Robust Synchronization-Based Graph Clustering

Junming Shao[1], Xiao He[2], Qinli Yang[2], Claudia Plant[3], and Christian Böhm[2]

[1] University of Mainz, 55122 Mainz, Germany
[2] University of Munich, 80538 Munich, Germany
[3] Florida State University, FL 32306-4120 Tallahassee, USA

Abstract. Complex graph data now arises in various fields like social networks, protein-protein interaction networks, ecosystems, etc. To reveal the underlying patterns in graphs, an important task is to partition them into several meaningful clusters. The question is: how can we find the natural partitions of a complex graph which truly reflect the intrinsic patterns? In this paper, we propose *RSGC*, a novel approach to graph clustering. The key philosophy of *RSGC* is to consider graph clustering as a dynamic process towards synchronization. For each vertex, it is viewed as an oscillator and interacts with other vertices according to the graph connection information. During the process towards synchronization, vertices with similar connectivity patterns tend to naturally synchronize together to form a cluster. Inherited from the powerful concept of synchronization, *RSGC* shows several desirable properties: (a) it provides a novel perspective for graph clustering based on proposed interaction model; (b) *RSGC* allows discovering natural clusters in graph without any data distribution assumption; (c) RSGC is also robust against noise vertices. We systematically evaluate *RSGC* algorithm on synthetic and real data to demonstrate its superiority.

Keywords: Graph Clustering, Synchronization, Kuramoto Model.

1 Introduction

As a data format, graphs are characterized as a set of interconnected units. These units, often called nodes or vertices, are linked to each other by edges expressing their relationships. In recent years, the study of graph clustering has attracted a huge attentions and many techniques have been developed based on different partitioning criteria, e.g. betweenness, modularity and clique. Although established approaches have already achieved some success, finding the real and intrinsic clusters in graphs is still a big challenge [7]. Moreover, previous studies of graph clustering mainly focused on unweighted graphs. A fresh and increasingly challenging care is to study weighted graphs where each link is associated with a large heterogeneity in the capacity and intensity.

In view of these challenges, in this paper we consider graph clustering from a different perspective: *synchronization*. Synchronization is a prevalent phenomenon in nature that a group of events spontaneously come into co-occurrence with a common rhythm through mutual interactions. A paradigmatic example of synchronization phenomena is the synchronous flashing of fireflies observed in some South Asian forests. Briefly, synchronization means adjustment of states of oscillators due to weak interaction so that their states can coincide. To better illustrate the concept of synchronization, let us take social networks as an example. People in a social network with similar characteristics, e.g. common interest, friends, or similar calling behaviors from phone companies tend to group together. In such network, for a certain problem, in the beginning, each person may has his/her own opinion. As time evolves, people tend to be affected by their friends and change opinion gradually. In principle, the closer relationship they are, the higher influence between each other. Through the discussion, finally people with similar characteristics tend to achieve the same opinion. The dynamic process of opinion formation in the social network can be reviewed as a common synchronization phenomenon. From this example, what makes us interested is that the process of opinion formation in the social networks (a dynamic process towards synchronization) is very similar to a dynamic clustering process. More importantly, the interactions among vertices during the process of synchronization are completely governed by the intrinsic structural of the graph.

Therefore, inspired by natural synchronization phenomena and established models, for graph clustering, a new intuitive idea is to consider it as a dynamic process towards synchronization. We consider each vertex as an oscillator and it interacts with other vertices relying on its intrinsic connection information. The graph clustering is thus transformed into investigating the dynamics of vertices during the process towards synchronization. A graph cluster is finally defined as the vertices which can finally group together after synchronization.

The remainder of this paper is organized as follows: in the following section, we briefly survey related work. Section 3 presents our algorithm in detail. Section 4 contains an extensive experimental evaluation and we finally conclude the paper in Section 5.

2 Related Work

During the past several decades, many approaches have been proposed for graph clustering. Due to space limitation, we only provide a very brief survey on graph clustering related to our work.

Spectral Clustering: These approaches refer to a class of well-known techniques which rely on the Eigenvector decomposition of a similarity matrix to partition objects into disjoint clusters. The algorithm proposed by [14] allows detecting arbitrarily shaped clusters by considering the clustering problem from a graph-theoretic perspective. A cluster is obtained by removing the weakest edges between highly connected subgraphs, which is formally defined by the normalized cut or similar objective functions. To overcome the difficulty of parametrization, Zelnik-Manor and Perona [18] proposed a new method to estimate the number of clusters by investigating the structure of the Eigenvectors.

Multi-Level Clustering. Metis is a set of serial of multi-level partitioning techniques proposed by Karypis and Kumar [11]. For graph partitioning, an initial clustering is performed on the coarsest graph, and then, a sequence of successively finer graphs is constructed level by level. At each level, an iterative refinement algorithm such as Kernighan-Lin (KL) or Fiduccia-Mattheyses (FM) is used to further improve the bisection. In all, these methods are fast and give high-quality partitions in most cases. However, like spectral clustering, a suitable number of k clusters has to be set for the algorithm. In addition, these multilevel algorithms restrict to detect clusters of nearly equal size.

Markov Clustering. The Markov Cluster algorithm (MCL) is a popular algorithm used in life sciences for fast clustering of weighted graphs. MCL [6] simulates a flow on the graph by calculating successive powers of the associated adjacency matrix. At each iteration, an inflation step is applied to enhance the contrast between regions of strong or weak flow in the graph. The process converges towards a partition of the graph, with a set of high-flow regions (the clusters) separated by boundaries with no flow. The value of the inflation parameter strongly influences the number of clusters.

MDL-Based Clustering. The key idea of these methods is to detect clusters by using a model of probability density functions (PDFs) to describe the data structure and link the clustering problem to data compression. One of fundamental techniques in this line is Cross-Association [5] approach, which finds groups in unweighted graphs by loss-less compression with Minimum Description Length (MDL). Similar to Cross-Association, the algorithm called PaCCo [13], is proposed for weighted graph, which combines the MDL principle with a bisecting k-Means strategy. The MDL principle provides a good way to qualify the clustering results and thus avoids the parameter setting.

Synchronization. Arenas et al. [1] apply the Kuramoto model for network analysis, and study the relationship between topological scales and dynamic time scales in complex networks. From bioinformatics, Kim et.al. [4] propose a strategy to find groups of genes by analyzing the cell cycle specific gene expression with a modified Kuramoto model. Recently, Shao et.al proposed an extension of the Kuraomto model for clustering and outlier detection [15], [2], [16] on vector data based on the concept of synchronization and MDL principle.

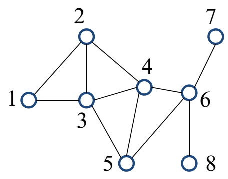
3 Synchronization-Based Graph Clustering

In this section, we introduce *RSGC* algorithm, which consider graph clustering as a dynamic process. In the following, we start with the vertex feature representation and then introduce an interaction model for graph clustering. In Section 3.3 we discuss the algorithm *RSGC* in detail.

3.1 Vertex Feature Representation

Given an undirected graph G, the only information we can gain is its connectivity patterns. In this study, we first compute the proposed *Transitive distance* of any two vertices and then transform them into a feature vector space. Each vertex

(a) Graph



(b) Jaccard Distance

Vi	Vj	*Jd*(Vi,Vj)
1	2	0.25
1	3	0.40
1	4	0.67
1	5	0.83
1	6	1
1	7	1
1	8	1

Jd(4,6) =0.57
Jd(5,6) =0.50

Transitive

Jd(6,7) =0.60
Jd(6,8) =0.60

(c) Transitive Distance

Vi	Vj	*Td*(Vi,Vj)
1	2	0.25
1	3	0.40
1	4	0.67
1	5	0.83
1	6	**1.24**
1	7	**1.84**
1	8	**1.84**

Fig. 1. Illustration of computing the dissimilarity among vertices

is finally mapped as a feature vector to represent its initial phase. Before that, let us first introduce some necessary definitions.

Definition 1 (Jaccard distance). Given any two vertices u and v in Graph G, the Jaccard distance $Jd(u, v)$ of two vertices u and v is defined as:

$$Jd(u, v) = 1 - \rho(u, v) \tag{1}$$

where $\rho(u, v)$ is the Jaccard coefficient [10] between vertices u and v,

$$\rho(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{2}$$

Here $\Gamma(\cdot)$ is neighbors of one vertex. Relying on Eq. (1), the Jaccard distance of the pairs of non-neighbor vertices is always 1, which is insufficient to represent the similarity of any two vertices effectively. Therefore, we further define *Transitive distance* as follows.

Definition 2 (Transitive distance). Given two vertices u and v have no common neighbors, $\{S_1, \cdots, S_N\}$ are all shortest paths between vertices u and v and $S_k = \{u, w_1^k, \cdots, w_{|S_k-2|}^k, v\}, k \in (1, \cdots, N)$. The Transitive distance between vertices u and v is defined as the minimal Jaccard distance of these paths.

$$Td(u,v) = \begin{cases} \min\limits_{k\in(1,\cdots,N)} \left(Jd(u, w_{|S_k-2|}^k) + Jd(w_{|S_k-2|}^k, v)\right) & Jd(u,v) = 1 \text{ AND} \\ & Jd(u, w_{|S_k-2|}^k) \neq 1 \\ Jd(u,v) & else \end{cases} \tag{3}$$

Figure 1 gives an example to illustrate the Transitive distance computation from vertex 1 to all other vertices. Based on Transitive distance, the distance matrix for all vertices can be computed. Finally, we map it into points in a feature vector space and each vertex is represented as a feature vector. Here, we apply the well-known method, called *FastMap* [8], to transform the distance matrix in a metric space into a feature vector space. In this study, we simply map each vertex into a 2-dimensional feature space. In principle, the higher dimensional feature space is selected, the less connection information among vertices is lost. However, the difficulty of clustering in high dimensional space is also increased.

3.2 Interaction Model

Currently, one of the most successful approaches to explore the synchronization phenomena is Kuramoto Model [12]. It describes the dynamics of a large set of phase oscillators by coupling the sine of their phase differences. Formally, the *Kuramoto model* (KM) consists of N phase oscillators where the phase of the *i-th* unit, denoted by θ_i, evolves in time according to the following dynamics:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i), (i = 1, \ldots, N), \tag{4}$$

where ω_i stands for its natural frequency and K describes the coupling strength between units. $sin(\cdot)$ is the coupling function.

The KM describes the global synchronization behavior of all coupled phase oscillators. However, in real-world graphs or networks, the connectivity among vertices are often not full but partial, which indicates that only a local ensemble of vertices are connected. Therefore, it is necessary to extensively reformulate $Eq.$(4). The natural and intuitive way is to model the interactions in graph with its intrinsic connection patterns.

Definition 3 (Interaction Model). Let u be a vertex in the graph G. $\Gamma(u)$ are the neighbors of vertex u and u_i is the $i-th$ feature of vertex u. The interaction range R_u of vertex u is the maximal distance to these neighbors, according to Eq.(4), the dynamics of $i-th$ phase u_i of vertex u is governed by:

$$\frac{du_i}{dt} = \omega_i + \frac{K}{\sum_{v\in\Gamma(u)} W(u,v)}\sum_{v\in\Gamma(u)} W(u,v)\cdot\Phi(u,v)\cdot\sin(v_i - u_i). \tag{5}$$

where $W(v,u)$ is the edge weight between vertices u and v. The $\Phi(u,v)$ is used to check whether the vertex v should interact with the vertex u, which is defined as:

$$\Phi(u,v) = \begin{cases} 0 \; dist(v,u) > R_u \\ 1 \; else \end{cases}$$

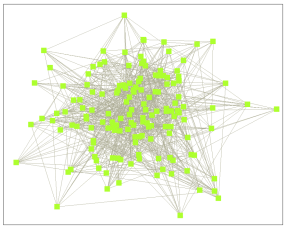
Then, let $dt = \Delta t$, the Equation (5) can be further written as:

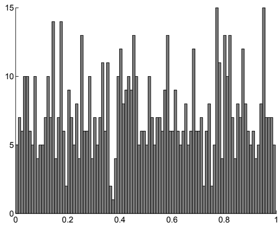
$$u_i(t{+}1) = u_i(t){+}\Delta t{\cdot}\omega_i{+}\frac{\Delta t\cdot K}{\sum_{v\in\Gamma(u)} W(v,u)}\sum_{v\in\Gamma(u)} W(v,u){\cdot}\Phi(u(t),v(t)){\cdot}\sin(v_i(t){-}u_i(t)) \tag{6}$$

Here, without knowing external knowledge, all vertices (oscillators) are assumed having the same frequency w. The term $\Delta t\cdot\omega_i$ is thus the same for each vertex and ignored. $\Delta t\cdot K = C$ is a constant and fixed as 1. Finally the dynamics of $i-th$ phase u_i of the vertex u over time is provided by:

$$u_i(t{+}1) = u_i(t){+}\frac{1}{\sum_{v\in\Gamma(u)} W(v,u)}\sum_{v\in\Gamma(u)} W(v,u){\cdot}\Phi(u(t),v(t)){\cdot}\sin(v_i(t){-}u_i(t)) \tag{7}$$

The vertex u at time step $t = 0$: $u(0)$ represents the initial phase of the vertex (original feature vector). The $u_i(t+1)$ describes the renewal phase value of i-th phase of vertex u at the $t = (0, \ldots, T)$ time evolution.


(a) Weighted Graph



(b) Edge Weights Distribution

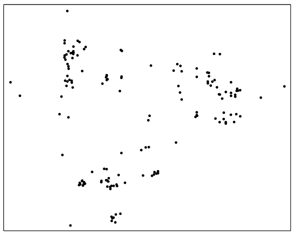
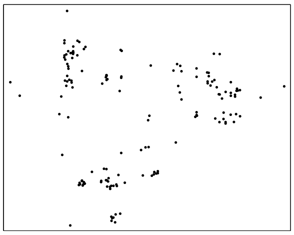
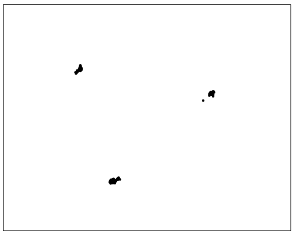
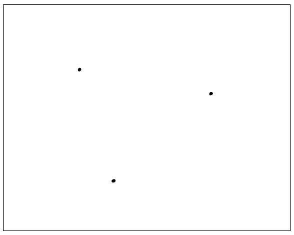
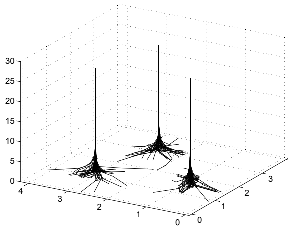

(c) Feature Representation

Fig. 2. Vertices Feature Representation

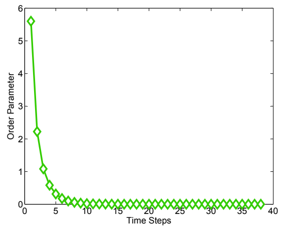

(a) t = 0


(b) t = 4


(c) t = 8



(d) Dynamics of Vertices



(e) Graph Order Parameter

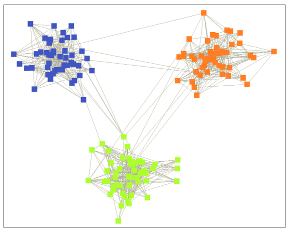

(f) Graph partitioning result

Fig. 3. The detail states of vertices during the process towards synchronization

In order to investigate local dynamic effects so that the clusters of synchronized vertices can be discovered, we define a graph order parameter r_g, measuring the coherence of the local oscillator population in graphs.

Definition 4 (Graph Order Parameter) The graph order parameter r_g characterizing the degree of local synchronization in graphs is provided by:

$$r_g = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\sum_{v\in\Gamma(u)} W(v,u)}\sum_{v\in\Gamma(u)} W(v,u)\cdot\Phi(u,v)\cdot||v-u||\right) \tag{8}$$

The more vertices are synchronized together, the value of r_g will become much smaller. The dynamics of all vertices will terminate when the r_g converges, which indicates that the vertices in clusters synchronize together (in phase).

3.3 The RSGC Algorithm

Generally, the process of the graph clustering based on synchronization involves three steps: (1) Vertices Feature Representation; (2) Dynamics on Graph and (3)

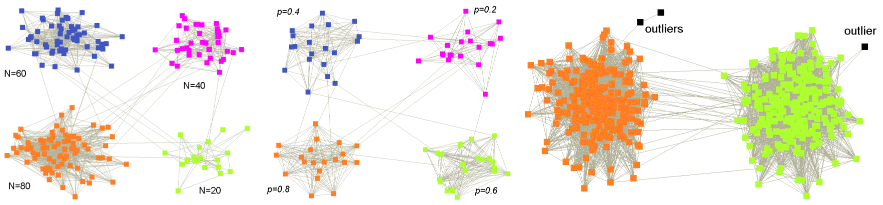
Clusters Discovering. For illustration, we introduce a simple weighted graph in Figure 2(a). Figure 2 (b) plots the edge weights distribution of the graph. With Transitive distance and FastMap, each vertex is projected into a feature space, which is indicated in Figure 2(c). After that, the dynamics of all vertices can be simulated according to $Eq.(7)$. Figure 3(f) displays the dynamic movement of all vertices during the process towards synchronization. Figure 3 (a)-(c) further depict the detailed states of vertices at time step $t = 0$ to $t = 8$. The process of synchronization will be terminated when the graph order parameter r_g converges, which indicates in Figure 3(e). The result of graph partitioning is shown in Figure 3(f). Finally, the pseudocode of *RSGC* algorithm is further described in Algorithm 1.

Algorithm 1. RSGC algorithm

```
Input: Graph $G(V, E, W)$

Compute matrix $A$ with Jaccobi Coefficient ;
Obtain dissimilarity matrix $D$ with transitive distance & $A$;
Transform $D$ into features vectors $F$ using FastMap;

for (Each $u \in F$) do
  $u(0) = F_u(0)$
end for
while (loopFlag = True) do
  for (Each $u \in F$) do
    Obtain new phase $u(t + 1)$ using $Eq.(7)$;
  end for
  Compute graph order parameter $r_g$;
  $F(t + 1) = \sum_1^N \bigcup u(t + 1)$;
  if $r_g$ converges then
    loopFlag = False;
    $C = synCluster(F(t + 1))$;
  end if
end while

return $C$;
```

4 Experiments

To extensively study the performance of *RSGC*, we conduct experiments on several synthetic and real-world data sets. We compare *RSGC* to representatives of various graph clustering paradigms: *Metis*, *MCL* and two parameter-free weighted graph clustering approaches: information-theoretic clustering *PaCCo* [13] and the spectral clustering approach [18] (named *Spectral* in the following). All experiments have been performed on a workstation with 2.0 GHz CPU and 8.0 GB RAM.



(a) Different Sizes (b) Various Densities (c) Outliers Handling

Fig. 4. Performance of *RSGC*

4.1 Evaluation Criteria

To provide an objective comparison of effectiveness, we evaluate the graph clustering algorithms in two ways. If class label information is available, three information-theoretical measures: normalized mutual information (NMI), adjusted mutual information (AMI) and adjusted variation information (AVI) [17] are directly applied for clustering comparison. For the comparison of different algorithms on the real data sets without class-label information, we evaluate them based on a measure called *modularity*[9], which is applied to quantifies the quality of a division of a network into modules or communities.

4.2 Proof of Concept

Intrinsic Cluster Structure Discovery. We first evaluate the performance of *RSGC* to discover natural graph partitioning in difficult settings, starting with subgraphs of arbitrary size. The data set displayed in Figure 4(a) consists of 4 clusters with different sizes, ranging from 20 to 80. The intra-connection is approximately 40% and the distribution of edge weights is Gaussian. *RSGC* successfully detects all clusters without any edge weights distribution assumptions. Moreover, we generate four clusters with different densities. For each cluster, it includes 20 vertices and the probabilities of intra-connection of each cluster vary from 20% to 80%, see Figure 4(b).

Outliers Handling. Inherit from the concept of synchronization, *RSGC* algorithm allows detecting outliers, where these vertices in graphs are difficult to synchronize with other vertices and have different dynamics. As displayed in Figure 4(c), the outliers are found by exploring these vertices which are out of synchronization.

4.3 Synthetic Data

For comparison, we further create a graph consisting of four clusters with different settings to evaluate their effectiveness. The number of vertices per cluster varied from 25 to 100. Meanwhile, the probabilities of intra-connection of each cluster ranging from 20% to 80% and the inter-connection among clusters is randomly interlinked with 10% . The weights of all connections are generated

Table 1. Performance of graph clustering algorithms on the simple synthetic data set

Algorithms	RSGC	Metis	MCL	PaCCo	Spectral
NMI	1	0.686	0.922	0.924	0.963
AMI	1	0.680	0.920	0.922	0.962
AVI	1	0.684	0.936	0.953	0.975

with Gaussian distribution. In addition, 6 nodes are randomly inserted into the graph with very few connections. The quality of clustering results based on different clustering approaches is illustrated in Table 1. *RSGC* successfully find all 4 clusters and irregular nodes automatically. The experiment shows that *RSGC* outperforms Metis and is also comparable to algorithms of *MCL*, *Spectral* and *PaCCo*.

4.4 Real World Data

In this section, we evaluate the performance of *RSGC* on several real-world data sets. Due to space limitation and difficult parametrization, we limit the comparison to the parameter-free graph clustering algorithms *RSGC*, *Spectral* and *PaCCo*. We obtain five author-collaboration networks from different communities: Network Theory (Netsci), PhD Student Network in Computer Science (CS-PhD), Computational Geometry (Geom), Arxiv General Relativity (CA-GrQc), Erdos Research (Erdos) [1] [2] ; three Protein-Protein Interaction networks from three species, which are *S. cerevisiae (Scere)*, *Escherichia coli (Ecoli)* and *C.elegans (Celeg)* and a Autonomous Systems network of routers comprising the Internet (As) [3].

Table 2 shows the clustering results in terms of *modularity score*. It is obvious that *RSGC* perform well on all these data sets, which obtain the best modularity scores for all experiments except the *Spectral* algorithm on *As* data set. For the algorithm of *PaCCo*, it can not yield good partition results for most data sets, especially for the unweighted graphs. The reason behind it is that *PaCCo* tends to fail if the weight distribution does not correspond to the cluster model. Like *PaCCo*, *Spectral* also obtain relatively few clusters except for the *CS-PhD* and *As* data sets.

Case Study: To further evaluate the performance of *RSGC*, we illustrate it on a case study on a protein-protein interaction (PPI) network. Here, we use the latest version of PPI network of C.elegans (Celeg), which contains 2880 proteins and 4812 known interactions. We analyze this interaction network with *RSGC* and also compare its performance to *PaCCo* and *Spectral*. *RSGC* discovers 32 clusters, while *PaCCo* and *Spectral* produce only 2 clusters, respectively. In the context of biology, we can evaluate the biological significance of obtained clusters with the help of the Gene Ontology database, which provides the ontology

[1] `http://vlado.fmf.uni-lj.si/pub/networks/data/default.htm`

[2] `http://www-personal.umich.edu/~mejn/netdata,`
`http://snap.stanford.edu/data/`

[3] `http://dip.doe-mbi.ucla.edu/dip/Main.cgi,http://snap.stanford.edu/data/`

Table 2. Performance of graph clustering algorithms on real world data sets

Data Set	PaCCo		Spectral		RSGC	
	#cluster	Modularity	#cluster	Modularity	#cluster	Modularity
NetSci	11	0.542	4	0.696	19	**0.779**
CS-Phd	2	0.077	40	0.531	36	**0.715**
Geom	54	0.327	6	0.067	44	**0.465**
CA-GrQc	2	0.352	4	0.144	46	**0.627**
As	2	0.151	19	**0.439**	39	0.382
Erdos	2	0.049	2	0.001	24	**0.422**
Scere	2	0.085	3	0.053	32	**0.196**
Ecoli	2	0.052	2	0.002	25	**0.224**
Celeg	2	0.019	2	0.005	32	**0.309**

Table 3. Biological significant clusters detected by different clustering algorithms

Molecular Function Annotations		P-value
RSGC	structural constituent of ribosome	1.2e-17
	acetylcholine receptor activity	3.1e-6
	protein binding	0.002
PaCCo	structural constituent of ribosome	2.3e-9
Spectral	structural constituent of ribosome	1.3e-19

Biological Processing Annotations		P-value
RSGC	embryo development	2.1e-24
	reproduction	2.9e-11
	growth	0.004
	multicellular organismal reproductive pro.	0.006
	protein localization	0.007
	morphogenesis of an epithelium	0.007
	germ cell development	0.009
	translation	0.011
	inductive cell migration	0.045
PaCCo	reproduction	3.7e-20
	embryo development ending in birth	6.8e-18
Spectral	reproduction	3.1e-37

of defined terms representing gene product properties on three vocabularies of annotations: Molecular Function, Biological Process and Cellular Component. Researchers can apply P-value to demonstrate the biological significance, which is defined as the probability to observe by chance at least x elements at the intersection between the query set and the reference set [3].

Under the evaluation with Molecular Function annotations, *RSGC* finds three clusters which are enriched for three molecular functions. In contrast, *PaCCo* and *Spectral* only obtain one biological significance cluster for molecular functions. In addition, for all three approaches, they find a significant cluster enriched for the function *structural constituent of ribosome*, where the P-values are 1.2e-17, 2.3e-9 and 1.3e-19 for *RSGC*, *PaCCo* and *Spectral* respectively. In addition,

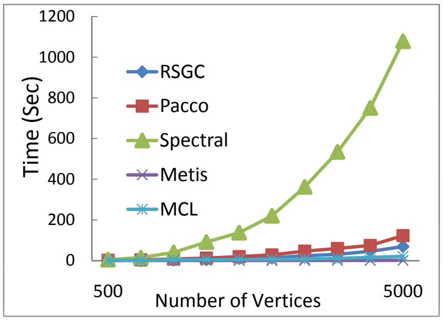


Fig. 5. The runtime of the different graph clustering algorithms

RSGC finds another two clusters enriched for *protein binding* (P-value = 2.5e-03), *acetylcholine receptor activity* (P-value = 3.1e-06). Therefore, *RSGC* can detect more clusters which make sense biologically. Similarly, we also evaluate the clusters using biological processing annotations. Here, *RSGC* successfully obtains 9 significant clusters which are enriched for biological processing while *PaCCo* and *Spectral* have two and one significant clusters respectively. All of *RSGC*, *PaCCo* and *Spectral* methods discover one significant cluster which is enriched for the term *reproduction* with the P-values of 2.8e-11, 3.7e-20, 3.1e-37 respectively. Moreover, *RSGC* also reveals another 8 significant clusters enriched for different biological process, such as *embryo development*, *multicellular organismal reproductive process* , *morphogenesis of an epithelium*, etc. Please refer to Table 3 for details.

4.5 Runtime

For runtime comparisons, we generated several synthetic data sets, where the number of clusters k varied ranges from 10 to 50 and each cluster contained 100 vertices. Approximately 30 % of the intra cluster edges were connected and 5% inter cluster edges were linked. To obtain more accurate runtime results, for each method, each data set was processed for 10 times and then found the mean of the 10 rounds. Fig. 5 clearly shows that *RSGC* is faster than *Spectral* and *PaCCo*. However, *RSGC* is slightly slower than the parameter dependent approach MCL and $Metis$.

5 Conclusions

In this paper, we introduce *RSGC*, a natural graph clustering algorithm based on synchronization. The key idea is to consider the graph clustering as a dynamic process towards synchronization. The extensive experiments demonstrate that *RSGC* algorithm has several desirable properties: *RSGC* provides a natural way for graph clustering, where the proposed interaction model well fits the real-world networks, such as the interaction weights and range. Relaying on the proposed interaction model, *RSGC* allows discovering graph clusters with arbitrary size and density without any data distribution assumption. *RSGC* is robust against noise vertices or outliers.

Acknowledgments. Junming Shao is supported by the Alexander von Humboldt-Foundation.

References

1. Arenas, A., Diaz-Guilera, A., Perez-Vicente, C.J.: Synchronization reveals topological scales in complex networks. Phys. Rev. Lett. 96(11), 1–4 (2006)
2. Böhm, C., Plant, C., Shao, J., Yang, Q.: Clustering by synchronization. In: KDD, pp. 583–592 (2010)
3. Brohee, S., Faust, K., Lima-Mendez, G., Vanderstocken, G., van Helden, J.: Network analysis tools: from biological networks to clusters and pathways. Nat. Protoc. 3, 1616–1629 (2008)
4. Bae, C.S., Kim, C.S., Tcha, H.J.: Synchronization clustering algorithm for identifying interesting groups of genes from cell cycle expression data. BMC Bioinformatics 9(56) (2008)
5. Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C.: Fully automatic cross-associations. In: KDD, New York, pp. 79–88 (2004)
6. Dongen, S.: A cluster algorithm for graphs. Technical report, CWI (Centre for Mathematics and Computer Science), The Netherlands (2000)
7. Evans, T.: Clique graphs and overlapping communities. Journal of Statistical Mechanics, P12037 (2010)
8. Faloutsos, C., Lin, K.: Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: SIGMOD (1995)
9. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99(12), 7821–7826 (2002)
10. Jaccard, P.: Distribution de la flore alpine dans la Bassin de Dranses et dans quelques regions voisines. Bulletin de la Société Vaudoise des Sciences Naturelles 37, 241–272 (1901)
11. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20, 359–392 (1998)
12. Kuramoto, Y.: Chemical oscillations, waves, and turbulence. Springer, Berlin (1984)
13. Mueller, N., Haegler, K., Shao, J., Plant, C., Böhm, C.: Weighted graph compression for parameter-free clustering with pacco. In: SDM (2011)
14. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS 14, pp. 849–856 (2001)
15. Shao, J., Plant, C., Yang, Q., Boehm, C.: Detection of Arbitrarily Oriented Synchronized Clusters in High-Dimensional Data. In: ICDM 2011, pp. 607–616 (2011)
16. Shao, J., Böhm, C., Yang, Q., Plant, C.: Synchronization based outlier detection. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part III. LNCS (LNAI), vol. 6323, pp. 245–260. Springer, Heidelberg (2010)
17. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: ICML 2009, New York, NY, USA, pp. 1073–1080 (2009)
18. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: NIPS, vol. 17, pp. 1601–1608 (2004)