGMOD).

Horn, D., & Gottlieb, Assaf. (2002). Algorithm for data clustering in pattern recognition problems based on quantum mechanics. Physical Review Letters, 88(1), 018702-018701–018702-018702.

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

Jadbabaie, A., Lin, J., Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans Automat Control, 48(6): 998–1001.

Jaromczyk, J. W., & Godfried, T. (1992). Relative neighborhood graphs and their relatives. In Proceedings of the IEEE, 80(9), 1502–1517.

Ji, P., Peron, T. K., Menck, P. J., Rodrigues, F. A., & Kurths, J. (2013). Cluster explosive synchronization in complex networks. Physical Review Letters, 110(21).

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

Leyva, I., Navas, A., Sendiña-Nadal, I., et al. (2013). Explosive transitions to synchronization in networks of phase oscillators. Science Reports, 3: 1281.

Liu, Z., Guo, L. (2008). Connectivity and synchronization of Vicsek model. Science in China Series F: Information Sciences, 51(7): 848–858.

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

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. The 5-th MSP Proceeding. Berkeley, University of California Press: 281-297.

Nagy, M., Ákos, Z., Biro, D., & Vicsek,T. (2010). Hierarchical group dynamics in pigeon flocks. Nature, 464 (7290), 890-893.

Reynolds, C. (1987). Flocks, birds, and schools: a distributed behavioral model."Computer Graphics 21: 25–34.

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

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

Schaeffer, S. E. (2007). Graph clustering. Computer Science Review, 1(1), 27-64.

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

Shao, J., Yang, Q., Böhm, C, & Plant, C. (2011). Detection of arbitrarily oriented synchronized clusters in high-dimensional data. In ICDM: 607-616.

Shao, J., Hahn, K.,Yang, Q., et al. (2010). Hierarchical density-based clustering of white matter tracts in the human brain. International Journal of Knowledge Discovery in Bioinformatics, 1(4): 1-25.

Shao, J., He, X., Plant, C., Yang, Q., & Böhm, C. (2013). Robust synchronization-based graph clustering. The 17-th Pacific-Asia Conference on Knowledge Discovery and Data Mining: 249-260.

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

Shao, J., Ahmadi, Z., Kramer, S. (2014). Prototype-based learning on concept-drifting data streams. Proceedings of ACM SIGKDD: 412-421.

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

Theodoridis, S., & Koutroumbas, K. (2006). Pattern recognition, Academic Press.

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

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

Wang, L., & Liu, Z. (2009). Robust consensus of multi-agent systems with noise.

Science in China Series F: Information Sciences, 52(5): 824–834.

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

Zhang, H. T., Chen, Z., Vicsek, T., Feng,G., Sun, L., Su, R., & Zhou, T. (2014). Route-dependent switch betweenhierarchical and egalitarian strategies in pigeon flocks. Scientific Reports, 4, 5805.

Zhang, T., Ramakrishnan R., and Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases, in: SIGMOD Conference, pp. 103 – 114.

Zou, Y., Pereira, T., Small, M., Liu, Z., & Kurths, J. (2014). Basin of attraction determines hysteresis in explosive synchronization. Physical Review Letters, 112(11): 114102.