Mixed-Order Spectral Clustering for Networks

Yan Ge, Haiping Lu, and Pan Peng

University of Sheffield, Sheffield, UK
{yge5,h.lu,p.peng}@sheffield.ac.uk

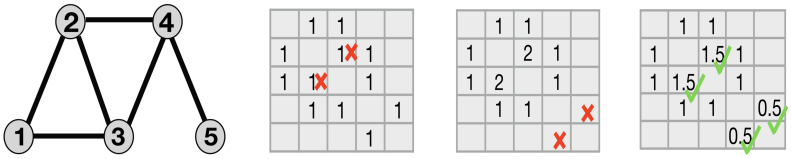
Abstract. Clustering is fundamental for gaining insights from complex networks, and spectral clustering (SC) is a popular approach. Conventional SC focuses on *second-order* structures (e.g., edges connecting two nodes) without direct consideration of *higher-order* structures (e.g., triangles and cliques). This has motivated SC extensions that directly consider higher-order structures. However, both approaches are limited to considering a single order. This novel research paper proposes a new *Mixed-Order* Spectral Clustering (MOSC) approach to model both second-order and third-order structures simultaneously, with two MOSC methods developed based on Graph Laplacian (GL) and Random Walks (RW). MOSC-GL combines edge and triangle adjacency matrices, with theoretical performance guarantee. MOSC-RW combines first-order and second-order random walks for a probabilistic interpretation. We automatically determine the mixing parameter based on cut criteria or triangle density, and construct new structure-aware error metrics for performance evaluation. Experiments on five real-world networks show that MOSC-GL with automatically determined mixing parameter improves by 5.4% over the best competing algorithm in average normalised mutual information (NMI) of four networks with low triangle density. For the remaining PBlogs network with high triangle density, MOSC-RW with automatically determined mixing parameter improves by 18.9 times over the best competing algorithm in NMI.

Keywords: Spectral clustering · Network analysis · Higher-order structures · Mixed-order structures

1 Introduction

Networks (a.k.a. graphs) are important data structures that abstract relations between discrete objects, such as social networks and brain networks [4]. A network is composed of nodes and edges representing node interactions. *Clustering* is an important and powerful tool in analysing network data, e.g., for community detection [6, 27].

Clustering aims to divide a data set into *clusters* (or *communities*) such that the nodes assigned to a particular cluster are similar or well connected in some predefined sense [9, 23, 26]. It helps us reveal functional groups hidden in data. As a popular clustering method, conventional spectral clustering (SC) [20, 24] encodes pairwise similarity into an adjacency matrix. Such encoding inherently



(a) A network. (b) Edges. (c) Triangles. (d) Mixed.

Fig. 1. Motivation: the second and third order structures in (a) can not be fully captured by edge/triangle adjacency matrix in (b)/(c). Our proposed mixed adjacency matrix in (d) can capture both.

restricts SC to *second-order structures* [4], such as undirected or directed edges connecting two nodes.[1] However, in many real-world networks, the minimal and functional structural unit of a network is not a simple edge but a small network subgraph (a.k.a. *motif*) that involves more than two nodes [21], which we call a higher-order structure.

Higher-order structures consist of at least three nodes (e.g., triangles, 4-vertex cliques) [4]. It can directly capture interaction among three or more nodes. When clustering networks, higher-order structures can be regarded as fundamental *units* and algorithms can be designed to minimise cutting them in partitioning. Clustering based on higher-order structures can help us gain new insights and significantly improve our understanding of underlying networks. For example, triangular structures, with three reciprocated edges connecting three nodes, play important roles in brain networks [30] and social networks [12, 13]. More importantly, higher-order structures allow for more flexible modelling. For instance, considering directions of edges, there exist 13 different third-order structures, but only two different second-order structures [28]. Thus, the application can drive which third-order structures to be preserved.

Thus, there are emerging interests in directly modelling higher-order structures in network clustering. These works can be grouped into five approaches: **1)** an *adjacency tensor* can be constructed to encode higher-order structures and then reduced to a matrix to apply conventional SC [10, 11], developed in the related *hypergraph clustering* problem; **2)** a transition tensor can be constructed based on random walk and then reduced to a matrix for conventional SC [3, 34]; **3)** a counting and reweighting scheme can be employed to capture higher-order structures and reveal clusters [4, 17, 32]; **4)** higher-order local clustering can be used to mitigate computation cost problem [16, 37, 39]; **5)** some network embedding methods [25, 31] preserve the higher-order proximity that considers relationship between more than two nodes, which can be applied to clustering.

However, it should be noted that most networks have both second-order and higher-order structures, and both can be important. Existing conventional and third-order SC methods model only either second-order or third-order structures, but *not both*. Second-order SC does not take triangles into consideration, while

[1] Edges are considered as first-order structures in [3] but second-order structures in [39]. We follow the terminologies in the latter [39] so that the "order" here refers to the number of nodes involved in a particular structure.

third-order SC loses information of some edges, in particular, those that do not belong to any triangle. A simple example is given by the network in Fig. 1(a), which contains both edges and triangles. In Figs. 1(b) and 1(c), which correspond to the representations used by second-order and third-order SC, respectively, each entry indicates the number of edges and triangles involving two nodes of Fig. 1(a). As the figures show: Second-order SC fails to capture the importance between nodes 2 and 3, but they participate in more triangles than any other two adjacent nodes (say 1 and 2), which is not reflected in Fig. 1(b); Third-order SC fails to model the importance of the relation between nodes 4 and 5, but there does exist an edge between them (and thus is more important than any two non-adjacent nodes, say nodes 2 and 5), which was missed in Fig. 1(c).

In this paper, we propose a new *Mixed-Order Spectral Clustering* (MOSC) to preserve structures of different orders simultaneously, as in Fig. 1(d). For clear and compact presentation, we focus on two *undirected unweighted* structures: edges (second-order structures) and triangles (third-order structures). Further extensions can be developed for mixing more than two orders, and/or orders higher than three. We summarise our three contributions as following:

1. **Mixed-order models**. We develop two MOSC models: one based on Graph Laplacian (GL) and the other based on Random Walks (RW). MOSC-GL combines edge and triangle adjacency matrices to define a mixed-order Laplacian, with its theoretical performance guarantee derived by proving a mixed-order Cheeger inequality. MOSC-RW combines first-order and second-order RW models for a probabilistic interpretation. The final clusters are obtained via a sweep cut procedure or k-means. See Sec. 3.1 and Sec. 3.2.
2. **Automatic model selection.** We propose cut-criteria-based and triangle-density-based strategies to automatically determine the mixing parameter (ranging from 0 to 1) that is the only hyperparameter in MOSC. See Sec. 3.4.
3. **Structure-aware error metrics**. We propose structure-aware error metrics to gain insights on the quality of structure preservation. Existing works on higher-order structure clustering use evaluation metrics that focus on *mis-clustered nodes* [11, 32, 37]. However, mis-clustered nodes do not have a monotonic relationship with mis-clustered structures so they may fail to reflect the errors in structures. See Sec. 3.5.

2 Preliminaries

Notations. We denote scalars by lowercase letters, e.g., a, vectors by lowercase boldface letters, e.g., $\mathbf{a}$, matrices by uppercase boldface, e.g., $\mathbf{A}$, and tensors by calligraphic letters, e.g., $\mathcal{A}$. Let $G = (V, E)$ be an undirected unweighted graph (network) with $V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_n\}$ being the set of n vertices (nodes), i.e., $n = |V|$, and E being the set of edges connecting two vertices.

2.1 Normalised Graph Laplacian

Let $\mathbf{W} \in \mathbb{R}^{n \times n}$ be an unweighted adjacency matrix of G where $\mathbf{W}(i, j) = 1$ if $(\mathrm{v}_i, \mathrm{v}_j) \in E$, otherwise $\mathbf{W}(i, j) = 0$. The degree matrix $\mathbf{D}$ is a diagonal matrix

with diagonal entries $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{W}(i,j)$, which is the *degree* of vertex v_i. Let $\mathbf{N} = \mathbf{D} - \mathbf{W}$ denote the *Laplacian matrix* of G. The *normalised Laplacian* of G is defined as $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}\mathbf{D}^{-\frac{1}{2}}$ [33].

Let $\mathbf{W}_T$ be *triangle adjacency matrix* of G with its entry $(i,\ j)$ being the number of triangles containing vertices i and j, which leads to a corresponding weighted graph G_T [4]. Similarly, we can define the *triangle Laplacian* as $\mathbf{N}_T = \mathbf{D}_T - \mathbf{W}_T$ and the *normalised triangle Laplacian* as $\mathbf{L}_T = \mathbf{D}_T^{-\frac{1}{2}}\mathbf{N}_T\mathbf{D}_T^{-\frac{1}{2}}$, where $\mathbf{D}_T(i,i) = \sum_{j=1}^{n} \mathbf{W}_T(i,j)$.

2.2 First-Order and Second-Order Random Walks

We define a second-order transition matrix $\mathbf{P}$ by normalising the adjacency matrix $\mathbf{W}$ to represent edge structures as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ [20]. The entry $\mathbf{P}_{ij}$ represents the probability of jumping from vertex v_i to v_j in one step. The transition matrix $\mathbf{P}$ represents a first-order random walk process on graph G [20].

To define second-order random walks, Benson *et al.* [3] firstly define a symmetric *adjacency tensor* $\mathcal{T} \in \mathbb{R}^{n\times n\times n}$ such that the connectivity information for three vertices $\{\mathrm{v}_i,\ \mathrm{v}_j,\ \mathrm{v}_k\} \in V$ can be represented explicitly in this tensor. Thus, $\mathcal{T}$ encodes triangle structures in G as:

$$\mathcal{T}(i,j,k) = \begin{cases} 1 & v_i, v_j, v_k \text{ form a triangle,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Next, they form a transition tensor $\mathcal{P}$ as

$$\mathcal{P}(i,j,k) = \mathcal{T}(i,j,k) / \sum_{m=1}^{n} \mathcal{T}(i,m,k), \tag{2}$$

where $\sum_{m=1}^{n} \mathcal{T}(i,m,k) \neq 0$, and $1 \leq i,\ j,\ k \leq n$. For $\sum_{m=1}^{n} \mathcal{T}(i,m,k) = 0$, Benson *et al.* [3] set $\mathcal{P}(i,j,k)$ to $\frac{1}{n}$. Here, $\mathcal{P}(i,j,k)$ represents a transition probability of a second-order random walk. And the probability of jumping to state i relies on the last two states j and k [35].

2.3 Spectral Clustering Basics and Cut Criteria

Bi-partitioning SC (Algorithm 1) first constructs a matrix $\mathbf{B}$ to encode structures in the input graph G [33]. It then computes a dominant eigenvector $\mathbf{v}$ of $\mathbf{B}$, thus making use of its *spectrum*. Each entry of $\mathbf{v}$ corresponds to a vertex. Next, we sort vertices by the values $\mathbf{v}(i)$ (or appropriately normalised values) and consider the set T_u consisting of the first u vertices in the sorted list, for each $1 \leq u \leq n-1$. Then the algorithm finds $S = \arg\min_{T_u} \tau(T_u)$, called the *sweep cut* w.r.t. some cut criterion τ [15]. The choice of the cut criterion can affect the quality of output clusters [15].

As the last step of SC, sweep cut aims to optimise a cut criterion to obtain good clustering results. Most cut criteria have two main objectives: 1) to

Algorithm 1 Bi-partitioning Spectral Clustering

1: Matrix $\mathbf{B}$ encodes structures of the input graph G.
2: Compute a dominant eigenvector $\mathbf{v}$ of $\mathbf{B}$.
3: $\mathbf{v} \leftarrow$ sorted ordering of $\mathbf{v}$ or a normalised version of sorted $\mathbf{v}$.
4: $\{S, \bar{S}\} \leftarrow$ sweep cut of $\mathbf{v}$ w.r.t. a cut criterion.

preserve a rich set of structures within S and $\bar{S}$, and 2) to avoid structures being broken due to partitioning of S and $\bar{S}$ [33]. Moreover, different cut criteria serve to preserve different structures, such as second-order cut criteria (e.g., $2nd$-conductance [27]) for the preservation of the edge structure and third-order cut criteria (e.g., $3rd$-conductance [3]) for the preservation of triangle structure. Besides sweep cut, k-means [19] is another popular choice for finding the final partitions.

2.4 Cheeger Inequalities

Given $G = (V, E)$ and a subset $S \subseteq V$, let $\bar{S}$ denote the complement of S. Let $\mathrm{cut}_2(S; G)$ denote the *edge cut* of S, i.e., the number of edges between S and $\bar{S}$ in G. Let $\mathrm{vol}_2(S; G)$ denote the *edge volume* of S, i.e., the total degrees of vertices in S. The *edge conductance* of S is defined as $\phi_2(S; G) = \frac{\mathrm{cut}_2(S;G)}{\min\{\mathrm{vol}_2(S;G), \mathrm{vol}_2(\bar{S};G)\}}$. The classical Cheeger inequality below relates the conductance of the sweep cut of SC to the minimum conductance value of the graph [7].

Lemma 1 (Second-Order Cheeger Inequality). *Let* $\mathbf{v}$ *be the second smallest eigenvector of* $\mathbf{L}$. *Let* T^* *be the sweep cut of* $\mathbf{D}^{-1/2}\mathbf{v}$ *w.r.t. cut criterion* $\phi_2(\cdot; G)$. *It holds that* $\phi_2(T^*; G) \leq 2\sqrt{\phi_2^*}$, *where* $\phi_2^* = \min_{\mathrm{S}\subseteq\mathrm{V}}\ \phi_2(\mathrm{S}; G)$ *is the minimum conductance over any set of vertices.*

Let $\mathrm{cut}_3(S; G)$ denote the *triangle cut* of S, i.e., the number of triangles that have at least one endpoint in S and at least one endpoint in $\bar{S}$. Let $\mathrm{vol}_3(S; G)$ denote the *triangle volume* of S, i.e., the number of triangle endpoints in S. The *triangle conductance* [3] of S is defined as $\phi_3(S; G) = \frac{\mathrm{cut}_3(S;G)}{\min\{\mathrm{vol}_3(S;G), \mathrm{vol}_3(\bar{S};G)\}}$. It is further proved in [4] that for any $S \subseteq V$, $\phi_3(S; G) = \phi_2(S; G_T)$, which leads to the following third-order Cheeger inequality.

Lemma 2 (Third-order Cheeger Inequality). *Let* $\mathbf{v}$ *be the second smallest eigenvector of* $\mathbf{L}_T$. *Let* T^* *denote the sweep cut of* $\mathbf{D}_T^{-1/2}\mathbf{v}$ *w.r.t. cut criteria* $\phi_2(\cdot; G_T)$. *It holds that* $\phi_3(T^*; G) \leq 4\sqrt{\phi_3^*}$, *where* $\phi_3^* = min_{\mathrm{S}\subseteq\mathrm{V}}\ \phi_3(\mathrm{S}; G)$.

3 Proposed Mixed-Order Approach

To model both edge and triangle structures simultaneously, we introduce a new Mixed-Order SC (MOSC) approach, with two methods based on Graph Laplacian (GL) and Random Walks (RW). MOSC-GL combines the edge and triangle

Algorithm 2 MOSC via Graph Laplacian (MOSC-GL)

Input: $G = (V, E)$, mixing parameter λ
Output: Two node sets S, $\bar{S}$
1: Compute the normalized mixed-order Laplacian $\mathbf{L}_X$ as defined in Eq. (3)
2: Compute the second smallest eigenvector $\mathbf{v}$ of $\mathbf{L}_X$.
3: $\mathbf{v} \leftarrow$ sorted ordering of $\mathbf{D}_X^{-\frac{1}{2}}\mathbf{v}$.
4: $\{S, \bar{S}\} \leftarrow$ sweep cut on $\mathbf{v}$ w.r.t. a cut criterion.

adjacency matrices, which leads to a mixed-order Cheeger inequality to provide a theoretical performance guarantee. MOSC-RW is developed under the random walks framework to combine the first-order and second-order random walks, providing a probabilistic interpretation. Next, we develop an automatic hyper-parameter determination scheme and define new structure-aware error metrics.

3.1 MOSC Based on Graph Laplacian (MOSC-GL)

MOSC-GL introduces a *mixed-order adjacency matrix* $\mathbf{W}_X$ that linearly combines the edge adjacency matrix $\mathbf{W}$ and the triangle adjacency matrix $\mathbf{W}_T$, with a *mixing parameter* $\lambda \in [0, 1]$. $\mathbf{W}_X$ can be seen as a weighted adjacency matrix of a weighted graph G_X, on which we can apply conventional SC (Algorithm 1). Specifically, we first construct the matrix $\mathbf{W}_X$ and the corresponding diagonal degree matrix $\mathbf{D}_X$ as:

$$\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}, \quad \mathbf{D}_X = (1-\lambda)\mathbf{D}_T + \lambda\mathbf{D}.$$

Let G_X denote an undirected weighted graph with adjacency matrix $\mathbf{W}_X$, we can define a mixed-order Laplacian $\mathbf{N}_X$ and its normalised version $\mathbf{L}_X$ as:

$$\begin{aligned} \mathbf{N}_X &= \mathbf{D}_X - \mathbf{W}_X = (1-\lambda)\mathbf{N}_T + \lambda\mathbf{N}, \\ \mathbf{L}_X &= \mathbf{D}_X^{-\frac{1}{2}}\mathbf{N}_X\mathbf{D}_X^{-\frac{1}{2}}. \end{aligned} \tag{3}$$

Then, we compute the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}_X$ and perform the sweep cut to find the partition with the smallest edge conductance in G_X. The MOSC-GL algorithm is summarised in Algorithm 2.

When $\lambda = 1$, MOSC-GL is equivalent to SC by Ng *et al.* [24] and only considers second-order structures. When $\lambda = 0$, MOSC-GL is equivalent to motif-based SC [4]. MOSC-GL maintains the advantages of traditional SC: computational efficiency, ease of implementation and mathematical guarantee on the near-optimality of resulting clusters, which we formalise and prove in the following.

Performance guarantee. Given a graph G and a vertex set S, we define its mixed-order cut and volume as

$$\begin{aligned} \mathrm{cut}_X(S; G) &= (1-\lambda)\mathrm{cut}_3(S; G) + \lambda\mathrm{cut}_2(S; G), \\ \mathrm{vol}_X(S; G) &= (1-\lambda)\mathrm{vol}_3(S; G) + \lambda\mathrm{vol}_2(S; G), \end{aligned}$$

respectively. Then, we define the *mixed-order conductance* of S as:

$$\phi_X(S;G) = \frac{\mathrm{cut}_X(S;G)}{\min(\mathrm{vol}_X(S;G), \mathrm{vol}_X(\bar{S};G))},$$

which generalises edge and triangle conductance. A partition with small $\phi_X(S;G)$ corresponds to clusters with rich edge and triangle structures within the same cluster while few both structures crossing clusters. Finding the exact set of nodes S with the smallest ϕ_X is computationally infeasible. Nevertheless, we can derive a performance guarantee for MOSC-GL to show that the output set obtained from Algorithm 2 is a good approximation.

Theorem 1 (Mixed-order Cheeger Inequality). *Given an undirected graph G, let T^* denote the set outputted by MOSC-GL (Algorithm 2) w.r.t. the cut criterion $\phi_2(\cdot;G_X)$. Let $\phi^* = \min_{S\subseteq V}\phi_X(S;G)$ be the minimum mixed-order conductance over any set of vertices. Then it holds that $\phi_X(T^*;G) \leq 2\sqrt{2\phi^*}$.*

Proof. It suffices for us to prove that for any set S,

$$\frac{1}{2}\phi_2(S;G_X) \leq \phi_X(S;G) \leq 2\phi_2(S;G_X). \tag{4}$$

Assume for now that the inequality (4) holds. By Lemma 1, the set T^* satisfies $\phi_2(T^*;G_X) \leq 2\sqrt{\psi^*}$, where $\psi^* = \min_{S\subseteq V}\phi_2(S;G_X)$. Let R be the set with $\phi_X(R;G) = \phi^* = \min_{S\subseteq V}\phi_X(S;G)$. Then by inequality (4), we have

$$\begin{aligned}\phi_X(T^*;G) &\leq 2\phi_2(T^*;G_X) \leq 2\sqrt{\psi^*} \leq 2\sqrt{\phi_2(R;G_X)} \\ &\leq 2\sqrt{2\phi_X(R;G)} = 2\sqrt{2\phi^*}.\end{aligned}$$

This will then finish the proof. Thus, we only need to prove the inequality (4).

By Lemma 4 in [4], we have $\mathrm{cut}_3(S;G) = \frac{1}{2}\mathrm{cut}_2(S;G_T)$. This gives that $\mathrm{cut}_X(S;G) = (1-\lambda)\mathrm{cut}_3(S;G)+\lambda\mathrm{cut}_2(S;G) = (1-\lambda)\frac{1}{2}\mathrm{cut}_2(S;G_T)+\lambda\mathrm{cut}_2(S;G)$.

By Lemma 1 in [4], we have $\mathrm{vol}_3(S;G) = \frac{1}{2}\mathrm{vol}_2(S;G_T)$. This gives that $\mathrm{vol}_X(S;G) = (1-\lambda)\mathrm{vol}_3(S;G)+\lambda\mathrm{vol}_2(S;G) = (1-\lambda)\frac{1}{2}\mathrm{vol}_2(S;G_T)+\lambda\mathrm{vol}_2(S;G)$.

Since the adjacency matrix of G_X is a linear combination of the adjacency matrix of G_T and the adjacency matrix of G, i.e., $\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}$, we have that $\mathrm{cut}_2(S;G_X) = (1-\lambda)\mathrm{cut}_2(S;G_T) + \lambda\mathrm{cut}_2(S;G)$, $\mathrm{vol}_2(S;G_X) = (1-\lambda)\mathrm{vol}_2(S;G_T) + \lambda\mathrm{vol}_2(S;G)$.

The above equations imply that for any set S, it holds that $\frac{1}{2}\mathrm{cut}_2(S;G_X) \leq \mathrm{cut}_X(S;G) \leq \mathrm{cut}_2(S;G_X)$, $\frac{1}{2}\mathrm{vol}_2(S;G_X) \leq \mathrm{vol}_X(S;G) \leq \mathrm{vol}_2(S;G_X)$.

The last inequality also implies that for any S, $\frac{1}{2}\mathrm{vol}_2(\bar{S};G_X) \leq \mathrm{vol}_X(\bar{S};G) \leq \mathrm{vol}_2(\bar{S};G_X)$. Therefore, by the definition of $\phi_X(S;G)$, we have

$$\begin{aligned}\phi_X(S;G) &\leq \frac{\mathrm{cut}_2(S;G_X)}{\min(\frac{1}{2}\mathrm{vol}_2(S;G_X), \frac{1}{2}\mathrm{vol}_2(\bar{S};G_X))} = 2\phi_2(S;G_X), \\ \phi_X(S;G) &\geq \frac{\frac{1}{2}\mathrm{cut}_2(S;G_X)}{\min(\mathrm{vol}_2(S;G_X), \mathrm{vol}_2(\bar{S};G_X))} = \frac{1}{2}\phi_2(S;G_X).\end{aligned}$$

This completes the proof of the inequality (4). □

Complexity analysis. The computational time of MOSC-GL is dominated by the time to form $\mathbf{W}_X$ and compute the second eigenvector of $\mathbf{L}_X$. The former requires finding all triangles in the graph, which can be as large as $O(n^3)$ for a complete graph. While most real networks are far from complete so the actual complexity is much lower than $O(n^3)$. For the latter, it suffices to use power iteration to find an approximate eigenvector, with each iteration at $\tilde{O}(g)$, where g denotes the number of non-zero entries in $\mathbf{L}_X$.

3.2 MOSC Based on Random Walks (MOSC-RW)

Alternatively, we can develop MOSC under the random walks framework. Edge- or triangle-based conductance can be viewed as a probability corresponding to the Markov chain. For a set S with edge volume at most half of the total graph edge volume, the edge conductance of S is the probability that a random walk will leave S conditioned upon being inside S, where the transition probabilities of the walk are defined by edge connections [33]. Similarly, for a set S with triangle volume at most half of the total graph triangle volume, the triangle conductance of S is the probability that a random walk will leave S conditioned upon being inside S, where the transition probabilities of the walk are defined by the triangle connections [3]. This motives us to directly combine random walks from edge and triangle connections to perform MOSC. Therefore, we propose MOSC-RW to consider both edge and triangle structures via the respective probability transition matrix and tensor, under the random walks framework.

Specifically, starting with the third-order adjacency tensor $\mathcal{T}$ (Eq. (1)), we define a third-order transition tensor $\mathcal{P}$ (Eq.(2)). In the case $\sum_{m=1}^{n} \mathcal{T}(i,m,k) = 0$, we set $\mathcal{P}(i,j,k)$ with 0. Let $\mathbf{T}_k \in \mathbb{R}^{n\times n}$ denote the kth $n \times n$ block of $\mathcal{P}$, i.e., $\mathbf{T}_k = \mathcal{P}(:,:,k)$. Next, we average $\{\mathbf{T}_k, k = 1, ..., n\}$ to reduce $\mathcal{P}$ to a similarity matrix $\mathbf{A}$:

$$\mathbf{A} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{T}_k.$$

Now recall that $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ denotes the probability transition matrix of random walks on the input graph. We construct a mixed-order similarity matrix $\mathbf{H}$ by a weighted sum of $\mathbf{A}$ and $\mathbf{P}$ via a mixing parameter $\lambda \in [0, 1]$ as:

$$\mathbf{H} = (1-\lambda)\mathbf{A} + \lambda\mathbf{P}. \tag{5}$$

Thus, we obtain the MOSC-RW algorithm with standard SC steps on $\mathbf{H}$, as summarised in Algorithm 3.

When $\lambda = 1$, MOSC-RW is equivalent to conventional SC by Shi and Meila [29] and considers only second-order structures. MOSC-RW with $\lambda = 0$ considers only third-order structures, which is a simplified (unweighted) version of tensor SC (TSC) by Benson *et al.* [3]. In the intermediate case, λ controls the trade-off.

Interpretation. Now we interpret the model (Eq. (5)) as a mixed-order random walks process. At every step, the random walker chooses either a first-order (with probability λ) or a second-order (with probability $(1-\lambda)$) random

Algorithm 3 MOSC via Random Walks (MOSC-RW)

Input: $G = (V, E)$, mixing parameter λ
Output: Two node sets S, $\bar{S}$
1: Compute the matrix $\mathbf{H}$ as defined in Eq (5).
2: Compute the second largest eigenvector $\mathbf{v}$ of $\mathbf{H}$.
3: $\mathbf{v} \leftarrow$ sorted ordering of $\mathbf{v}$
4: $\{S, \bar{S}\} \leftarrow$ sweep cut of $\mathbf{v}$ w.r.t. a cut criterion.

walk. For the first-order random walk, the walker jumps from the current node i to a neighbour j with probability $\mathbf{P}(i,j) = \frac{1}{\mathbf{D}(i,i)}$. For the second-order random walk in $\mathbf{A}$, $\mathbf{A}(i,j)$ is the probability of the following random process: supposing the walker is at vertex i, it first samples a vertex k with probability $\frac{1}{n}$, then in the case that some neighbour k of i is sampled and i, j, k forms a triangle, the walker jumps from i to j with probability $1/\mathbf{W}_T(i,k)$, where $\mathbf{W}_T(i,k)$ is the number of triangles containing both i and k.

Complexity analysis. The running time of MOSC-RW is again dominated by the time of finding all the triangles and the approximate eigenvector, and thus asymptotically the same as the running time of MOSC-GL. However, since MOSC-RW involves tensor construction, normalisation and averaging, it is more complex than MOSC-GL in implementation.

3.3 Multiple Clusters and Higher-order Cheeger Inequalities

To cluster a network into $k > 2$ clusters based on mixed-order structures, MOSC-GL and MOSC-RW follow the conventional SC [33]. Specifically, MOSC-GL treats the first k row-normalised eigenvectors of $\mathbf{L}_X$ as the embedding of nodes that can be clustered by k-means. Similarly, MOSC-RW uses the first k eigenvectors of $\mathbf{H}$ as the node embedding to perform k-means.

Regarding performance guarantee, following [4] and [14], MOSC-GL and MOSC-RW do not have performance guarantee with respect to higher-order Cheeger inequalities. However, by replacing k-means with a different clustering algorithm, MOSC-GL can derive a theoretical performance guarantee [14].

3.4 Automatic Determination of λ

The mixing parameter λ is the only hyperparameter in MOSC. To improve the usability, we design schemes to automatically determine its best value λ^* from a set Λ based on the quality of output clusters [5, 15, 36]. For bi-partitioning networks, the cut criterion used to obtain output clusters can help determine the best λ^* from Λ. For multiple partitioning networks, we can use the sum of triangle densities of individual clusters to determine the best λ^* from Λ.

Specifically, for each $\lambda' \in \Lambda$, let $\{S_{\lambda'}, \overline{S_{\lambda'}}\}$ denote the MOSC bi-partitioning clusters obtained with $\lambda = \lambda'$. For a specific minimisation or maximisation cut

criterion τ (e.g., edge conductance ϕ_2), we choose λ to be the one that optimises τ, i.e.,

$$\lambda^* = \arg \min_{\lambda' \in \Lambda} \tau(S_{\lambda'}) \text{ or } \lambda^* = \arg \max_{\lambda' \in \Lambda} \tau(S_{\lambda'}),$$

respectively.

For the case of multiple partitions, we propose a triangle-density-based scheme to determine λ as follows:

$$\lambda^* = \arg \max_{\lambda' \in \Lambda} \sum_{c=1}^{k} \frac{\sum_{v_i, v_j, v_k \in S_c(\lambda')} \mathcal{T}(i,j,k)}{6|S_c(\lambda')|},$$

where $S_c(\lambda')$ denotes the c-th cluster resulted from λ', and the factor $1/6$ is used to avoid repeated count of triangles in an undirected graph.

3.5 Structure-Aware Error Metrics

If we have ground-truth clusters available, we can use them to measure performance of clustering algorithms. Existing works commonly use mis-clustered nodes [11] or related metrics (e.g., normalised mutual information (NMI) [2]). We denote the ground-truth partition of G with k clusters as $\mathbb{S}^* = \{S_1^*, S_2^*, \ldots, S_k^*\}$ and a candidate partition to be evaluated as $\mathbb{S} = \{S_1, S_2, \ldots, S_k\}$. The mis-clustered node metric is defined as $\epsilon_N(\mathbb{S}^*, \mathbb{S}) = \frac{\min_\sigma \sum_{c=1}^{k} |S_c^* \oplus S_{\sigma(c)}|}{|V|}$, which measures the difference between two partitions $\mathbb{S}^*$ and $\mathbb{S}$, where σ indicates all possible permutations of $\{1, 2, \ldots, k\}$ and $\oplus$ denotes the symmetric difference between the two corresponding sets. A smaller ϵ_N indicates a more accurate partition.

A limitation of the above metric is that it fails to truly reflect the errors made in preserving structures such as edges or triangles. Our studies (Fig. 3 in Sec. 4.2) show that mis-clustered nodes do not have a *monotonic* relationship with mis-clustered edges or triangles. That is, a smaller number of mis-clusterd nodes does not imply a smaller number of mis-clustered edges or triangles, and vice versa. This motivates us to propose two new error metrics ϵ_E and ϵ_T that measure the *mis-clustered edges* and *triangles*, respectively. These new metrics can provide more insights in the preservation of edges and triangles.

Specifically, we define ϵ_E as

$$\epsilon_E(\mathbb{S}^*, \mathbb{S}) = \frac{\sum_{c=1}^{k} E_N(S_c^*) - \max_\sigma \sum_{c=1}^{k} E_N(S_c^* \cap S_{\sigma(c)})}{|E|}, \tag{6}$$

where $E_N(S)$ and $|E|$ are the number of edges in S and G respectively. We can define ϵ_T similarly by replacing $E_N(S)$ and $|E|$ in Eq. (6) with $T_N(S)$ and $|T|$, where $T_N(S)$ and $|T|$ are the number of triangles in S and G.

4 Experiments

This section aims to evaluate MOSC against existing SC methods. In addition, we will examine the effect of hyperparameter λ, and gain insights from the newly designed error metrics and show computational time.

4.1 Experimental Settings

Datasets. The experiments were conducted on five popular real-world networks with very different triangle densities: **1)** Zachary's karate club (Zachary) [38], **2)** Dolphin social network (Dolphin) [18], **3)** American college football (Football) [22], **4)** U.S. politics books (Polbooks) [22], and **5)** Political blogs (PBlogs) [1]. The above networks have ground-truth communities. Their statistics are summarised in Table 1.

Compared algorithms. We evaluate MOSC-GL and MOSC-RW against the following five state-of-the-art methods, including both edge-based SC and triangle-based SC: **1)** SC-Shi [29], **2)** SC-Ng [24], **3)** Tensor Spectral Clustering (TSC) [3], **4)** Higher-order SVD (HOSVD) [10], and **5)** Motif-based SC (MSC) [4] / Tensor Trace Maximisation (TTM) [11] (We have verified that TTM and MSC are equivalent).

We study two versions for each MOSC: 1) MOSC ($\lambda = 0.5$): MOSC with a fixed (recommended) λ value of 0.5; 2) MOSC (Auto-λ): MOSC with automatically determined λ. We set λ values from 0 to 1 with step 0.1.

Evaluation metrics. We use the proposed structure-aware metrics, misclustered edges (ϵ_E) and triangles (ϵ_T). We also use two popular metrics, misclustered nodes (ϵ_N) [11] and NMI [2].

Selection of cut criteria. We study seven different cut criteria covering second-order ones: **1)** $2nd$-conductance [27], **2)** $2nd$-Ncut [29], **3)** $2nd$-expansion [8], and third-order ones: **4)** $3rd$-conductance [3], **5)** $3rd$-Ncut [17], **6)** $3rd$-expansion [3], **7)** $3rd$-Nassociation [11] and also k-means [19]. By doing this, second-order methods can use third-order cut criteria (e.g., SC-Shi+$3rd$-conductance), and third-order methods can use second-order cut criteria (e.g., TSC+$2nd$-Ncut). For fair comparison, we show the performance of all algorithms with the best cut criteria for bi-partitioning networks. For multi-partitioning networks, all algorithms use k-means.

Reproducibility. We used Matlab implementation of compared algorithms released by the authors of MSC,[2] HOSVD,[3] and TSC via multilinear PageRank.[4] We followed guidance from the original papers to set their hyperparameters. All experiments were performed on a Linux machine with one 2.4GHz Intel Core and 16G memory. We have released the Matlab implementation for MOSC.[5]

[2] `https://github.com/arbenson/higher-order-organization-matlab`
[3] `http://sml.csa.iisc.ernet.in/SML/code/Feb16TensorTraceMax.zip`
[4] `https://github.com/dgleich/mlpagerank`
[5] `https://bitbucket.org/Yan_Sheffield/mosc/`

Table 1. Statistics of the five networks. PBlogs has significantly higher triangle density than the others. #Interaction edges are the number of edges among the ground-truth communities.

Network	$\|V\|$	$\|E\|$	Triangle density	#Communities	#Interaction edges
Zachary	34	78	1.32	2	11
Dolphin	62	159	1.53	2	6
Polbooks	105	441	5.33	3	70
Football	115	613	7.04	12	219
PBlogs	1,490	16,716	67.80	2	1,576

4.2 Performance Comparison

The clustering results on five networks are summarised in Table 2. We have three observations:

1. MOSC-GL(Auto-λ) improves by 5.4% (from 0.796 to 0.839) over the best competing algorithm (SC-Shi) in average NMI of four networks with low triangle density. For the remaining PBlogs with high triangle density, MOSC-RW(Auto-λ) improves by 18.9 times (from 0.023 to 0.458) over the best competing algorithm (MSC). It demonstrates that automatic determination of λ is effective in these scenarios. We visualise the obtained clusters by MOSC-GL(Auto-λ) for Polbooks in Fig. 2.
2. MOSC-GL and MOSC-RW have different performance w.r.t triangle density of networks. We will give a discussion at the end of this section.
3. The value of mis-clustered node (ϵ_N) does not have a monotonic relationship with the value of mis-clustered edge (ϵ_E)/triangle (ϵ_T). Fig. 3 shows an example that HOSVD vs. TSC on ϵ_N has non-monotonic relationship with ϵ_T. Specifically, TSC achieves lower value of error nodes than that of HOSVD, but it loses more triangles than HOSVD. It demonstrates that the mis-clustered node cannot reflect errors in structures, while our proposed structure-aware metrics (ϵ_E/ϵ_T) can gain insights on the quality of structure preservation.

From the above results, we see that MOSC-GL and MOSC-RW have different performance on networks with different triangle densities. MOSC-RW tends to be better for networks with high triangle densities while MOSC-GL tends to be better for networks with low triangle densities. For MOSC-GL, $\mathbf{W}_T$ can dominate $\mathbf{W}_X$ in $\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}$, especially for dense networks. Each entry of $\mathbf{W}_T$ denotes the number of triangles containing the corresponding edge while $\mathbf{W}$ is a binary matrix. Therefore, for most non-zero pairs (i, j), $\mathbf{W}_T(i, j)$ is much larger than $\mathbf{W}(i, j)$, especially for dense networks, which makes determining the appropriate λ more difficult. In contrast, MOSC-RW does not have such issue since $\mathbf{A}$ and $\mathbf{P}$ are normalised and thus they are in similar scales before the linear combination. According to the above, we recommend MOSC-GL for networks with low triangle density while MOSC-RW for networks with high triangle density.

Table 2. Clustering performance of algorithms with the best cut criteria. The best is in **bold** and the second best is underlined. A larger NMI indicates a better result, while a smaller $\epsilon_N/\epsilon_E/\epsilon_T$ indicates a better result. Note that there are ties. Avg. NMI is the average NMI for each algorithm on the first four datasets.

		Edge based		Triangle based			MOSC-RW		MOSC-GL	
Method		SC-Shi	SC-Ng	HOSVD	MSC	TSC	$\lambda = 0.5$	Auto-λ	$\lambda = 0.5$	Auto-λ
Zachary	NMI	**0.837**	**0.837**	0.069	0.732	0.677	**0.837**	**0.837**	**0.837**	**0.837**
	ϵ_N	**0.029**	**0.029**	0.412	0.059	0.059	**0.029**	**0.029**	**0.029**	**0.029**
	ϵ_E	**0.026**	**0.026**	0.436	0.038	0.038	**0.026**	**0.026**	**0.026**	**0.026**
	ϵ_T	**0.022**	**0.022**	0.356	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**	**0.022**
Football	NMI	0.883	0.905	0.896	0.924	0.866	0.924	0.924	0.900	**0.931**
	ϵ_N	0.200	0.130	0.139	0.087	0.226	0.087	0.087	0.130	**0.078**
	ϵ_E	0.103	0.060	0.059	**0.011**	0.114	**0.011**	**0.011**	0.059	**0.011**
	ϵ_T	0.122	0.062	0.048	**0.002**	0.136	**0.002**	**0.002**	0.048	**0.002**
Polbooks	NMI	0.575	0.542	0.092	0.542	0.180	0.575	0.575	0.563	**0.589**
	ϵ_N	**0.162**	0.171	0.533	0.171	0.524	**0.162**	**0.162**	**0.162**	**0.162**
	ϵ_E	0.061	0.075	0.420	0.077	0.637	0.061	0.061	0.063	**0.048**
	ϵ_T	0.013	0.018	0.418	0.014	0.686	0.013	0.013	0.013	**0.002**
Dolphin	NMI	0.889	0.889	0.081	0.536	0.582	0.889	0.889	0.889	**1.000**
	ϵ_N	0.016	0.016	0.306	0.113	0.097	0.016	0.016	0.016	**0.000**
	ϵ_E	0.006	0.006	0.270	0.063	0.050	0.006	0.006	0.006	**0.000**
	ϵ_T	**0.000**	**0.000**	0.305	**0.000**	0.011	**0.000**	**0.000**	**0.000**	**0.000**
Avg. NMI		0.796	0.793	0.285	0.684	0.576	0.806	0.806	0.797	**0.839**
PBlogs	NMI	0.007	0.007	0.014	0.023	-	0.012	**0.458**	0.098	0.016
	ϵ_N	0.450	0.491	0.454	0.412	-	0.442	**0.154**	0.321	0.434
	ϵ_E	0.437	0.437	0.437	0.434	-	0.437	**0.011**	0.437	0.437
	ϵ_T	0.360	0.360	0.360	0.360	-	0.360	**0.005**	0.360	0.360

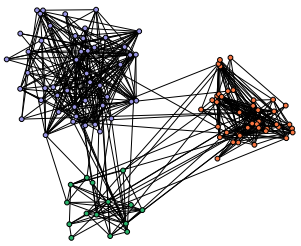


Fig. 2. Communities in Polbooks detected by MOSC-GL(Auto-λ).

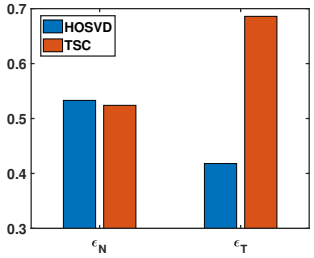
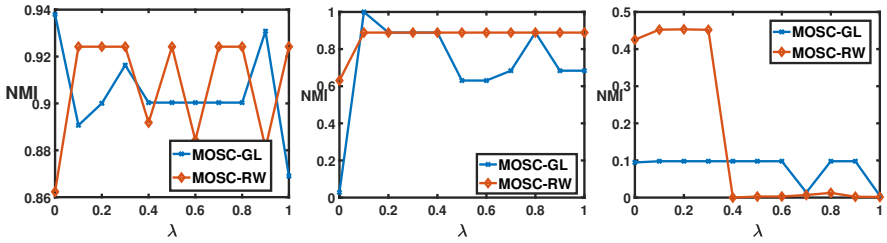


Fig. 3. Non-monotonic relationship between ϵ_N and ϵ_T on Polbooks.

4.3 Effect of λ

The mixing parameter λ is the only hyperparamter in MOSC. To gain insight of MOSC, we conduct sensitivity analysis on λ as shown in Fig. 4 w.r.t. NMI. We can see that the choice of λ can significantly affect the performance while there are large regions of stable performance as well. This was the motivation



(a) Football (b) Dolphin (c) PBlogs

Fig. 4. Sensitivity analysis of λ on Football, Dolphin and PBlogs w.r.t NMI.

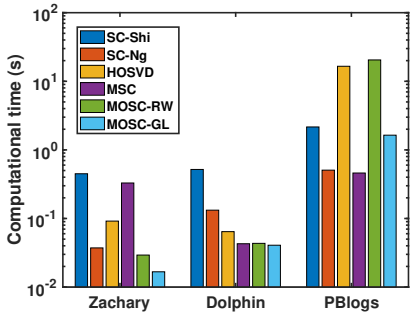


Fig. 5. Computational time (in log scale).

of developing schemes to automatically determine the best λ. For PBlogs, Table 2 shows that MOSC-RW achieves significantly better performance than the others. From Fig. 4(c), MOSC-RW does not have good performance for large λ values (>0.4). Fortunately, benefiting from automatic λ determination scheme, an outstanding performance has been achieved.

4.4 Computational Time

Figure 5 compares the computational time of different methods on three networks, where k-means was used to obtain the final clusters to avoid the effect of cut criteria. MOSC-GL is more efficient than MOSC-RW in all cases because MOSC-GL does not have tensor construction and tensor dimension reduction. For PBlogs, HOSVD and MOSC-RW are time-consuming, because both involve tensor construction and operations that are much more expensive than that in networks with low triangle density. SC-shi and SC-Ng are slower than MOSC-RW and MOSC-GL on Zachary and Dolphin since they use use more time on converging of k-means step. But for PBlogs that is dense and large, MOSC-RW spends lots of time on constructing the triangle tensor while MOSC-GL is scalable to construct the triangle matrix. Our future work will apply our algorithms to large network datasets.

5 Conclusion

This paper proposed two mixed-order spectral clustering (MOSC) methods, MOSC-GL and MOSC-RW, which model both second-order and third-order structures for network clustering. MOSC-GL combines edge and triangle adjacency matrices with theoretical performance guarantee. MOSC-RW combines first-order and second-order random walks with a probabilistic interpretation. We designed a scheme to automatically determine the mixing parameter and proposed new structure-aware error metrics for structure evaluation. Experiments on five real-world networks showed that MOSC algorithms outperform existing SC methods on the whole.

Acknowledgement

This work is supported by the Amazon Research Awards. This article solely reflects the opinions and conclusions of its authors and not Amazon.

References

1. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 us election: divided they blog. In: Proceedings of the 3rd International Workshop on Link Discovery. pp. 36–43. ACM (2005)
2. Ana, L., Jain, A.K.: Robust data clustering. In: CVPR. vol. 2, pp. II–II (2003)
3. Benson, A.R., Gleich, D.F., Leskovec, J.: Tensor spectral clustering for partitioning higher-order network structures. In: SDM. pp. 118–126. SIAM (2015)
4. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. Science **353**(6295), 163–166 (2016)
5. Chakraborty, T., Dalmia, A., Mukherjee, A., Ganguly, N.: Metrics for community analysis: A survey. ACM Computing Surveys (CSUR) **50**(4), 54 (2017)
6. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Physical Review E **70**(6), 066111 (2004)
7. Fiedler, M.: Algebraic connectivity of graphs. Czechoslovak Mathematical Journal **23**(2), 298–305 (1973)
8. Flake, G.W., Tarjan, R.E., Tsioutsiouliklis, K.: Graph clustering and minimum cut trees. Internet Mathematics **1**(4), 385–408 (2004)
9. Fortunato, S.: Community detection in graphs. Physics Reports **486**, 75–174 (2010)
10. Ghoshdastidar, D., Dukkipati, A.: Consistency of spectral partitioning of uniform hypergraphs under planted partition model. In: NIPS (2014)
11. Ghoshdastidar, D., Dukkipati, A.: Uniform hypergraph partitioning: Provable tensor methods and sampling techniques. JMLR **18**(50), 1–41 (2017)
12. Granovetter, M.S.: The strength of weak ties. In: Social Networks, pp. 347–367. Elsevier (1977)
13. Kossinets, G., Watts, D.J.: Empirical analysis of an evolving social network. Science **311**(5757), 88–90 (2006)
14. Lee, J.R., Gharan, S.O., Trevisan, L.: Multiway spectral partitioning and higher-order cheeger inequalities. Journal of the ACM (JACM) **61**(6), 37 (2014)

15. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: WWW. pp. 631–640. ACM (2010)
16. Li, P., He, N., Milenkovic, O.: Quadratic decomposable submodular function minimization. In: NIPS. pp. 1062–1072 (2018)
17. Li, P., Milenkovic, O.: Inhomogoenous hypergraph clustering with applications. In: NIPS. pp. 2305–2315 (2017)
18. Lusseau, D.: The emergent properties of a dolphin social network. Proceedings of the Royal Society of London B: Biological Sciences **270**, S186–S188 (2003)
19. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
20. Meila, M., Shi, J.: Learning segmentation by random walks. In: NIPS (2001)
21. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science **298**(5594) (2002)
22. Newman, M.E.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences **103**(23), 8577–8582 (2006)
23. Newman, M.E.: Communities, modules and large-scale structure in networks. Nature Physics **8**(1) (2012)
24. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS. pp. 849–856 (2002)
25. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: KDD. pp. 1105–1114. ACM (2016)
26. Porter, M.A., Onnela, J.P., Mucha, P.J.: Communities in networks. Notices of the AMS **56**(9), 1082–1097 (2009)
27. Schaeffer, S.E.: Graph clustering. Computer Science Review **1**(1), 27–64 (2007)
28. Serrour, B., Arenas, A., Gómez, S.: Detecting communities of triangles in complex networks using spectral optimization. Computer Communications **34**(5) (2011)
29. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. on PAMI **22**(8), 888–905 (2000)
30. Sporns, O., Kötter, R.: Motifs in brain networks. PLoS Biology **2**(11), e369 (2004)
31. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW. pp. 1067–1077 (2015)
32. Tsourakakis, C.E., Pachocki, J., Mitzenmacher, M.: Scalable motif-aware graph clustering. In: WWW. pp. 1451–1460 (2017)
33. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17**(4), 395–416 (2007)
34. Wu, T., Benson, A.R., Gleich, D.F.: General tensor spectral co-clustering for higher-order data. In: NIPS. pp. 2559–2567 (2016)
35. Xu, J., Wickramarathne, T.L., Chawla, N.V.: Representing higher-order dependencies in networks. Science Advances **2**(5), e1600028 (2016)
36. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems **42**(1), 181–213 (2015)
37. Yin, H., Benson, A.R., Leskovec, J., Gleich, D.F.: Local higher-order graph clustering. In: KDD. pp. 555–564. ACM (2017)
38. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research **33**(4), 452–473 (1977)
39. Zhou, D., Zhang, S., Yildirim, M.Y., Alcorn, S., Tong, H., Davulcu, H., He, J.: A local algorithm for structure-preserving graph cut. In: KDD. pp. 655–664 (2017)