2011 11th IEEE International Conference on Data Mining

Detection of Arbitrarily Oriented Synchronized Clusters in High-dimensional Data

Junming Shao*, Claudia Plant†, Qinli Yang‡ and Christian Böhm*
*Institute for Computer Science, University of Munich, Munich, Germany
*Department of Scientific Computing, Florida State University, Tallahassee, FL, USA.
‡School of Engineering, University of Edinburgh, Edinburgh, UK

***Abstract*—How to address the challenges of the "curse of dimensionality" in clustering? Clustering is a powerful data mining technique for structuring and organizing vast amounts of data. However, the high-dimensional data space is usually very sparse and meaningful clusters can only be found in lower dimensional subspaces. In many applications the subspaces hosting the clusters provide valuable information for interpreting the major patterns in the data. Detection of subspace clusters is challenging since usually many of the attributes are noisy, some attributes may exhibit correlations among each other and only few of the attributes truly contribute to the cluster structure. In this paper, we propose ORSC (Arbitrarily ORiented Synchronized Clusters), a novel effective and efficient method to subspace clustering inspired by synchronization. Synchronization is a basic phenomenon prevalent in nature, capable of controlling even highly complex processes such as opinion formation in a group. Control of complex processes is achieved by simple operations based on interactions between objects. Relying on the interaction model for synchronization, our approach ORSC (1) naturally detects correlation clusters in arbitrarily oriented subspaces, including (2) arbitrarily-shaped non-linear correlation clusters. Our approach is (3) robust against noise points and outliers. In contrast to previous methods, ORSC is (4) easy to parameterize, since there is no need to specify the subspace dimensionality and all interesting subspace clusters can be detected. Finally,(5) ORSC outperforms most comparison methods in terms of runtime efficiency and is highly scalable to large and high-dimensional data sets.**

***Keywords*-subspace clustering; synchronization; interaction model; high-dimensional data;**

I. Introduction

Nowadays, tremendous amounts of data in a large variety of application domains are produced. These real-world data sets are often represented as sparse, high-dimensional feature vectors, contaminated by outliers and noise objects. Clustering such high-dimensional data becomes difficult for traditional clustering methods due to the famous problem called "curse of dimensionality". To cure the problem in clustering, the concept of *subspace clustering* has been introduced to detect clusters in subspaces of the original space.

Currently, a number of subspace clustering algorithms have already been proposed, which can be mainly distinguished as axis-parallel subspaces clustering, e.g. CLIQUE [5], PROCLUS [3], SUBCLU[11] and arbitrarily oriented subspace clustering (also called generalized subspace clustering, or correlation clustering), e.g. ORCLUS [4], 4C [8], Curler [19]. However, most of these established algorithms fail to discover clusters of complex shapes and various densities. Another challenge for subspace clustering is the search strategy for the suitable subspaces. The number of possible axis-parallel subspaces is exponential (2^d) in the number of dimensions d, and the number of arbitrarily oriented subspaces is even infinite. Therefore, a complete enumeration of all possible subspaces to be checked for clusters is not feasible. Consequently, all previous solutions rely on specific assumptions and heuristics, and try to find promising subspaces during the clustering process, for instance in an iterative optimization.

To deal with these problems, in this paper, we consider subspace clustering from a new perspective: *synchronization*. Here, let us first illustrate some fundamental concepts for synchronization-based subspace clustering.

A. Synchronization and Synchronized Cluster

Synchronization is a prevalent phenomenon in nature that a group of events spontaneously come into co-occurrence with a common rhythm, despite of the differences between individual rhythms of the events. It is known that synchronization is rooted in human life from the metabolic processes in our cells to the highest cognitive tasks we perform as a group of individuals [6]. A paradigmatic example of synchronization phenomena in nature is the synchronous flashing of fireflies observed in some South Asian forests [1]. At night, in the beginning, they flash independently, but after a short period of time the whole swarm is flashing in unison, creating one of the most striking visual effects ever seen. During last several decades, synchronization has attracted a large volume of interest in physics, biology, ecology, sociology and other fields of science and technology. For example, Rhouma et al. [17] introduce an efficient synchronization model that organizes a population of integrate-and-fire oscillators into stable and structured groups. Arenas et al. [7] investigate the synchronization phenomena for network analysis, and study the relationship between topological scales and dynamic time scales in complex networks. Aeyels et al. [2] introduce a mathematical model for investigating the dynamics of chaos system and apply it to a system of interconnected water basins. Recently, Böhm et al. [9],[18] propose an extensive

1550-4786/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDM.2011.50

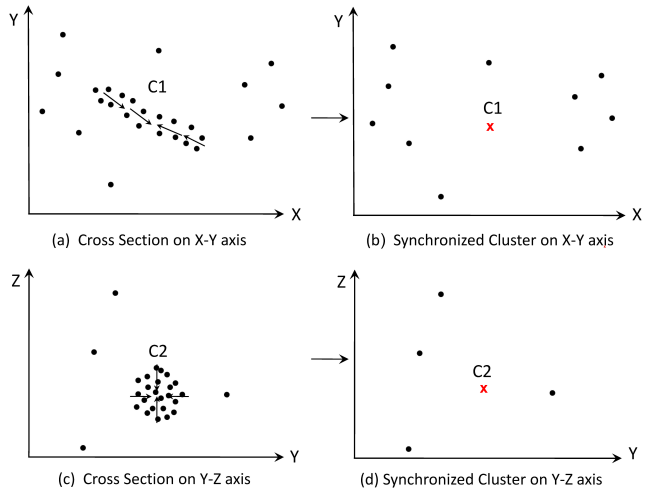


Figure 1. Illustration of Synchronized Clusters. Arrows indicate the main directions of movements of objects during the process towards synchronization and the red cross points illustrate the final states of cluster objects, which are formed as synchronized clusters in subspaces.

Kuromato model to simulate the dynamics of each object during the process of synchronization and explore clusters and outliers in combination with Minimum Description Length.

Inspired by these synchronization phenomena and models, we propose a novel method to subspace clustering. The key idea is to consider subspace clustering as a dynamic process towards synchronization. We view each object as an oscillator which interacts with other objects in arbitrarily oriented subspaces. Through interactions, all cluster objects in arbitrarily oriented subspaces will synchronize together, which called *synchronized clusters* in this paper. Let us demonstrate the concept with the help of a simple 3-dimensional data set. Fig. 1(a) and Fig. 1(c) represent the X-Y and Y-Z projection of the data, respectively. In Cluster $C1$, the X and Y dimensions are strongly correlated and the Z-dimension is not cluster-specific white noise. Cluster $C2$ only exists in the Y-Z space and the X dimension is noise. To find such clusters, in this context, like oscillators, each object interacts with similar objects in axis-parallel or arbitrarily oriented subspaces. Objects will gradually change their states and move to other objects driving by its intrinsic structure, which we will elaborate in detail in Section III. The arrows in Fig. 1(a) and Fig. 1(c) illustrate the main directions of movements for objects. Fig. 1(b) and Fig. 1(d) further indicate the final states of objects after synchronization. Finally, all objects of a common subspace cluster move together and form as synchronized clusters (cf. Fig. 1(b) and Fig. 1(d)).

B. Contributions

Inherited by the concept of synchronization, in this paper, we propose a new subspace clustering approach, ORSC (arbitrarily ORiented Synchronized Clusters). The major benefits of our algorithm ORSC can be summarized as follows:

1) *Natural data structure exploring.* The synchronized cluster formation is driven by the mutual interactions among objects relying on intrinsic data structure.
2) *Detection of arbitrarily shaped correlation clusters.* Without any data distribution assumption, ORSC allows detecting clusters with arbitrary numbers, shapes and sizes in subspaces.
3) *Outlier robustness.* Since the outliers cannot easily synchronize with other objects, they can be effectively distinguished from cluster objects.
4) *Efficient subspace searching.* Inherited by the properties of synchronization, the subspace search strategy of ORSC does not need scan possible subspaces but simply, finds all subspace clusters by searching all synchronized phases.

The remainder of this paper is organized as follows: we briefly survey related work in Section II. Section III presents our new concept and algorithm in detail. Section IV contains an extensive experimental evaluation and Section V concludes the paper.

II. Related Work

As most traditional clustering algorithms fail to detect clusters embedded in high-dimensional data, various subspace clustering approaches have been studied. Subspace clustering algorithms like CLIQUE [5] ENCLUS [10], projected clustering algorithms such as PROCLUS [3], DOC [16], and SUBCLU [11] only find axis-parallel clusters. Pattern based subspace clustering methods (e.g. [20],[14]) aim to group objects that exhibit a similar trend in a subset of attributes into clusters rather than objects with low distance. However, they limit themselves to finding only clusters that represent pairwise positive correlations in the data set. Here, we focus on the more recent correlation clustering algorithms which can find clusters in arbitrarily oriented subspaces. Currently, many arbitrarily oriented subspace clustering algorithms such as ORCLUS [4], 4C [8], Curler [19] have been proposed.

ORCLUS [4] is one of the iterative top-down search methods for arbitrarily projected subspaces. It integrates PCA into k-means and includes three steps: assign clusters, determinate subspaces and merge them. However, it requires users to specify the number of clusters and the subspace dimensionality in advance. ORCLUS tends to fail if the estimation does not match with the actual number of clusters. Moreover, due to using random sampling to improve computation speed and scalability, it may suffer from missing

smaller clusters. 4C [8] combines PCA and density-based clustering to identify local subgroups of the data objects sharing a uniform but arbitrarily complex correlation. It formalizes the correlation connected cluster as a dense region of points in the d-dimensional feature space having at least one principal axis with low variation along this axis. Like ORCLUS, 4C limits itself to identify linear correlation clusters without considering nonlinear correlation clusters. Actually, in real-life datasets, correlation between attributes could however be nonlinear. To address the nonlinear issue, Curler [19] is proposed which allows detecting both global and local orientations of the clusters. This method merges the microclusters generated by the EM variant algorithm according to their co-sharing level. Therefore, the resulting clusters may represent a more complex, not necessarily liner correlation. However, the performance of Curler is very sensitive to noise and strongly depends on the suitable parametrization.

III. The Algorithm ORSC

In this section, we introduce ORSC to discover all possible clusters in arbitrarily oriented subspaces. We first give an overview of our novel perspective of subspace clustering and then construct the weighted interaction model to simulate the dynamics of objects towards synchronization. Finally an efficient searching strategy is used to find all synchronized clusters.

A. A New Perspective of Subspace Clustering

As we introduced in Section I, we consider the subspace analysis from a novel perspective: synchronization. The key point is to view each object as an oscillator which moves according to an interaction model (cf. Section III-B). For each object, we define suitable interaction partners in the local neighborhood. The strength of corresponding interactions are provided by the local structure of the neighborhood. To detect arbitrarily oriented subspace clusters we integrate PCA into the interaction model. The interactions are weighted by the main directions of the PCA result. Thus, all objects in an arbitrarily oriented subspace cluster can easily synchronize together. In the following, we elaborate this weighted interaction model.

B. Weighted Interaction Model

Currently, the most successful way to explore the synchronization phenomena is the Kuramoto Model [12], [13], which is motivated by the behavior of systems of chemical and biological oscillators. It is a mathematical model used to describe the dynamics of a large set of phase oscillators by coupling the sine of their phase differences. All frequencies of oscillators should be identical or nearly identical. Formally, the *Kuramoto model* (KM) consists of a population of N coupled phase oscillators, where the phase of the i-*th* unit, denoted by θ_i, evolves in time according to the following dynamics:

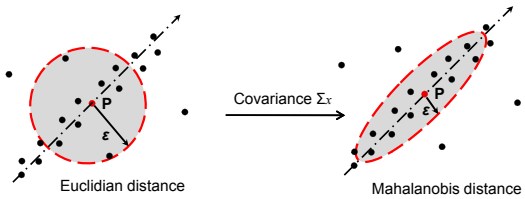


Figure 2. ϵ-Range search with different distance functions.

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i), (i = 1, \ldots, N) \quad (1)$$

where ω_i stands for natural frequency of the i-*th* unit and the constant K describes the coupling strength between units. $sin(\cdot)$ is the coupling function. This model well describes the collective behavior of all coupled phase oscillators towards synchronization, which implies that all the oscillators interact with each other and will synchronize together finally. However, this situation rarely occurs in real-life systems. Local synchronization is observed more frequently, which means a local ensemble of oscillators synchronize together, where the whole set of oscillators is split into several clusters of mutually synchronized oscillators.

Therefore, in order to introduce the Kuramoto model into subspace clustering, we reconsider it in a different way.

1) **Local Interaction**. To exploit the hidden clusters or patterns in arbitrarily oriented subspaces, the local structure of data should be investigated. Therefore, we focus on the dynamics of objects in a local way.
2) **Weighted Interaction**. In high dimensional space, the correlations in the dimensions are often specific to data locality, which means that some objects are correlated with respect to one given set of dimensions and others are correlated with respect to a different set of dimensions. The coupling strength of the interaction of objects should be considered with different weights, depending how relevant the dimensions locally are.

In the following, we will reformulate our interaction model based on the above two criteria.

1) Local Interaction: To formalize the local interaction for each object, the intuitive way is to consider its ε-neighborhood as follows.

Definition 1 (ε-Neighborhood) Given a $\varepsilon \in \mathcal{R}$ and $x \in \mathcal{D}$, the ε-neighborhood of an object x, denoted by $N_\varepsilon(x)$, is defined as:

$$N_\varepsilon(x) = \{y \in \mathcal{D} | dist(y, x) \leq \varepsilon\} \quad (2)$$

where $dist(y, x)$ is a metric distance function and Euclidean distance is used here.

However, this definition of the ε-neighborhood search cannot meet well our goals since it does not consider the

local data distribution. We intend to search for objects which are near together according to the local subspace cluster structure. Therefore, we use the Mahalanobis distance instead of Euclidean distance to define similarity.

DEFINITION 2 (ε-NEIGHBORHOOD WITH MAHALANOBIS DISTANCE) Given a $\varepsilon \in \mathcal{R}$ and $x \in \mathcal{D}$, the ε-neighborhood of an object x with Mahalanobis distance, $N_\epsilon^m(x)$, is defined as:

$$N_\varepsilon^m(x) = \{y \in \mathcal{D} | \sqrt{(y-x) \cdot \Sigma_x^{-1} \cdot (y-x)^T} \leq \epsilon\} \quad (3)$$

where Σ_x is the covariance matrix of ε-neighborhood of x. Since the Mahalanobis distance considers the local data distribution, it can better search similar objects considering the local cluster structure and is also less sensitive to noise. Fig. 2 illustrates the intuition of the similarity according to Euclidean and Mahalonobis distance, respectively.

According to Definition 2, we extend the Kuramoto model in a local fashion, where each object interacts with its ε-Neighborhood with mahalanobis distance during time revolution. Moreover, since we have no external knowledge of each object, all objects are assumed to be the same frequency ω, which well fits the condition of Kuramoto model. Here, we view each dimension of an object as a phase oscillator and its original value represents the initial phase.

Formally, let $x \in R^d$ be an object in the data set $\mathcal{D}$ and x_i be the i-th dimension of the data object x. $N_\varepsilon^m(x)$ is the ε-neighborhood of object x. According to Eq.(1), the dynamics of each dimension x_i of the object x with a local interaction is further written as:

$$\frac{dx_i}{dt} = \omega + \frac{K}{|N_\varepsilon^m(x)|} \sum_{y \in N_\varepsilon^m(x)} \sin(y_i - x_i) \quad (4)$$

Let $dt = \Delta t$, then:

$$x_i(t+1) = x_i(t) + \Delta t \cdot \omega + \quad (5)$$
$$+ \frac{\Delta t \cdot K}{|N_\varepsilon^m(x(t))|} \cdot \sum_{y(t) \in N_\varepsilon^m(x(t))} \sin(y_i(t) - x_i(t))$$

Since the term $\Delta t \cdot \omega$ is the same for each dimension of all objects and thus can be ignored. Let $C = \Delta t \cdot K$, the dynamics of each dimension x_i of an object x over time is written as:

$$x_i(t+1) = x_i(t) + \frac{C}{|N_\varepsilon^m(x(t))|} \cdot \sum_{y(t) \in N_\varepsilon^m(x(t))} \sin(y_i(t) - x_i(t)) \quad (6)$$

where $t = (0, \cdots, T)$ is the time step.

2) Weighted Interaction Determination: For most existing interaction models, e.g. [2], [7], [9], [12], the coupling strength of the object interactions is constant. However, this is not appropriate for subspace clustering since clusters exist in different subspaces. Thus, to ensure all cluster objects can synchronize in corresponding subspaces, the strength of interactions is guided by the local data structure of the objects. For an object x, we expect that the interactions along the main directions of the local cluster structure (relevant and potentially correlated dimensions) are imposed much higher weights while those in irrelevant dimensions have lower weights. Therefore, to determine the main directions of the local cluster structure, PCA is used to decompose the covariance matrix Σ of objects $N_\varepsilon^m(x)$, which is denoted by $\Sigma = VEV^T$. The orthogonal Matrix V is called eigenvector matrix and the diagonal matrix E is called eigenvalue matrix. The eigenvectors represent the principal directions of these similar objects and the eigenvalues represent the variance along these directions. We normalize the eigenvalues into the interval (0, 1) by dividing each eigenvalue λ_i with the sum of all eigenvalues. Then, the normalized eigenvalues are viewed as the interaction weights along with the corresponding principal directions. Finally, we project the difference vector between two objects onto these orthogonal eigenvectors and couple the difference with corresponding normalized eigenvalues. Formally, the weighted interaction between two objects is defined as follows:

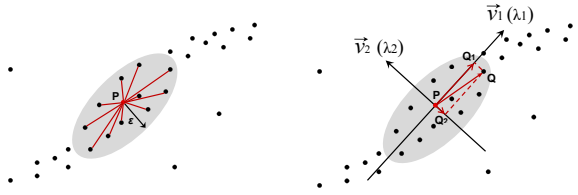


Figure 3. Weighted interaction between two objects.

DEFINITION 3 (WEIGHTED INTERACTION) Let $x \in R^d$ be an object in the data set $\mathcal{D}$. $N_\varepsilon^m(x(t))$ is the ε-Neighborhood of the object x and $y \in N_\varepsilon^m(x(t))$. $\vec{v_1}, ..., \vec{v_d}$ and $\lambda_1, ..., \lambda_d$ are the eigenvectors and eigenvalues by PCA decomposition of the covariance matrix of $N_\varepsilon^m(x(t))$. The weighted interaction between the object $y \in N_\varepsilon^m(x(t)$ and the object x, denoted by $WI_{(y-x)}$, is defined as:

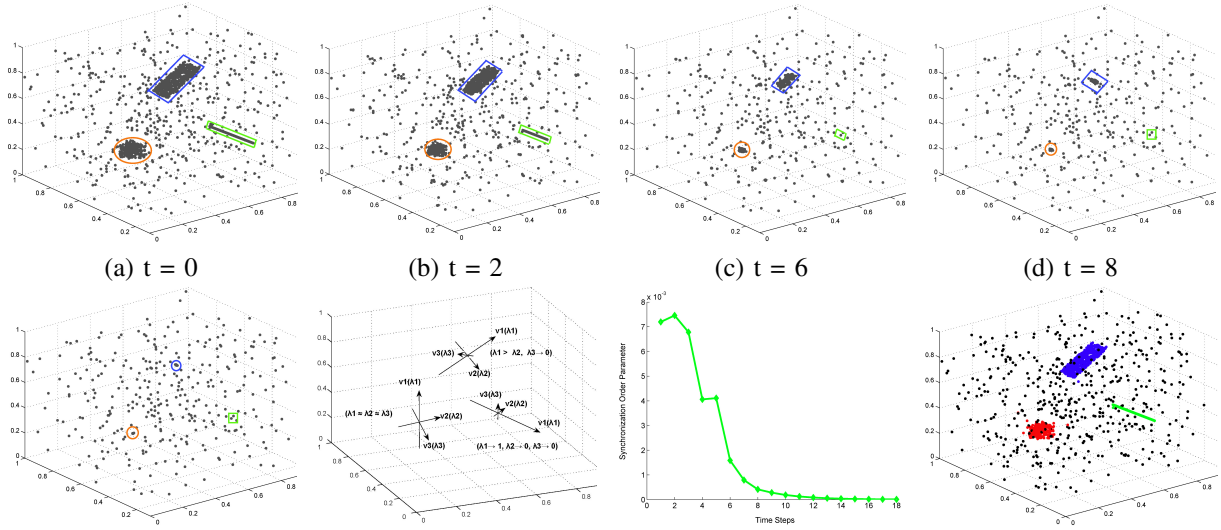
$$WI_{(y-x)} = \sum_{k=1}^{d} \lambda_i \cdot sin(proj(\Delta(y,x), \vec{v_k})) \quad (7)$$

where $\Delta(y,x) = y - x$ means the difference vector between y and x, $proj(\Delta(y,x), \vec{v_i})$ means the projection of vector $\Delta(x,y)$ onto $\vec{v_i}$. Since the eigenvectors $\vec{v_i}$ are unit vectors, therefore,

$$proj(\Delta(y,x), \vec{v_i}) = (\Delta(y,x) \odot \vec{v_i}) \cdot \vec{v_i} \quad (8)$$

where $\odot$ means the inner product.

To illustrate the weighted interaction, Fig. 3 gives an example with a 2-dimensional data set. Given an object P, first, the ε-neighborhood of object P with Mahalanobis distance are obtained. Then we decompose the covariance matrix of these objects by PCA and obtain the eigenvectors ($\vec{v_1}$, $\vec{v_2}$) and eigenvalues (λ_1, λ_2) respectively. For each interaction with object P, e.g. $Q - P$ interaction, the



(a) t = 0 (b) t = 2 (c) t = 6 (d) t = 8

(e) t = 10 (f) Main Directions (g) Order Parameter (h) Synchronized Clusters

Figure 4. Illustration of dynamics of objects according to the interaction model, where t is time step.

difference vector $\overrightarrow{QP}$ is projected to the first direction $\vec{v_1}$ with $\overrightarrow{Q_1P}$ and the second direction $\vec{v_2}$, denoted by $\overrightarrow{Q_2P}$. The interaction between objects Q and P is finally determined with $\lambda_1 \cdot sin(\overrightarrow{Q_1P}) + \lambda_2 \cdot sin(\overrightarrow{Q_2P})$.

Finally, the dynamics of each dimension x_i of the object x is governed by:

$$x_i(t+1) = x_i(t) + \frac{1}{|N_\varepsilon^m(x(t))|} \cdot \cdot \sum_{y(t)\in N_\varepsilon^m(x(t))} \sum_{k=1}^{d} \lambda_k \cdot sin(proj_{(i)}(\Delta(x(t), y(t)), \vec{v_k})) \quad (9)$$

where $proj_{(i)}$ means the i-th dimension of the projected vector. The object x at time step $t = 0 : x(0)(x_1(0); \cdots; x_d(0))$ represents the initial state of the object. The term $x_i(t+1)$ describes the renewed state value of i-th dimension of object x at time point $(t+1)$.

To determine the termination of the dynamic process, a synchronization order parameter r is defined as measuring the degree of synchronization of objects.

DEFINITION 4 (SYNCHRONIZATION ORDER PARAMETER) The synchronization order parameter r characterizing the degree of synchronization is defined as the average movements of objects over time:

$$r = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{|N_\varepsilon(x)|} \sum_{y\in N_\varepsilon(x)} WI_{(y-x)} \right) \quad (10)$$

The value of r decreases when more and more objects synchronize together as time evolves. The process of synchronization terminates when r converges, which indicates there is no further change of objects. Due to space limitation, the proof of the convergence of r is not provided here.

C. Simulation of the Object Dynamics

After formulating our interaction model, we can simulate the dynamics of objects to investigate all clusters in arbitrarily oriented subspaces. Generally, the dynamics of objects involve the following steps:

1) After initialization ($t = 0$), all objects in the data set have their own states (feature vectors).
2) As time evolves ($t > 0$), for each object $x(t)$, the similar objects according to Mahalanobis distance $N_\varepsilon^m(x(t))$ are searched. Then PCA is applied to decompose the covariance matrix of $N_\varepsilon^m(x(t))$ to obtain the corresponding eigenvectors and eigenvalues. The renewed state of object $x(t)$ is then determined by Eq. 9.
3) During the process towards synchronization, the order parameter $r(t)$ (Eq. 10) is calculated to test for convergence.

To illustrate the dynamics of objects according to our interaction model, for the ease of illustration, a 3-dimensional data set is provided as an example. Fig. 4(a) (t=0) shows the initial states of objects, where three clusters exist in the data set: a 3-dimensional Gaussian cluster in full dimensions, a 2-dimensional linear correlation cluster in arbitrarily oriented subspace and 1-dimensional linear cluster in Z axis. When $t > 0$, each object starts to interact with similar objects according to the local cluster structure. For relevant dimensions, the object interaction is imposed much higher strength while lower impact for irrelevant dimensions. E.g., in Fig. 4(b)-(e), the objects in the Gaussian cluster gradually move together from all directions in the three dimensions since these objects belong to a common 3-d subspace cluster. We can see that the eigenvalues along with main directions of these objects (eigenvectors) decomposed by PCA are

very similar ($\lambda_1 \approx \lambda_2 \approx \lambda_3$, see Fig. 4(f)). Therefore, the object interactions are imposed similar strengths and gradually group together. This situation is different from objects in the other two clusters. For the objects in the 2-dimensional correlation cluster, the main directions of the cluster structure are not along with the coordinate axis but arbitrarily oriented. The eigenvalues on the corresponding eigenvectors decomposed by PCA are thus very different, where ($\lambda_1 > \lambda_2$ and $\lambda_3 \approx 0$). These cluster objects therefore mainly move towards two directions ($\vec{v_1}$) and ($\vec{v_2}$) with weights λ_1 and λ_2 respectively (Fig. 4(f)). Similarly, the 1-dimensional linear cluster objects tend to move towards one direction ($\vec{v_1}$). Through such weighted interactions, finally, all cluster objects synchronize together in the corresponding subspaces (Fig. 4(e) and Fig. 4(h)). During the process towards synchronization, the synchronization order parameter will decrease and finally converge. (Fig. 4(g)).

D. Synchronized Clusters Search

After the simulation of dynamics of objects by our interaction model, cluster objects in arbitrarily oriented subspaces synchronize together. To find these synchronized clusters, the intuitive way is to find all synchronized phases and corresponding objects. The principle of our strategy is to consider the subspace search from objects instead of dimensionality. Specifically, after the simulation of dynamics of objects, for each object we investigate whether other objects synchronize with it in any dimension that has the same phase. If it does not synchronize with any dimension of other objects (with same phase), this object is viewed as noise. If it synchronizes with some dimensions of other objects, these synchronized dimensions are regarded as a synchronized subspace and these synchronized objects are formed as a synchronized cluster. We repeat this process for each object and finally we get all synchronized subspaces and corresponding synchronized clusters.

To illustrate the search strategy, let us look at Table I. Supposing there are 7 objects with 4 dimensions, after weighted interaction among objects, the final states of objects are outlined in the left part of Table I. For each object, we find its synchronized dimensions and corresponding objects. E.g., object #1 synchronizes with objects #2 to #4 in dimensions 1 and 2. Therefore, a synchronized subspace (1,2) is created and the corresponding objects #1 to #4 are added to a cluster C1 (1,2,3,4) in this subspace. At the same time, object #1 also synchronizes with object #5 in dimension 4. A new synchronized subspace (4) is created and corresponding new cluster C2 (1,5). Similarly object #2 synchronizes with objects #3 and #4 in dimensions (1,2,4) and a new subspace is thus created and obtains corresponding cluster C3. This process is repeated until all objects are investigated. In addition, since object #7 does not synchronize with any other object at any dimension, it is thus viewed as noise in the full dimensional space.

Table I
ILLUSTRATION OF THE SUBSPACE SEARCH STRATEGY.

Obj.	$d1$	$d2$	$d3$	$d4$	Syn. Dim.	New Subs.	Cluster
1	0.1	0.2	0.1	0.3	1,2	(1,2)	C1 (1 2 3 4)
					4	(4)	C2 (1 5)
2	0.1	0.2	0.2	0.2	1,2,4	(1,2,4)	C3 (2 3 4)
3	0.1	0.2	0.7	0.2	1,2,3,4	(1,2,3,4)	C4 (3 4)
4	0.1	0.2	0.7	0.2	1,2,3,4	-	-
5	0.3	0.4	0.3	0.3	3	(3)	C5 (5 6)
6	0.9	0.5	0.3	0.1	3	-	-
7	0.7	0.6	0.4	0.5	Null	-	Noise

Once we have detected all synchronized clusters, we can also determine their dimensionality:

DEFINITION 5 (DIMENSIONALITY OF A SYNCHRONIZED CLUSTER) Let $\mathcal{S} \subseteq \mathcal{D}$ be a synchronized cluster and $\Sigma = VEV^T, E = \text{diag}(\lambda_1, \cdots \lambda_d)$ be the eigenvalues of $\mathcal{S}$ in descending order. Given a $\delta \approx 0$, the dimensionality of $\mathcal{S}$ $w.r.t$ δ is η if $d - \eta$ eigenvalues of $\mathcal{S}$ are close to zero ($\Phi(\lambda_i) \leq \delta$ and $\Phi(\lambda_i) = \lambda_i/\lambda_1$), which is denoted by $\text{DIMSYNCLU}_\delta^\eta$.

Finally, the Pseudocode of the ORSC Algorithm is illustrated in Fig. 5.

E. Parameter Setting

To simulate the dynamics of objects, an interaction range (ε) needs to be specified in our Interaction Model. The question is: how to determine the ε value and how does the clustering results change when the ε value is adjusted? In order to generate a stable interaction among objects, a heuristic way is to use the average value of the k-nearest neighbor distance determined by a sample from the data set for a small k. Fig. 6(a) shows a simple data set which consists of 2 clusters plus outliers. We start with a very small value and then gradually increase this value to see the change of clustering results. With different parameter ε, we compute the number of clusters which are detected. We can see that the same clustering results (2 clusters) are obtained with a fairly long stable range (see Fig. 6(b)). In comparison to other subspace clustering algorithms, the parametrization of our algorithm is more flexible and robust. The reason is that our clustering is a dynamic process, which moves each object during the process towards synchronization. With a small interaction range, in the beginning, a few objects interact with each other. But after several time steps, the objects in a cluster will gradually move closer and thus more and more objects can interact with each other and finally synchronize together. If a larger interaction range is selected, the only difference is that more objects interact with each other at the beginning and thus tend to synchronize earlier.

F. Complexity Analysis

For the simulation of the dynamics of objects at each time step, we need to search ε-Neighborhood of each object with

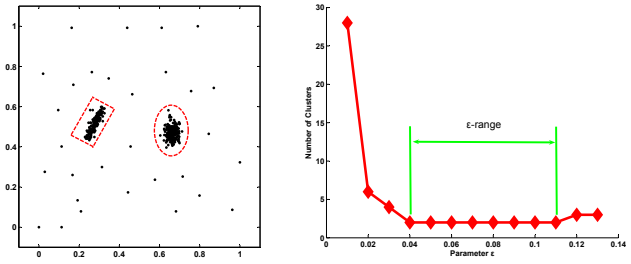
algorithm $[S, C] = ORSC(D, \varepsilon)$

D' = Simulation(D, ε);
$[S, C]$ = SearchCluster(D')
Return S, C;

//Objects' dynamics Simulation.
Function D'=Simulation(D, ε);
while(loopFlag=true)
for(each object $x \in D$)
Search $N_\varepsilon(x)$ of object x;
Compute $N_\varepsilon^m(x)$ by Definition 2 in $N_\varepsilon(x)$;
Determine the interaction directions and weights by PCA;
Obtain new state of object x with interaction model $(Eq.9)$;
end for
Set D' to be the new states of all objects;
Compute synchronization order parameter r;
if(r converges)
loopFlag=false;
end if
end while
return D';

//Search Synchronized Clusters.
Function $[S, C]$=SearchCluster(D');
$S = null; C = null$; //S saves subspaces and C saves clusters
for(each object $x \in D'$)
for(each object $y \in D'$)
If y synchronized with x in dimensions L (e.g. $dist(y_i, x_i) < 1e-4$);
If (L exists in S) and synchronized phases are same
Add y to the exist corresponding cluster;
Else
Create a new synchronized subspace L;
Create a new cluster CL;
Add x and y to the new cluster CL;
$S.add(L), C.add(CL)$;
End If
Else
y is viewed as noise object;
End For
End For
return S, C;

Figure 5. Pseudocode of the ORSC Algorithm.



(a) (b)

Figure 6. Impact of parameter setting for clustering.

Mahalanobis distance. The covariance matrix with Euclidean distance query computation can be evaluated in $O(N \cdot d)$ time. The covariance matrix is then used to calculate the Mahalanobis distance and the time complexity approximately requires $O(d^3)$ time. We further use PCA to decompose the covariance matrix with Mahalanobis distance and it thus requires $O(d^3)$ time. For each object's interaction, $O(d^3)$ time is required. Therefore, for all objects together, the simulation of dynamics at one time step is $O(N^2 \cdot d + N \cdot d^3)$. If there exists an efficient index, the complexity reduces to $O(T \cdot N \log N \cdot d + \log N \cdot d)$. For all time steps towards synchronization, the time complexity of objects' simulation is $O(T \cdot (N^2 \cdot d + N \cdot d^3))$ in worst case. T is the number of time steps. In most cases, T is small with $5 \leq T \leq 20$.

For the time complexity of subspaces and the corresponding clusters search, the most bottom-up establishing algorithms of subspace clustering need $O(2^d)$ time. For our search, we do not need to enumerate all subspaces. Instead, we find all synchronized subspaces by investigating the synchronized dimensions of each object. Therefore, the time complexity becomes $O(N^2 \cdot d)$. Finally, the time complexity of our algorithm in the worst case is $O(T \cdot (N^2 \cdot d + N \cdot d^3) + N^2 \cdot d)$.

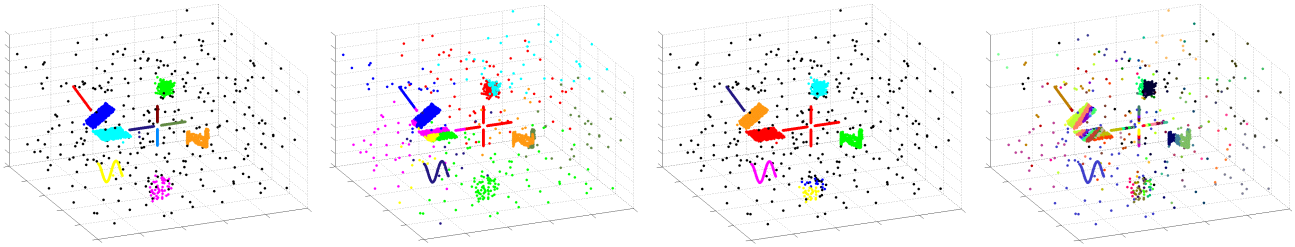
IV. Experimental Evaluation

To extensively study the performance of ORSC, we perform experiments on several synthetic and real world data sets. We compare the performance of ORSC to ORCLUS [4], 4C [8] and Curler [19]. We select these particular algorithms because they are representatives of different algorithmic paradigms: ORCLUS is a K-means style iterative partitioning algorithm; 4C is a local density-based method; Curler tries to find non-linear correlation clusters based on EM clustering. All the data are normalized and all experiments have been performed on a workstation with 2.4 GHz CPU and 2.0 GB RAM.

Moreover, we report two established measures for cluster quality [15]: Normalized Mutual Information (NMI), Adjusted Mutual Information (AMI) to evaluate different clustering results. For both measures, higher values represent better clustering performance. We also provide precision (P) and recall (R) as validity measures to analyze each individual cluster.

A. *Effectiveness*

1) Synthetic Data: We start the evaluation of ORSC with several synthetic data sets to facilitate presentation and demonstrate its benefits. Fig. 7 displays the clustering results on a 3-dimensional synthetic data with all comparing subspace clustering algorithms. The data consists of 11 clusters of different dimensionality, object density and correlation strength plus noise (Fig. 7(a)). ORSC with parameter $\varepsilon = 0.06$ successfully detects all these correlation clusters, including five 1-dimensional linear correlation clusters, two 2-dimensional linear plane clusters, two 2-dimensional non-linear clusters and two 3-dimensional clusters (Fig. 7(a)). ORCLUS requires the number of cluster K and the average subspace dimensionality l as input parameter. We specify $K = 11$ and obtain the best results with $l = 3$, which are indicated in Fig. 7(b). Fig. 7(c) displays the best clustering results of 4C with parameters $\epsilon = 0.03$, $MinPts = 6$ and $\lambda = 3$. For Curler, we use all default parameter values which are suggested by authors. It obtains as many as



(a) ORSC (b) ORCLUS (c) 4C (d) Curler

Figure 7. Comparison with different algorithms on a 3-d synthetic data.

Table II
COMPARATIVE EVALUATION OF DIFFERENT SUBSPACE APPROACHES ON HIGH-DIMENSIONAL SYNTHETIC DATA SETS.

Data	d	#Clu.	Dim. of Clu.	True clusters found by			
				ORCLUS	4C	Curler	ORSC
DS1	5	1	3	1 (Dim.: 3) (P=32.7%;R=91.2%)	1 (Dim.: 3) (P=100%;R=100%)	1 (Dim.: 3) (P=99.0%;R=20.4%)	1 (Dim.: 3) (P=100%;R=97.0%)
DS2	10	1	5	1 (Dim.: 5) (P=27.8%;R=62.2%)	1 (Dim.: 5) (P=100%;R=94.2%)	1 (Dim.: 5) (P=48.4%;R=11.8%)	Dim.: 5) (P=100%;R=97.4%)
DS3	15	2	10,5	2 (Dim.: 10,5) (P=16.9%,74.4%) (R=14.0%,61.6%)	1 (Dim.: 10) (P=100%,R=99.6%)	2 (Dim.: 10,5) (P=14.5%,12.0%) (R=19.3%,16.0%)	2 (Dim.: 10,5) (P=100%,99.6%) (R=99.6%,100%)
DS4	20	2	10,10	2 (Dim.: 10,10) (P=17.9%,100%) (R=13.2%,74.0%)	2 (Dim.: 10,10) (P=100%,100%) (R=100%,100%)	2 (Dim.: 10,10) (P=12.5%,100%) (R=12.5%,100%)	2 (Dim.: 10,10) (P=100%,99.6%) (R=100%,99.6%)
DS5	30	3	20,15,10	3 (Dim.: 20,15,10) (P=21.7%,12.3%,13.2%) (R=100%,67.5%,72.5%)	1 (Dim.: 20) (P=100%; R=98.5%)	3 (Dim.: 20,15,20) (P=100%,99.5%,76.9%) (R=99.0%,100%,5%)	3 (Dim.: 20,15,10) (P=100%,98.5%,99.6%) (R=100%,99.5%,100%)

150 clusters (Fig. 7(d)). For better investigating different clustering results, we focus on the detailed view of the 2-d linear plane cluster and four 1-d linear clusters in the center of the data set (cf. Fig. 8). It is obvious that ORSC outperforms the competitors, where all correlation clusters are successful detected with high precision and recall. The evaluation of the clustering results is further illustrated in Table III.

To further evaluate our algorithm ORSC, we generate five high-dimensional data sets. In each data set, several clusters are hidden in subspaces of varying dimensionality plus noise. All dimensions for noise objects were drawn independently at random from the uniform distribution. The values of dimensions for cluster objects were generated from a d-dimensional (the number of dimensionality for clusters) multivariate Gaussian distribution with different means and covariance matrices. We successively increased the dimensionality and also added noise dimensions. For comparison, we check whether each algorithm can detect these clusters with suitable parameters. We report its precision and recall for each cluster. The results with different subspace clustering algorithms are depicted in Table II. In all these data sets, ORSC finds the synthetic clusters in corresponding subspaces with both high recall and precision. In contrast to the comparing algorithms, there is no need

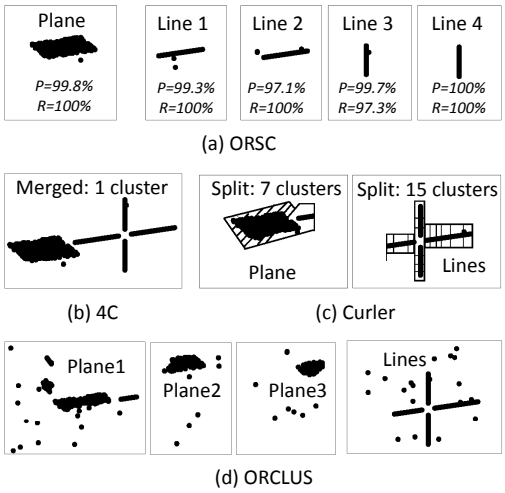


Figure 8. Detailed view of clustering results with different algorithms on part of 3-d synthetic data.

to specify the subspace dimensionality and all interesting subspace clusters are detected. For 4C and ORCLUS, we manually specify the subspace dimensionality although it is difficult to know in real-world. 4C performs very well on the first two low-dimensional data sets. But it fails to find 5-dimensional clusters in the 15-dimensional data set. This

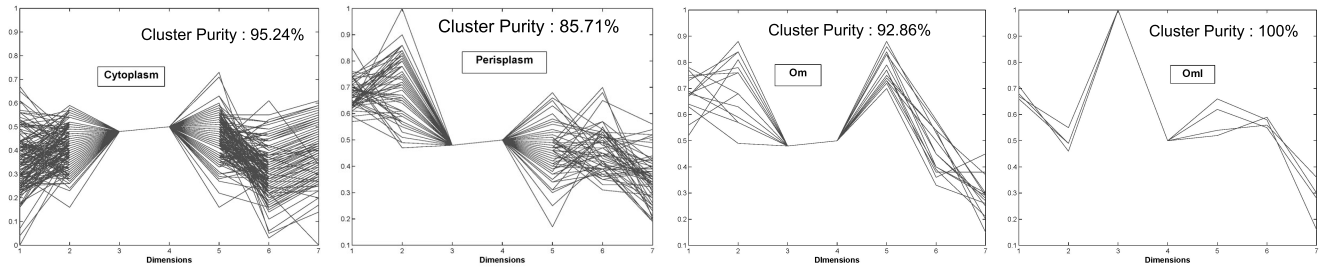


Figure 9. Clusters found by ORSC on the Ecoli data set ($\varepsilon = 0.25$)

Table III
EVALUATION ON DIFFERENT DATA SETS.

Methods	Synthetic data		Ecoli data		Wine data	
	NMI	AMI	NMI	AMI	NMI	AMI
ORSC	**0.981**	**0.980**	**0.682**	**0.670**	**0.701**	**0.695**
4C	0.830	0.829	0.338	0.328	0.474	0.469
ORCLUS	0.598	0.596	0.452	0.430	0.191	0.182
Curler	0.583	0.561	0.060	0.049	0	0

Table IV
ORSC CLUSTERING RESULTS ON WINE DATA.

Cluster ID	Type1	Type2	Type3	Prec.	Rec.
1	58	3	0	95.2%	98.3%
2	0	53	0	100%	73.6%
3	0	5	48	90.6%	100%
4	0	4	0	100%	5.6%

is also the same for the fifth data set, where it cannot find the 10-dimensional cluster and 15-dimensional cluster because these two clusters are not dense in the full 30-dimensional feature space any more. The algorithm ORCLUS is sensitive to noise and usually the noise objects are included in some clusters. In addition, ORCLUS often merges these subspace clusters together, which thus results in low precision. The algorithm Curler is also sensitive to noise and cannot yield good results for all data sets. However, in contrast to ORCLUS, it tends to split true clusters into several distinct clusters. Therefore, the obtained clusters are usually of low recall.

2) Real World Data: In the following, we evaluate the performance of ORSC on two real-world data sets which are publicly available at the UCI machine learning repository.

Ecoli Data: This Ecoli data deriving from a study on protein location consists of 336 instances. It includes eight highly unbalanced classes having from 2 to 142 objects per class. Each instance is described by 7 attributes. ORSC detects 6 meaningful clusters for this data set. Each cluster is mainly represented as one class, see Fig. 9 in detail. Cluster 1 is composed of 147 instances and 140 out of them belong to cytoplasm, which results in $P = 95.24\%$ and $R = 97.90\%$. Cluster 2 includes 56 instances and mainly represents perisplasm with $P = 85.71\%$ and $R = 92.31\%$. The type of inner membrane without signal sequence is identified by cluster 3 and cluster 4 together. Similarly, the cluster 5 and cluster 6 represent the type outer membrane and outer membrane lipoprotein with high precision 92.86% and 100% respectively. The 4C algorithm obtains best results with parameters $Minpts = 6$, $\epsilon = 0.15$ and $\lambda = 7$. It detects 6 clusters but with low class purity. ORCLUS obtains its best results with parameters $k = 8$ and $l = 7$. However, like 4C, many instances are wrongly assigned. The algorithm Curler has similar results like 4C and ORCLUS, which cannot predict the protein location effectively. The evaluation of the clustering results is further illustrated in Table III.

Wine Data: The well-known wine data set is the result of a chemical analysis of wine grown in the same region in Italy but deriving from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wine. The class distribution is as follows: type1: 59; type2: 71, type3: 48. Table IV gives the clustering results of ORSC with parameter $\varepsilon = 0.6$. It detects 4 clusters and 8 instances are viewed as noise. Three main detected clusters match with corresponding wine types with high precision and recall. In detail, cluster 1 includes nearly all instances (58 out of 59 instances) of type 1 plus 3 instances of type 2. The 53 instances in cluster 2 completely belong to type2. Cluster 3 consists of all instances of type 3. In addition, the cluster 4 includes 4 instances of type 2. It is clear that ORSC can discover interesting patterns of the data set effectively. For 4C with parameter $Minpts = 6$, $\epsilon = 0.5$ and $\lambda = 13$, it discovers 2 clusters and 34 instances are regarded as noise. As many instances of the two clusters are wrongly clustered and it is difficult to distinguish the three cultivar types. The algorithm of ORCLUS cannot obtain good clustering results although we manually specify the number of clusters with parameters $k = 3$ and $l = 13$. Curler even cannot find any correlation cluster for the data set with different parameters. All instances are assigned to the same cluster. The evaluation of these clustering results is further indicated in Table III.

B. Efficiency

Fig. 10(a) shows the results of runtime experiments for a 3-d synthetic data set with varying number of objects in the range of 500 to 30,000. ORCLUS and ORSC scale nearly linear against the size of the data set while 4C scales

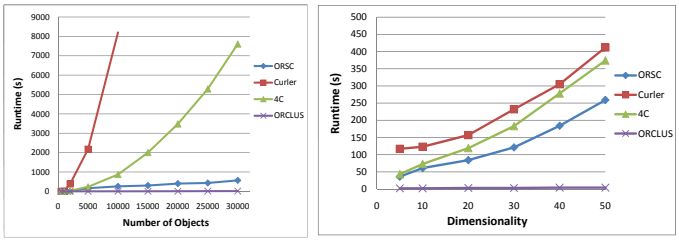


Figure 10. Scalability of ORSC against the size and dimensionality of the data set.

quadratically. The runtime of Curler is rather high for large number of objects and thus only the runtime for maximal 10,000 objects is displayed. For comparison of the scalability in the dimensionality d, data sets consisting of 2000 objects of dimensionality ranging from 5 to 50 are generated. Fig. 10 (b) reports the runtime against dimensionality. Since all algorithms are involved with PCA, they scale similar with the dimensionality. ORCLUS is the fastest approach but its effectiveness is not satisfying (cf. Section IV-A). ORSC outperforms 4C and Curler. In summary, ORSC scales very well with the number of objects and dimensionality.

V. Conclusions

In this paper, we propose ORSC, a novel subspace clustering algorithm based on synchronization. We consider each dimension of object as a phase oscillator and simulate the dynamics of each object according to our proposed interaction model. Through the weighted non-linear interaction among objects, all axis-parallel and arbitrarily oriented subspace clusters naturally synchronize together. To the best of our knowledge, ORSC is the first approach relating the problem of finding subspace clusters in high-dimensional data to synchronization. Extensive experiments demonstrated the superior efficiency and effectiveness of our ORSC algorithm.

References

[1] J. A. Acebron, L. L. Bonilla, C. J. P. Vicente, F. Ritort and R. Spigler. The Kuramoto model: A simple paradigm for synchronization phenomena. *Rev. of Modern Physics*,77(2):137-185, 2005.

[2] D. Aeyels and F. D. Smet. A mathematical model for the dynamics of clustering. *Physica D: Nonlinear Phenomena*, 273(19):2517–2530, 2008.

[3] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD Conference*, pages 61–72, 1999.

[4] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD Conference*, pages 70–81, 2000.

[5] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD Conference*, pages 94–105, 1998.

[6] A. Arenas, A. D.-Guilera, J. Kurths, Y. Moreno, and C. S. Zhou. Synchronization in complex networks. *Phys. Rep.* 469:93–153, 2008.

[7] A. Arenas, A. Diaz-Guilera, and C. J. Perez-Vicente. Synchronization reveals topological scales in complex networks. *Phys. Rev. Lett.*, 96(11):1–4, 2006.

[8] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *SIGMOD Conference*, pages 455–466, 2004.

[9] C. Böhm, C. Plant, J. Shao, and Q. Yang. Clustering by synchronization. In *KDD Conference*, pages 583–592, 2010.

[10] C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD Conference*, pages 84–93, 1999.

[11] P. Kröger, H.-P. Kriegel, and K. Kailing. Density-connected subspace clustering for high-dimensional data. In *SDM Conference*, 2004.

[12] Y. Kuramoto. Self-entrainment of a population of coupled nonlinear oscillators. In *Proceedings of the International Symposium on Mathematical Problems in Theoretical Physics,Lecture Notes in Physics*, pages 420–422. edited by H. Araki (Springer, New York, USA), 1975.

[13] Y. Kuramoto. *Chemical oscillations, waves, and turbulence*. Springer-Verlag, Berlin, 1984.

[14] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional space. In *ICDM Conference*, pages 187–194, 2003.

[15] X. V. Nguyen, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *ICML Conference*, pages 1073–1080, 2009.

[16] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *SIGMOD Conference*, pages 418–427, 2002.

[17] M. B. H. Rhouma, H. Frigui. Self-organization of pulse-coupled oscillators with application to clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2): 180-195, 2001.

[18] J. Shao, C. Böhm, Q. Yang, and C. Plant. Synchronization based outlier detection. In *ECML/PKDD Conference*, pages 245–260, 2010.

[19] A. K. H. Tung, X. Xu, and B. C. Ooi. Curler: Finding and visualizing nonlinear correlated clusters. In *SIGMOD Conference*, pages 467–478, 2005.

[20] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *SIGMOD Conference*, pages 394–405, 2002.