
Article:
Ge, Y., Peng, P. and Lu, H. (2021) Mixed-order spectral clustering for complex networks. Pattern Recognition, 117. 107964. ISSN: 0031-3203

https://doi.org/10.1016/j.patcog.2021.107964

Mixed-Order Spectral Clustering for Complex Networks

Yan Ge, Pan Peng, Haiping Lu*

Department of Computer Science, The University of Sheffield, 211 Portobello, Sheffield S1 4DP, United Kingdom

Abstract

Spectral clustering (SC) is a popular approach for gaining insights from complex networks. Conventional SC focuses on *second-order* structures (e.g. edges) without direct consideration of *higher-order* structures (e.g. triangles). This has motivated SC extensions that directly consider higher-order structures. However, both approaches are limited to considering a single order. To address this issue, this paper proposes a novel *Mixed-Order* Spectral Clustering (MOSC) framework to model both second-order and third-order structures simultaneously. To model mixed-order structures, we propose two new methods 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. Moreover, we design mixed-order cut criteria to enable existing SC methods to preserve mixed-order structures, and develop new mixed-order evaluation metrics for structure-level evaluation. Experiments on community detection and superpixel segmentation show 1) the superior performance of the MOSC methods over existing SC methods, 2) enhanced performance of conventional SC due to mixed-order cut criteria, and 3) new insights of output clusters offered by the mixed-order evaluation metrics.

Keywords: Spectral clustering, higher-order structures, mixed-order structures

*Corresponding author. Address: The University of Sheffield, 211 Portobello, Sheffield, S1 4DP, United Kingdom. Tel: +44 (0) 114 222 1853. Email: h.lu@sheffield.ac.uk

Email addresses: yge5@sheffield.ac.uk (Yan Ge), p.peng@sheffield.ac.uk (Pan Peng), h.lu@sheffield.ac.uk (Haiping Lu)

1. Introduction

Networks (a.k.a. graphs) are important data structures that abstract relations between discrete objects, such as social networks and brain networks [1]. 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 [2, 3] and image segmentation [4].

Clustering aims to divide the data set into *clusters* (or *communities*) such that the nodes assigned to a particular cluster are similar or well connected in some predefined sense [5, 6]. It helps us reveal functional groups hidden in data. As a popular clustering method, conventional spectral clustering (SC) [7, 8] encodes pairwise similarity into an adjacency matrix. Such encoding inherently restricts SC to *second-order structures* [1], 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 [9], which we call a higher-order structure.

Higher-order structures consist of at least three nodes (e.g. triangles, 4-vertex cliques) [1]. 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 [11] and social networks [12]. More importantly, higher-order structures allow for more flexible

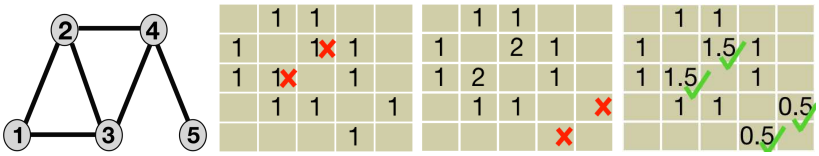
[1] Edges are considered as first-order structures in [9] but second-order structures in [10]. We follow the terminologies in the latter [10] so that the "order" here refers to the number of nodes involved in a particular structure.

modelling. For instance, considering directions of edges, there exist 13 different third-order structures, but only two different second-order structures [13]. 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 four approaches: 1) The first approach constructs an affinity tensor to encode higher-order structures and then reduces it to a matrix [14], followed by conventional SC [8]. These methods, such as tensor trace maximisation (TTM) [15], are developed in a closely related problem of hypergraph clustering that considers "hyperedges" connecting multiple nodes. 2) The second approach develops higher-order SC by constructing a transition tensor based on random walks model and then reduces it to a matrix for conventional SC, such as tensor spectral clustering (TSC) [9]. 3) The third approach uses a counting and reweighting scheme to capture higher-order structures and reveal clusters [16], such as motif-based SC (MSC) [1].[2] 4) The fourth approach is higher-order local clustering aiming to reduce computation cost [17], such as High-Order Structure Preserving Local Clustering (HOSPLOC) [10].

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 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);

[2] We have verified that TTM and MSC are equivalent, nevertheless.



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

Figure 1: Motivation of mixed-order structures: the second and third order structures in (a) can not be fully captured by edge/triangle adjacency matrix in (b) or (c). A mixed adjacency matrix in (d) can capture both.

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 novel *Mixed-Order Spectral Clustering* (MOSC) framework 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.

The MOSC framework can be decomposed into three blocks, and we summarise our contributions based on building blocks as following:

1. **Mixed-order structure**. We propose two mixed-order structure approaches: one based on Graph Laplacian (GL) and the other based on Random Walks (RW). Based on these approaches, we develop two new algorithms MOSC-GL and MOSC-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. Besides, mixing parameter (ranging from 0 to 1) can be automatically decided based on cut criteria and triangle density. See Sec. 3.1, Sec. 3.2 and Sec. 3.4.

2. **Mixed-order cut criterion**. To enable existing single-order SC methods [18, 1] to preserve mixed-order structures, we consider cut criteria of different orders from the order used to encode a structure (e.g. second-order SC with a third-order cut criterion). We then empirically study the effectiveness of mixed-order cut criterion, finding that this strategy can enhance the performance of conventional SC methods. See Sec. 3.5 and 4.2.
3. **Mixed-order evaluation metric**. Given ground truth, existing works only consider the number of error nodes to evaluate the quality of output clusters [17, 15]. However, it may fail to reflect the errors in structures. To address this issue, we propose structure-aware error metrics to evaluate performance at the level of structures. Additionally, we design mixed-order evaluation metrics by further utilising proposed metrics of different orders from the order used to encode a structure (e.g. evaluate second-order SC with a third-order-aware error metric), which aims to gain new insights on the quality of structure preservation. See Sec. 3.6 and 4.3.

2. Preliminaries

2.1. 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.

2.2. 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 $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}\mathbf{D}^{-\frac{1}{2}}$.

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 [1]. For implementation, based on [1], we formulate $\mathbf{W}_T = \mathbf{A} \cdot \mathbf{A} \circ \mathbf{A}$, where $\mathbf{A}$ is an edge adjacency matrix, '$\cdot$' is matrix multiplication, and '$\circ$' is Hadamard product. 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 $\mathbf{D}_T(i,i) = \sum_{j=1}^{n} \mathbf{W}_T(i,j)$.

2.3. First-Order Random Walks for Second-Order Structures

We define a second-order transition matrix $\mathbf{P}$ by normalising the adjacency matrix $\mathbf{W}$ to represent edge structures as [7] $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. 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 random walk process on graph G [7]. From random walk perspective, SC can be interpreted as trying to find a partition of the graph such that the random walk stays long within the same cluster and seldom jumps between clusters [19].

2.4. Second-Order Random Walks for Third-Order Structures

Benson *et al.* [9] extend the above using a three-dimensional transition tensor to encode triangle structures. They 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. All entries in $\mathcal{T}$ with a permutation of indices $i,\ j,\ k$ have the same value (hence symmetric). 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 third-order 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.* [9] set $\mathcal{P}(i,j,k)$ to $\frac{1}{n}$.

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 $\mathbf{v}$.
4: $\{S, \bar{S}\} \leftarrow$ sweep cut of $\mathbf{v}$ w.r.t. some cut criteria.

Table 1: Edge-based and triangle-based cut criteria.

	Edge-based cuts	**Triangle-based cuts**
Conductance (ϕ)	$\phi_2(S) = \frac{cut_2(S)}{\min(vol_2(S), vol_2(\bar{S}))}$ [20]	$\phi_3(S) = \frac{cut_3(S)}{\min(vol_3(S), vol_3(\bar{S}))}$ [9]
Ncut (η)	$\eta_2(S) = cut_2(S)(\frac{1}{vol_2(S)} + \frac{1}{vol_2(\bar{S})})$ [18]	$\eta_3(S) = cut_3(S)(\frac{1}{vol_3(S)} + \frac{1}{vol_3(\bar{S})})$ [21]
Nassoc (ξ)	$\xi_2(S) = \frac{assoc_2(S)}{vol_2(S)} + \frac{assoc_2(\bar{S})}{vol_2(\bar{S})}$[18]	$\xi_3(S) = \frac{assoc_3(S)}{vol_3(S)} + \frac{assoc_3(\bar{S})}{vol_3(\bar{S})}$[15]
Expansion (α)	$\alpha_2(S) = \frac{cut_2(S)}{\min(\lvert S\rvert, \lvert\bar{S}\rvert)}$ [22]	$\alpha_3(S) = \frac{cut_3(S)}{\min(\lvert S\rvert, \lvert\bar{S}\rvert)}$ [9]

2.5. Spectral Clustering Basics

Bi-partitioning SC (Algorithm 1) first constructs a matrix $\mathbf{B}$ to encode structures in the input graph G. 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. Table 1 lists eight cut criteria, both second and third orders (edge and triangle based). We omit G in the criterion notation $\tau(S; G)$ in the table for simplicity.

2.6. Cheeger Inequalities and Cut Criteria

Given $G = (V, E)$ and a subset $S \subseteq V$, let $\bar{S}$ denote the complement of S. Let $cut_2(S; G)$ denote the *edge cut* of S, i.e, the number of edges between S and $\bar{S}$ in G. Let $vol_2(S; G)$ denote the *edge volume* of S, i.e. the total degrees of vertices in S, and $assoc_2(S; G)$ is the total degrees in the subgrapgh induced by vertices in S. The *edge conductance* of S is defined as $\phi_2(S; G) = \frac{cut_2(S;G)}{\min\{vol_2(S;G), vol_2(\bar{S};G)\}}$. Other popular edge-based cut criteria are shown in Table 1 (left column). The

classical Cheeger inequality below relates the conductance of the sweep cut of SC to the minimum conductance value of the graph [23].

Lemma 1 (Second-Order Cheeger Inequality [23])**.** *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* $\phi_2^* = \min_{\mathrm{S}\subset\mathrm{V}}\ \phi_2(\mathrm{S}; G)$ *is the minimum conductance over any set of vertices.*

Let $cut_3(S; G)$ 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 $\bar{S}$, and let $assoc_3(S; G)$ count the number of vertices in triangles in the subgraph induced by vertices in S. Let $vol_3(S; G)$ denote the *triangle volume* of S, i.e. the number of triangle endpoints in S. The *triangle conductance* [9] of S is defined as $\phi_3(S; G) = \frac{cut_3(S;G)}{\min\{vol_3(S;G), vol_3(\bar{S};G)\}}$. It is further proved in [1] that for any $S \subset V$, $\phi_3(S; G) = \phi_2(S; G_T)$, which leads to the following third-order Cheeger inequality. Other popular triangle-based cut criteria are summarised in Table 1 (right column).

Lemma 2 (Third-order Cheeger Inequality [1])**.** *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* $\phi_3^* = min_{\mathrm{S}\subset\mathrm{V}}\ \phi_3(\mathrm{S}; G)$*.*

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 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 and second order random walks, providing a probabilistic interpretation. Moreover, we define new mixed-order cut criteria to enable existing single-order SC methods to preserve mixed-order structures, and propose mixed-order evaluation metrics to evaluate clustering

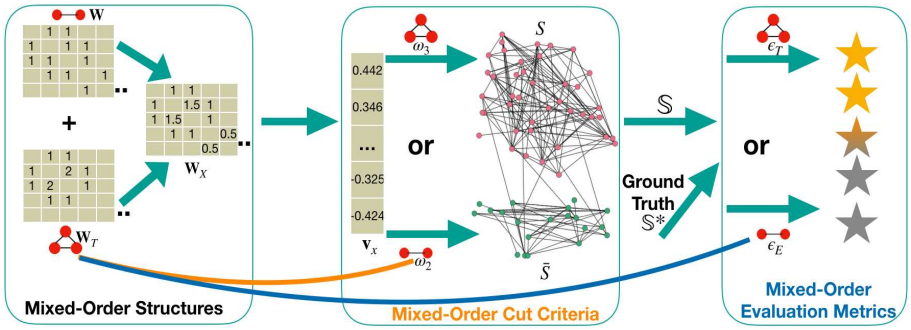


Figure 2: Illustration of mixed-order structures (i.e. MOSC-GL), cut criteria and evaluation metrics. The left shows a mixed-order adjacency matrix $\mathbf{W}_X$ that linearly combines an edge-based matrix ($\mathbf{W}$) and a triangle-based matrix ($\mathbf{W}_T$). The middle shows output clusters $\mathbb{S} = \{S, \bar{S}\}$ generated by either a second-order (ω_2) or third-order cut criterion (ω_3). An orange line from the left to the middle indicates an instance of a mixed-order cut criterion where a sorted dominant eigenvector ($\mathbf{v}_X$) derived from only triangle-based adjacency matrix is split by an edge-based cut criterion. The right shows that given the ground-truth, we evaluate the quality of output clusters $\mathbb{S} = \{S, \bar{S}\}$ by either a second-order (ϵ_E) or a third-order structural error metric (ϵ_T). A blue line from the left to right indicates an instance of a mixed-order evaluation metric where output clusters $\mathbb{S}$ derived from only a triangle-based adjacency matrix ($\mathbf{W}_T$) are evaluated by an edge-based error metric (ϵ_E).

methods at the level of structures. The proposed MOSC framework is illustrated in Fig. 2.

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

Let G_X denote an undirected weighted graph with adjacency matrix $\mathbf{W}_X$, as illustrated in Fig. 2 (the left block). We can define a mixed-order Laplacian $\mathbf{N}_X$ and its normalised version $\mathbf{L}_X$ as $\mathbf{N}_X = \mathbf{D}_X - \mathbf{W}_X = (1-\lambda)\mathbf{N}_T + \lambda\mathbf{N}, \mathbf{L}_X = \mathbf{D}_X^{-\frac{1}{2}}\mathbf{N}_X\mathbf{D}_X^{-\frac{1}{2}}$. Then, we compute the eigenvector corresponding to the second

Algorithm 2 MOSC-GL

Require: $G = (V, E)$, a mixing parameter λ

Ensure: Two node sets $\{S, \bar{S}\}$

1: Construct the edge adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$.
2: Construct the triangle adjacency matrix $\mathbf{W}_T \in \mathbb{R}^{n \times n}$.
3: Let $\mathbf{D}$ be diagonal with $\mathbf{D}(i,i) = \sum_i^n \mathbf{W}(i,j)$.
4: Let $\mathbf{D}_T$ be diagonal with $\mathbf{D}_T(i,i) = \sum_i^n \mathbf{W}_T(i,j)$.
5: $\mathbf{W}_X = (1-\lambda)\mathbf{W}_T + \lambda\mathbf{W}$.
6: $\mathbf{D}_X = (1-\lambda)\mathbf{D}_T + \lambda\mathbf{D}$.
7: $\mathbf{N}_X = \mathbf{D}_X - \mathbf{W}_X = (1-\lambda)\mathbf{N}_T + \lambda\mathbf{N}$.
8: $\mathbf{L}_X = \mathbf{D}_X^{-\frac{1}{2}} \mathbf{N}_X \mathbf{D}_X^{-\frac{1}{2}}$.
9: Compute the second smallest eigenvector $\mathbf{v}_X$ of $\mathbf{L}_X$.
10: $\mathbf{v}_X \leftarrow$ Sort entries of $\mathbf{D}_X^{-\frac{1}{2}} \mathbf{v}_X$.
11: $\{S, \bar{S}\} \leftarrow$ Sweep cut on $\mathbf{v}_X$ w.r.t. some cut criteria.

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.* [8] and only considers second-order structures. When $\lambda = 0$, MOSC-GL is equivalent to motif-based SC [1]. 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 $cut_X(S;G) = (1-\lambda)cut_3(S;G) + \lambda cut_2(S;G)$, and $vol_X(S;G) = (1-\lambda)vol_3(S;G) + \lambda vol_2(S;G)$, respectively. Then, we define the *mixed-order conductance* of S as: $\phi_X(S;G) = \frac{cut_X(S;G)}{\min(vol_X(S;G), vol_X(\bar{S};G))}$, 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. To prove Theorem 1, we need the following Lemma.

Lemma 3 (Lemma 4 and 1 in [1]). *Let $G = (V, E)$ be an undirected, unweighted graph and let G_T be the weighted graph for the triangle adjacency matrix. Then for any $S \subset V$, it holds that $cut_3(S;G) = \frac{1}{2}cut_2(S;G_T)$, $vol_3(S;G) = \frac{1}{2}vol_2(S;G_T)$.*

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

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

$$\frac{1}{2}\phi_2(S;G_X) \leq \phi_X(S;G) \leq 2\phi_2(S;G_X). \tag{3}$$

Assume for now that the above inequality (3) 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 $\phi_X(R;G) = \phi_X^* = \min_{S\subseteq V}\phi_X(S;G)$. Then by inequality (3), we have $\phi_X(T^*;G) \leq 2\phi_2(T^*;G_X) \leq 2\sqrt{\psi^*} \leq 2\sqrt{\phi_2(R;G_X)} \leq 2\sqrt{2\phi_X(R;G)} = 2\sqrt{2\phi_X^*}$. This will then finish the proof. Therefore, we only need to prove the inequality (3). We will make use of the Lemma 3 from [1].

By Lemma 3, we have $cut_X(S;G) = (1-\lambda)cut_3(S;G) + \lambda cut_2(S;G) = (1-\lambda)\frac{1}{2}cut_2(S;G_T) + \lambda cut_2(S;G)$, $vol_X(S;G) = (1-\lambda)vol_3(S;G) + \lambda vol_2(S;G) = (1-\lambda)\frac{1}{2}vol_2(S;G_T) + \lambda vol_2(S;G)$.

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 $cut_2(S;G_X) = (1-\lambda)cut_2(S;G_T) + \lambda cut_2(S;G), vol_2(S;G_X) = (1-\lambda)vol_2(S;G_T) + \lambda vol_2(S;G)$. The above equations imply that for any set S, $\frac{1}{2}cut_2(S;G_X) \leq cut_X(S;G) \leq cut_2(S;G_X), \frac{1}{2}vol_2(S;G_X) \leq vol_X(S;G) \leq vol_2(S;G_X)$. The last inequality also implies that for any S, $\frac{1}{2}vol_2(\bar{S};G_X) \leq vol_X(\bar{S};G) \leq vol_2(\bar{S};G_X)$.

Therefore, by the definition of $\phi_X(S;G)$, we have

$$\begin{aligned}
\phi_X(S;G) &\leq \frac{cut_2(S;G_X)}{\min(\frac{1}{2}vol_2(S;G_X), \frac{1}{2}vol_2(\bar{S};G_X))} \\
&= 2\phi_2(S;G_X), \\
\phi_X(S;G) &\geq \frac{\frac{1}{2}cut_2(S;G_X)}{\min(vol_2(S;G_X), vol_2(\bar{S};G_X))} \\
&= \frac{1}{2}\phi_2(S;G_X).
\end{aligned}$$

This completes the proof of the inequality (3). □

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 $O(n^3)$. In general, for a network with n nodes and m edges, building a triangle adjacency matrix $\mathbf{W}_T$ is at least as hard as the problem of triangle detection (i.e. to check if a network contains a triangle or not), which in turn is conjectured to require $m^{1+\delta+o(1)}$ time, for some constant $\delta > 0$ [24]. In this paper, we build $\mathbf{W}_T$ by checking each edge and then finding all possible common neighbours, which requires $O(mn)$ time. For the calculation of the second eigenvector of $\mathbf{L}_X$, 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/triangle 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 [19]. 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 [9]. 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 $\mathcal{T}$, we define a third-order transition tensor $\mathcal{P}$ as Eq. (2). Each entry of $\mathcal{P}$ represents the transition probability of a random walk such that the probability of jumping to a state j depends on the last two states i and k [25]. 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 $\mathbf{H} = (1-\lambda)\mathbf{A} + \lambda\mathbf{P}$. 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 Malik [18] 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.* [9], so we name it as simplified TSC (STSC). In the intermediate case, λ controls the trade-off.

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

Algorithm 3 MOSC-RW

Require: $G = (V, E)$, a mixing parameter λ

Ensure: Two node sets $\{S, \bar{S}\}$

1: Construct the adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$.
2: Construct the adjacency tensor $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$.
3: **for** $1 \leqslant i, j, k \leqslant n$ **do**
4: **if** $\sum_{m=1}^{n} \mathcal{T}(i, m, k) \neq 0$ **then**
5: $\mathcal{P}(i, j, k) = \mathcal{T}(i, j, k) \ / \sum_{m=1}^{n} \mathcal{T}(i, m, k)$.
6: **else**
7: $\mathcal{P}(i, j, k) = 0$.
8: **end if**
9: **end for**
10: $\mathbf{T}_k \leftarrow \mathcal{P}(:, :, k)$ for $k = 1, \cdots, n$.
11: Compute the reduced similarity matrix $\mathbf{A}$.
12: Let $\mathbf{D}$ be diagonal with $\mathbf{D}_{ii} = \sum_{i}^{n} \mathbf{W}(i, j)$.
13: $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$.
14: $\mathbf{H} = (1 - \lambda)\mathbf{A} + \lambda\mathbf{P}$.
15: Compute the second largest eigenvector $\mathbf{v}$ of $\mathbf{H}$.
16: $\mathbf{v} \leftarrow$ Sorting entries of $\mathbf{v}$.
17: $\{S, \bar{S}\} \leftarrow$ Sweep cut on $\mathbf{v}$ w.r.t. some cut criteria.

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.

The computation of the second largest eigenvector of $\mathbf{H}$ in step 15 is another costly procedure, and its complexity depends on the sparsity of $\mathcal{T}$ and $\mathbf{P}$. Let a and p be the number of non-zeros entries in $\mathcal{T}$ and $\mathbf{P}$, respectively. In theory, the number of non-zero entries of $\mathbf{H}$ can be $O(a + p)$, and an eigenvector can be computed via power iterations, with each iteration at $O(a + p)$.

3.3. Multiple Clusters and Higher-order Cheeger Inequalities of MOSC

To cluster a network into $k > 2$ clusters based on MOSC-GL and MOSC-RW, we follow the conventional SC [19]. 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 [1] and [26], 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 [26].

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 optimal value λ^* from a set Λ based on the quality of output clusters [27, 28, 29]. For bi-partitioning networks, the cut criterion used to obtain output clusters can help to determine the best λ^* from Λ. For multiple partitioning networks, we can use the sum of triangle densities of the individual cluster 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. $\lambda^* = \arg\ \min_{\lambda' \in \Lambda} \tau(S_{\lambda'})$ 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. Mixed-order Cut Criteria

A cut criterion measures the quality of output clusters when performing the sweep cut procedure. However, conventional SC is limited to use edge-based cut criteria [18, 8], while triangle-based SC is limited to use triangle-based

cut criteria [1, 9]. Thus, to enable existing single-order SC methods [18, 1] to preserve mixed-order structures, we consider cut criteria of different orders from the order used to encode a structure (e.g. second-order SC with a third-order cut criterion). Therefore, we formally define a mixed-order cut criterion as follows:

Definition 1. *(Mixed-order Cut Criterion) Given a graph $G = (V, E)$, a mixed-order cut criterion performs a triangle-based cut criterion $\omega_3 \in \{\phi_3, \eta_3, \xi_3, \alpha_3\}$ on a sorted dominant eigenvector $\mathbf{v}_e$ derived from an edge-based similarity matrix $\mathbf{B}$ of G, or performs an edge-based cut criterion $\omega_2 \in \{\phi_2, \eta_2, \xi_2, \alpha_2\}$ on a sorted dominant eigenvector $\mathbf{v}_t$ derived from a triangle-based similarity matrix $\mathbf{T}$.*

Mixed-order cut criteria are illustrated in the middle block of Fig.2. In this paper, we will empirically study the effectiveness of eight cut criteria including edge- and triangle-based (Table 1) ones on representative edge- and triangle-based SC algorithms. Note that maximisation of $\mathrm{Nassoc}_2(S)$ ($\xi_2(S)$) and minimisation of $\mathrm{Ncut}_2(S)$ ($\eta_2(S)$) are equivalent [18], but we theoretically prove that maximisation of $\mathrm{Nassoc}_3(S)$ ($\xi_3(S)$) and minimisation of $\mathrm{Ncut}_3(S)$ ($\eta_3(S)$) are not equivalent, see Proposition 1 (omit G for simplicity). Thus, we have seven different cut criteria from Table 1.

Proposition 1 (Non-equivalent Relation between $\xi_3(S)$ and $\eta_3(S)$)**.** *Given* $\mathrm{G} = (\mathrm{V}, \mathrm{E})$*, and a subset* $\mathrm{S} \subseteq \mathrm{V}$*, $\bar{S}$ denote the complement of* S*. We have $\xi_3(S)$ [15] and $\eta_3(S)$ [21], then it holds that maximisation of $\xi_3(S)$ and minimisation of $\eta_3(S)$ are not equivalent.*

Proof. According to $\eta_3(S)$ [21] and $cut_3(S, \bar{S})$ [21], we have

$$\eta_3(S) = \tfrac{2}{3} - \tfrac{1}{3}\left(\tfrac{assoc_3(S)}{vol_3(S)} + \tfrac{assoc_3(\bar{S})}{vol_3(\bar{S})}\right) + \tfrac{1}{3}\left(\tfrac{vol_3(\bar{S}) - assoc_3(\bar{S})}{vol_3(S)} + \tfrac{vol_3(S) - assoc_3(S)}{vol_3(\bar{S})}\right).$$

Then based on $\xi_3(S) = \frac{assoc_3(S)}{vol_3(S)} + \frac{assoc_3(\bar{S})}{vol_3(\bar{S})}$, we have

$$\eta_3(S) = \frac{2}{3} - \frac{1}{3}\xi_3(S) + \frac{1}{3}\left(\frac{vol_3(\bar{S}) - assoc_3(\bar{S})}{vol_3(S)} + \frac{vol_3(S) - assoc_3(S)}{vol_3(\bar{S})}\right).$$

□

3.6. Mixed-order Evaluation 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 [15] or related metrics (e.,g. NMI) [30]. 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 $\epsilon_N(\mathbb{S}^*, \mathbb{S}) = \min_{\sigma} \sum_{c=1}^{k} |S_c^* \oplus S_{\sigma(c)}|$, 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.

To break existing rigid evaluation scenarios based on the level of nodes, we design a flexible way to evaluate the quality of communities by leveraging the level of structures (e.g. triangles). Depending on diverse application scenarios, we can flexibly choose one to evaluate communities. For example, if the triangle structure plays an important role in communities (e.g. in social networks), we can evaluate communities in terms of triangles to truly reflect the quality of communities. A limitation of the above metric is that it fails to truly reflect the errors at the level of structures. Also, our studies 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 smaller number of mis-clustered edges or triangles, and vice versa. This motivates us to propose structure-aware error metrics to measure the quantity of *mis-clustered edges* (ϵ_E) and *triangles* (ϵ_T), respectively. Specifically, we define ϵ_E as $\epsilon_E(\mathbb{S}^*, \mathbb{S}) = \sum_{c=1}^{k} E_N(S_c^*) - \max_{\sigma} \sum_{c=1}^{k} E_N(S_c^* \cap S_{\sigma(c)})$ where $E_N(S)$ is the number of edges in S. We can define ϵ_T similarly by replacing $E_N(S)$ with $T_N(S)$, where $T_N(S)$ is the number of triangles in S.

Based on the proposed structure-aware error metrics, we define a *mixed-order evaluation metrics* that can give us new insights on the preservation of edges and triangles for existing single-order SC. It is illustrated in the right block of Fig. 2.

Table 2: Statistics of the 2,005 networks. The number in parentheses is the median for each range.

Network	$\|V\|$	$\|E\|$	Size	Triangle density	#Interaction edges	#Clusters/network	#Network(s)
DBLP	317K	1.05M	14∼303 (22)	7.4∼167.9 (15.4)	1∼278 (15)	2	500
YouTube	1.13M	2.99M	6∼389 (91)	1∼22.9 (3.73)	1 ∼1054 (89)	2	500
Orkut	3.07M	117M	88∼379 (206)	213.7∼1526 (452.6)	37∼10470 (2411)	2	500
LJ	4.00M	34.7M	33∼193 (98)	116.3∼2968 (422.4)	1∼9179 (1489)	2	500
Zachary	34	78	34	1.32	11	2	1
Dolphin	62	159	62	1.53	6	2	1
Polbooks	105	441	105	5.33	70	3	1
Football	115	613	115	7.04	219	12	1
PBlogs	1490	16716	1490	67.8	1576	2	1
Facebook	22,470	170,912	22,470	35.50	19,590	4	1

Definition 2. *(Mixed-order Evaluation Metrics) Given a graph $G = (V, E)$ and ground-truth $\mathbb{S}^*$, a mixed-order evaluation metric evaluates a candidate partition $\mathbb{S}$ derived from an edge-based similarity matrix $\mathbf{B}$ of G by the triangle-aware error metric ϵ_T, or evaluates a candidate partition $\mathbb{S}$ derived from a triangle-based similarity matrix $\mathbf{T}$ of G by an edge-aware error metric ϵ_E.*

4. Experiments

This section aims to evaluate MOSC against existing SC methods in two applications: community detection and superpixel segmentation. Furthermore, we will explicitly study the effect of mixed-order cut criteria, and gain insights from the newly designed mixed-order evaluation metrics for the community detection task.

4.1. Experimental Settings on Community Detection

Datasets. The experiments were conducted on two popular groups of networks with very different triangle densities: 1) five full real-world networks: Zachary's karate club (Zachary) [31], Dolphin social network (Dolphin) [32], American college football (Football) [33], U.S. politics books (Polbooks) [33], Political blogs (PBlogs) [34] and Facebook[3] [35]; 2) four complex real-world networks: DBLP, YouTube, Orkut, and LiveJournal (LJ) from the Stanford

[3]https://archive.ics.uci.edu/ml/datasets/Facebook+Large+Page-Page+Network

Network Analysis Platform (SNAP) [29].[4] All networks have ground-truth communities available. For the four SNAP networks, we extract paired communities to focus on bi-partitioning problems with the following procedures: 1) For each network, we select communities with the top 500 highest triangle densities, among those communities having no more than 200 nodes (for DBLP, YouTube, and Orkut) or 100 nodes (for LJ) because it has high density; 2) For every community in the top list, we choose another community having the most connections with it, among all the other communities in the respective network (without limiting the community node size). These two communities form a bi-partitioning network. In this way, we extracted 2,000 networks from SNAP. The statistics of networks are summarised in Table 2.

Compared algorithms. We evaluate MOSC-GL and MOSC-RW against the following seven state-of-the-art methods, including both edge-based SC and triangle-based SC, and both global and local methods: 1) SC-Shi [18]: Shi and Malik developed a method aiming to minimise Ncut_2 criterion via a generalised eigenvalue problem of Eq. (2.3). 2) SC-Ng [8]: Ng *et al.* designed a method built upon [18]. Instead of using one dominant eigenvector, it used the first k eigenvectors of $\mathbf{L}$ for performing k partitions and then an additional row normalisation step before k-means. 3) Tensor Spectral Clustering (TSC) [9]: TSC is a higher-order spectral clustering method developed by Benson *et al.* They constructed a transition tensor $\mathcal{P}$ as in Eq. (2) and used an expensive multilinear PageRank algorithm [36] to produce a vector as the weight for reducing the tensor to a matrix via weighted average, followed by conventional SC. 4) Higher-order SVD (HOSVD) [14]: To address the hypergraph clustering problem, this method used an adjacency tensor $\mathcal{T}$ to encode hyperedge, which is equivalent to the adjacency tensor definition in Eq. (1). $\mathcal{T}$ is then reduced to a matrix via computing a modelwise covariance matrix, followed by conventional SC. 5) Motif-based SC (MSC) [1] / Tensor Trace Maximisation (TTM) [15]: MSC is a general higher-order spectral clustering method via re-weighting edges

[4] `https://snap.stanford.edu/data/index.html`

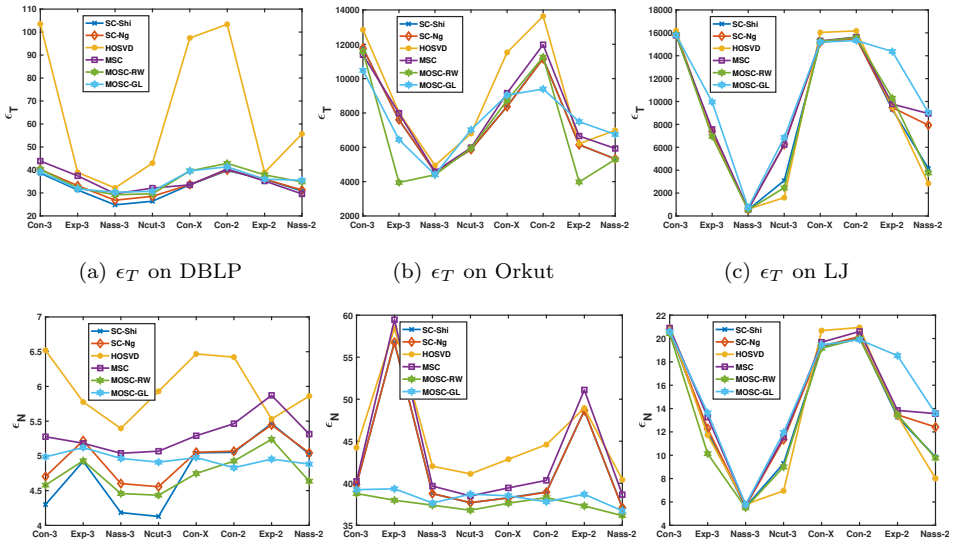
according to the number of motifs containing corresponding edges, followed by conventional SC. TTM is independently proposed but equivalent to MSC, which we have verified both analytically and experimentally. 6) HOSPLOC [10]: This is a higher-order local clustering method aiming for more efficient processing while taking higher-order network structures into account. 7) DeepWalk [37]: DeepWalk adopts an unsupervised Skip-Gram [6] neural network model to learn the embedding of each node. This approach samples random walks from each node,and then maximises the co-occurrence probability among the nodes that appear as neighbours. Following [38], we employ the learned embedding of nodes in a k-means to conduct communities.

We study two versions for each MOSC: MOSC ($\lambda = 0.5$): MOSC with a fixed (recommended) λ value of 0.5; MOSC (Auto-λ): MOSC with automatically determined λ; Additionally, for MOSC-RW, we study simplified TSC (STSC) when $\lambda = 0$.

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 (Eq. (3.6)) and normalised mutual information (NMI) [30, 3]. For the SNAP networks, we show the average results of the 500 bi-partitioning networks.

To define NMI, we need the Shannon entropy for $\mathbb{S}$ that can be defined as $H(\mathbb{S}) = -\sum_{c=1}^{k}(n_{S_c}/n)\log(n_{S_c}/n)$, where n_{S_c} is the number of vertices in community S_c. The mutual information between $\mathbb{S}$ and $\mathbb{S}^*$ can be expressed as $I(\mathbb{S},\mathbb{S}^*) = \sum_{c=1}^{k}\sum_{d=1}^{k}\frac{n_{S_c S_d^*}}{n}\log(\frac{n_{S_c S_d^*}/n}{(n_{S_c}/n)\times(n_{S_d^*}/n)})$, where $n_{S_c S_d^*}$ is the number of vertices shared by communities S_c and S_d^*. The NMI between two partitions $\mathbb{S}$ and $\mathbb{S}^*$ is defined as $\mathrm{NMI}(\mathbb{S},\mathbb{S}^*) = \frac{2I(\mathbb{S},\mathbb{S}^*)}{H(\mathbb{S})+H(\mathbb{S}^*)}$. If $\mathbb{S}$ and $\mathbb{S}^*$ are identical, $\mathrm{NMI}(\mathbb{S},\mathbb{S}^*) = 1$. If $\mathbb{S}$ and $\mathbb{S}^*$ are independent, $\mathrm{NMI}(\mathbb{S},\mathbb{S}^*) = 0$.

Reproducibility. We implemented compared algorithms using Matlab



(a) ϵ_T on DBLP (b) ϵ_T on Orkut (c) ϵ_T on LJ

(d) ϵ_N on DBLP (e) ϵ_N on Orkut (f) ϵ_N on LJ

Figure 3: Mixed-order cut criteria analysis on SC-Shi, SC-Ng, HOSVD, MSC, MOSC-GL($\lambda = 0.5$) and MOSC-RW($\lambda = 0.5$) w.r.t mis-clustered triangles (ϵ_T) and nodes (ϵ_N) for eight cut criteria: Conduct_3(Con-3), Exp_3, Nassoc_3(Nass-3), Ncut_3, Conduct-X(Con-X), Conduct_2(Con-2), Exp_2, Nassoc_2(Nass-2). In general the cut criteria Nassoc_3 is the best criteria. The correlations between criteria and errors are that triangle-based criteria can lead to a low number of errors and depend on the properties of networks.

code released by the authors of MSC,[5] HOSVD,[6] HOSPLOC,[7] and TSC via multilinear PageRank.[8] 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 code for MOSC.[9]

[5] `https://github.com/arbenson/higher-order-organization-matlab`
[6] `http://sml.csa.iisc.ernet.in/SML/code/Feb16TensorTraceMax.zip`
[7] `http://www.public.asu.edu/~dzhou23/Code/HOSPLOC.zip`
[8] `https://github.com/dgleich/mlpagerank`
[9] `https://bitbucket.org/Yan_Sheffield/mosc/`

4.2. Effectiveness of Mixed-order Cut Criteria

Firstly, we study the effect of mixed-order cut criteria on clustering performance. We have seven existing cut criteria from Table 1 and the proposed mixed-order conductance (ϕ_X). We study their effect on SC-Shi, SC-Ng, MSC, HOSVD, MOSC-GL($\lambda = 0.5$) and MOSC-RW ($\lambda = 0.5$) on DBLP, Orkut and LJ w.r.t ϵ_T, ϵ_E, ϵ_N. Note that we omit experimental results about ϵ_E on DBLP, Orkut and LJ since they show the same scenarios with ϵ_T on these three datasets. From Fig. 3, we have the following observations:

1. Mixed-order cut criteria have a greater impact on networks with dense triangles (e.g. Orkut and LJ) than networks with the sparse triangles (e.g. DBLP). The reason is that one error node in a dense triangle network, in comparison to in a sparse network, is normally shared by more error triangles.
2. Some optimised cut criteria do not truly reflect the quality of output communities w.r.t ϵ_T when comparing with ground-truth communities. In particular, in DBLP the optimised Nassoc_2 of SC-Shi still has great quantity of error triangles ϵ_T when comparing with some other optimised cut criteria (e.g. Ncut_3).
3. Mixed-order cut criteria can improve the performance of SC-Shi and SC-Ng that are conventional SC. In general, the cut criteria Nassoc_3 is the best criteria. It consistently gives the lowest number of error triangle ϵ_T and edge ϵ_E over three datasets than other seven cut criteria except for ϵ_E of MOSC-RW on Orkut. For error nodes, Nassoc_3 has the lowest number on LJ and second lowest on DBLP and Orkut.
4. Triangle-based criteria can lead to the less errors than edge-based criteria. In particular, Nassoc_3 can achieve a very low number of errors. In general edge-based cut criteria (e.g. Nassoc_2) do not show such low number of errors. From Table 1, Nassoc_3 is a only one that considers the number of triangles within communities. Therefore, the maximisation of the number of triangle within communities is effective to reduce errors in output

communities.

5. The correlation between criteria and errors depends on the properties of datasets. For LJ with small networks and dense triangles, the criterion Nassoc_3 can significantly reduce errors in terms of ϵ_N, ϵ_E and ϵ_T when comparing with other criteria. By contrast, for DBLP with big networks and sparse triangles, the performance of criterion Nassoc_3 is weakened so that the performance of criteria Ncut_3 is closed to it in terms of ϵ_E and ϵ_T.

Based on the above observations, we recommend the criterion Nassoc_3 to achieve a low number of error edges and triangles. We recommend Ncut_3 to reduce the number of error nodes for big and sparse networks.

4.3. Performance Comparison

We study the results of all algorithms in combination of all eight criteria and k-means (KM). Fig. 3 shows that cut criteria can affect the performance of all algorithms. Therefore, for fair comparison, we report the clustering results conducted by the best criteria for each algorithm. The top two results are in bold (best) or underlined (second best).

Results on SNAP Networks. We show the performance of all clustering algorithms with the best cut criteria in terms of NMI, ϵ_N, ϵ_E, and ϵ_T on SNAP networks in Table 3. The results for some settings of TSC and HOSPLOC are not available either due to long running time (not finished within 40 hours) or out of memory. In particular, the multilinear PageRank algorithm in TSC is very expensive.

We have four observations:

1. MOSC-RW ($\lambda = 0.5$) achieves the best in 10 out of 16 settings.
2. MOSC-RW outperforms MOSC-GL, although MOSC-GL achieves top two results in 4 settings. We will give a detailed discussion about it at Sec. 4.4.
3. Both MOSC-RW($\lambda = 0.5$) and MOSC-GL($\lambda = 0.5$) have better results than MOSC-RW(Auto-λ) and MOSC-GL(Auto-λ). This demonstrates

Table 3: Performance of clustering algorithms with the best cut criteria on SNAP networks. 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.

		Second order			Third order					MOSC-RW		MOSC-GL	
	Method	SC-Shi	SC-Ng	DeepWalk	HOSVD	MSC	TSC	HOSPLOC	STSC	$\lambda = 0.5$	Auto-λ	$\lambda = 0.5$	Auto-λ
DBLP	NMI	0.650	**0.656**	0.614	0.550	0.620	0.648	0.286	0.628	0.654	0.646	0.648	0.645
	ϵ_N	**4.13**	4.56	4.99	5.40	4.65	4.30	17.28	4.24	4.43	4.76	4.82	4.87
	ϵ_E	**13.36**	14.58	17.46	17.99	15.98	18.72	70.49	18.72	14.27	15.68	15.87	16.67
	ϵ_T	**24.81**	26.84	35.15	32.18	29.57	37.93	236.65	45.46	29.23	28.46	30.30	32.01
YouTube	NMI	0.248	0.270	0.258	0.124	0.184	-	-	0.150	**0.284**	0.260	0.275	0.263
	ϵ_N	22.48	23.31	26.63	25.69	24.41	-	-	23.83	**22.18**	23.46	23.44	23.83
	ϵ_E	44.42	46.46	61.43	50.61	47.18	-	-	63.12	**44.28**	46.74	52.58	47.96
	ϵ_T	**27.70**	29.49	47.36	30.78	29.1	-	-	59.78	28.90	29.29	38.63	29.46
Orkut	NMI	0.397	0.397	0.399	0.3618	0.390	-	-	0.387	**0.410**	0.393	0.397	0.394
	ϵ_N	37.13	37.09	38.06	40.43	38.49	-	-	37.60	**36.05**	37.02	36.72	36.93
	ϵ_E	574.6	574.6	635.5	624.7	582.3	-	-	569.2	**521.6**	571.8	550.4	574.9
	ϵ_T	4557	4557	5703	4937	4575	-	-	5104	**3949**	4541	4405	4614
LJ	NMI	0.226	0.224	0.156	0.218	0.224	0.214	-	0.201	**0.229**	0.221	0.208	0.212
	ϵ_N	5.58	5.63	23.08	5.79	5.74	5.52	-	**5.15**	5.49	5.66	5.76	5.64
	ϵ_E	49.83	50.01	1134	55.09	52.54	58.19	-	57.06	**47.88**	51.01	58.64	54.17
	ϵ_T	546.1	547.6	3404	600.6	574.5	737.7	-	773.0	**530.6**	556.4	730.3	617.8

that a fixed mixing parameter is effective, but it also shows the automatic schemes are not effective in these settings.

4. Mixed-order evaluation metrics can give insights on the quality of structure preservation for single-order SC. Specifically, SC-Shi preserves the most number of nodes, edges and triangles in DBLP than others. Additionally, although SC-Shi does not preserve the most number of nodes in YouTube, it still can preserve the most number of triangles than others.

Results on Full Networks. We show the performance of all clustering algorithms with the best cut criteria for five full networks in terms of NMI, ϵ_N, ϵ_E, and ϵ_T in Table 4, except HOSPLOC, for which we were not able to obtain comparable results. We have four observations:

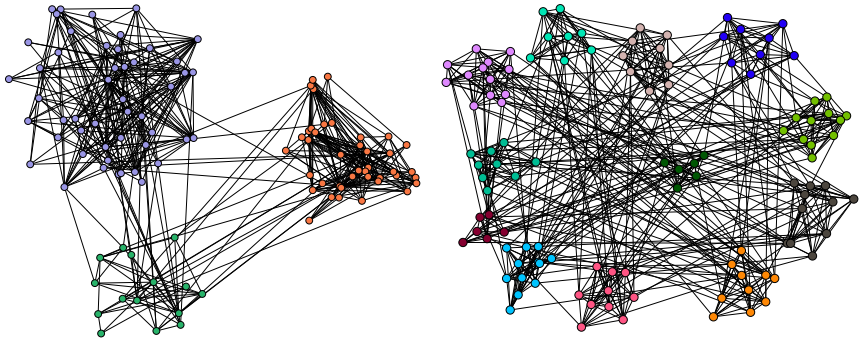
1. MOSC-GL (Auto-λ) achieves the best performance in 17 out of 24 settings, demonstrating that automatic determination of λ is effective in these settings. Specifically, in Dolphin MOSC-GL (Auto-λ) produces perfect results in all metrics. For networks with multiple clusters (Polbooks, Football), MOSC-GL (Auto-λ) is also superior to others. We visualise output clusters of Polbooks and Football in Fig. 4(a) and 4(b) respectively.

Table 4: 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.

		Second order			Third order				MOSC-RW		MOSC-GL	
	Method	SC-Shi	SC-Ng	DeepWalk	HOSVD	MSC	TSC	STSC	$\lambda = 0.5$	Auto-λ	$\lambda = 0.5$	Auto-λ
Zachary	NMI	**0.837**	**0.837**	0.732	0.069	0.732	0.677	0.325	**0.837**	**0.837**	**0.837**	**0.837**
	ϵ_N	**1**	**1**	2	14	2	2	8	**1**	**1**	**1**	**1**
	ϵ_E	**2**	**2**	7	34	3	3	24	**2**	**2**	**2**	**2**
	ϵ_T	**1**	1	10	16	**1**	**1**	14	**1**	**1**	**1**	**1**
Football	NMI	0.883	0.904	0.529	0.896	0.924	0.866	0.862	0.924	0.924	0.9	**0.931**
	ϵ_N	23	15	65	16	10	26	26	10	10	15	**9**
	ϵ_E	63	37	291	36	**7**	70	72	**7**	**7**	36	**7**
	ϵ_T	99	50	584	39	**2**	110	114	**2**	**2**	39	**2**
Polbooks	NMI	0.575	0.542	**0.615**	0.092	0.542	0.180	0.103	0.575	0.575	0.563	0.589
	ϵ_N	**17**	18	**17**	56	18	55	51	**17**	**17**	**17**	**17**
	ϵ_E	27	33	39	185	34	281	172	27	27	28	**21**
	ϵ_T	7	10	19	234	8	384	227	7	7	7	**1**
Dolphin	NMI	0.889	0.889	0.889	0.081	0.536	0.582	0.631	0.889	0.889	0.889	**1**
	ϵ_N	1	1	1	19	7	6	5	1	1	1	**0**
	ϵ_E	1	1	1	43	10	8	6	1	1	1	**0**
	ϵ_T	**0**	**0**	**0**	29	**0**	1	**0**	**0**	**0**	**0**	**0**
PBlogs	NMI	0.007	0.007	**0.740**	0.014	0.023	-	0.430	0.012	0.458	0.098	0.016
	ϵ_N	671	732	**54**	677	614	-	204	659	230	478	647
	ϵ_E	7,302	7,302	**159**	7,307	7,260	-	362	7,302	184	7,302	7,301
	ϵ_T	36,401	36,402	**423**	36,400	36,400	-	631	36,401	456	36,402	36,400
Facebook	NMI	0.023	0.255	0.163	0.177	0.055	-	0.0228	0.023	0.139	0.032	**0.258**
	ϵ_N	15,553	11,171	13,062	12,234	14,206	-	15,636	15,553	12,955	15,517	**11,166**
	ϵ_E	69,789	**57,630**	64,195	60,853	69,889	-	81,438	69,789	67,973	69,740	57,635
	ϵ_T	244,499	190,424	220,356	**182,274**	243,998	-	381,685	244,499	244,498	244,487	190,335

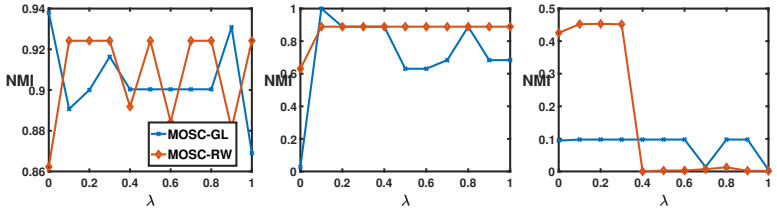
2. MOSC-GL (Auto-λ) outperforms MOSC-RW (Auto-λ), although MOSC-RW (Auto-λ) achieves the best results in 10 settings, which is still better than all existing SC algorithms (Note that there are ties).
3. Mixed-order evaluation metrics can gain insights on the quality of structure preservation. In Football MSC achieves the best ϵ_E and ϵ_T but not for ϵ_N, which also indicates existing mis-clustered node cannot reflect errors in structures.
4. In Facebook, MOSC-GL (Auto-λ) achieves the best performance w.r.t NMI and ϵ_N, and achieves the second best w.r.t ϵ_E and ϵ_T. Also, the results of SC-Ng are closed to the results of MOSC-GL (Auto-λ) since MOSC-GL(Auto-λ) is the generalisation of SC-Ng.

In Table 3, our auto-learning strategy that is based on cut criteria do not show superior performance for bi-partitioning and dense networks extracted from large networks. The reason is that optimised cut criteria do not truly



(a) Polbooks (b) Football (12 clusters)

Figure 4: Clusters in Polbooks and Football networks discovered by MOSC-GL (Auto-λ).



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

Figure 5: Sensitivity analysis of λ on Football, Dolphin and PBlogs w.r.t NMI.

reflect the quality of output communities w.r.t NMI, ϵ_N, ϵ_E, ϵ_T when comparing with ground-truth communities. However, in Table 4, our auto-learning strategy that is based on triangle density is effective to multi-cluster networks. Triangle density of communities can better reflect the quality of communities.

4.4. Sensitivity Analysis on 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. 5 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 of developing schemes to automatically determine the best λ. For PBlogs, Table 4 shows that MOSC-RW achieves significantly better performance than the

others. From Fig. 5(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.

From performance comparison in Section 4.3 and the above sensitivity analysis, 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. This can be the reason that MOSC-GL is less sensitive to tuning λ, or finding the appropriate λ is more difficult. That is, $\mathbf{W}_X$ tends to encode much less edge information. 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 linear combination. Therefore, MOSC-RW has a better performance than MOSC-GL in SNAP networks and the dense full graph PBlogs. Furthermore, based on the above discussion, we can give an explanation that MOSC-RW is quite sensitive to λ on PBlogs. In PBlogs, triangle information is likely to dominate the mixed-order structure and thus tuning λ may not change its mixture proportion largely especially for the PBlogs with dense triangles.

4.5. Computational Time

Fig. 6 compares the computational time of different methods on YouTube, LJ, PBlogs and Football, using k-means to obtain the final clusters to avoid the effect of cut criteria. We have the following three observations: 1) Both HOSVD and MOSC-RW involve tensor construction and operations so they are both more time consuming, in particular on dense networks such as LJ and PBlogs, where HOSVD is the slowest and MOSC-RW is the second slowest. The reason is that HOSVD uses a more complicated dimension reduction method

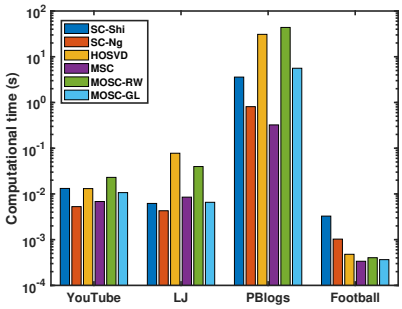


Figure 6: Computational time (in log scale) on YouTube, LJ, PBlogs and Football.

than MOSC-RW. 2) MOSC-GL is more efficient than MOSC-RW in all cases and has similar efficiency as conventional SC methods on the whole. 3) SC-Shi and SC-Ng are slower than MOSC-RW and MOSC-GL on Football since they 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.

4.6. Superpixel Segmentation

A superpixel is a group of similar pixels in colour or other low-level properties [39]. Superpixel segmentation as a preprocessing technique is increasingly popular in many computer vision tasks such as object tracking [40]. The main merit of superpixel is to provide a more natural and perceptually meaningful representation of the input image [41]. Therefore, compared with the traditional pixel representation of images, the superpixel representation greatly reduces the number of image primitive and improves the representative efficiency [41].

We perform experiments on the test set of Berkeley Segmentation Dataset[10] including two hundred images (size 481 $\times$ 321, equivalent to 154,401 nodes in networks) with human-labelled ground-truth segmentations. We compare MOSC-GL ($\lambda = 0.5$) with four algorithms: 1) simple linear iterative clustering (SLIC) [42]. It is a state-of-the-art method that is specifically designed to

[10] `https://www2.eecs.Berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`

Table 5: The best is in **bold** and the second best is underlined. MOSC-GL uses $\lambda = 0.5$. Comparing with LSC, MOSC-GL achieves competitive results especially for ASA.

	SLIC	SC-Shi	SC-Ng	MSC	DeepWalk	LSC	MOSC-GL
UE ↓	0.123	0.256	0.112	0.110	0.697	**0.092**±0.046	0.108±0.046
ASA ↑	0.867	0.767	0.880	0.882	0.438	**0.894**±0.045	0.883±0.043

superpixel segmentation task, 2) SC-Shi, 3) SC-Ng, 4) MSC, 5) linear spectral clustering (LSC)[43]. Results of SLIC are generated by the official Matlab implementation[11]. For LSC, we use the source code from the author with default settings[12]. We exclude the comparison with tensor-based method, e.g. MOSC-RW, HOSVD and TSC due to their large space consumption (still out of memory using 100G memory to construct the adjacency tensor). For all compared algorithms, the total number of superpixel is set to 100 for each image.

In order to obtain an objective and intuitive comparison, we quantitatively evaluate our algorithm by two popular metrics that are 1) undersegmentation error (UE) [41], 2) achievable segmentation accuracy (ASA) [44]. To compute the above metrics, we use $\mathbb{C} = \{C_j\}_{j=1}^{K}$ to denote the K segmentations of a ground-truth image, and $\mathbb{T} = \{T_i\}_{i=1}^{L}$ to represent the L segmentations by the superpixel algorithm.

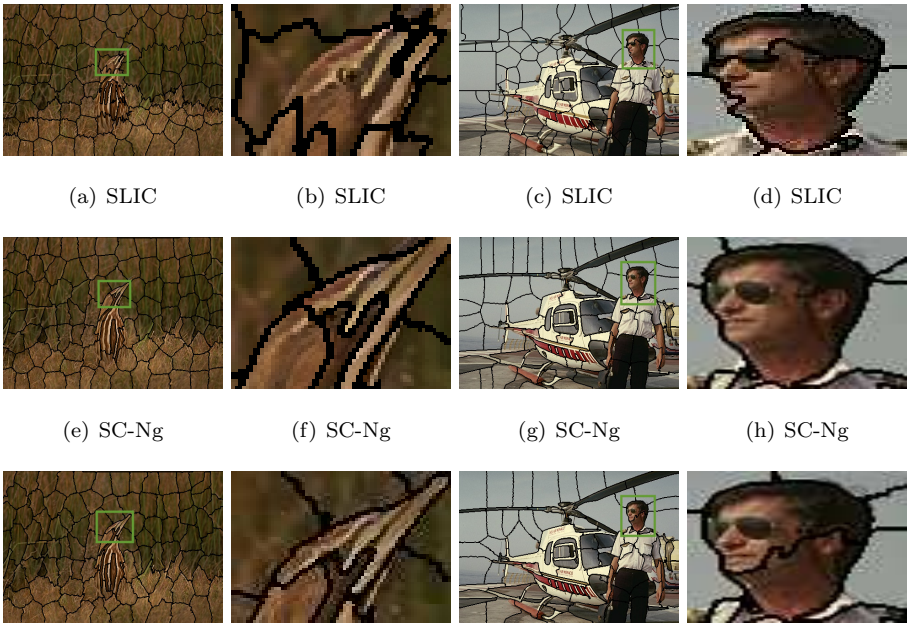
4.7. Performance Comparison

For quantitative evaluation, from Table 5, we observe that comparing with LSC, MOSC-GL achieves competitive results especially for ASA. Although our MOSC-GL is not motivated to handle superpixel segmentation, superpixel segmentation still turns out to be an application that is worthy to be applied. Furthermore, LSC is not applicable to the community detection task in networks since it cannot form networks.

For qualitative evaluation, Fig. 7 shows two example segmentations generated by MOSC ($\lambda = 0.5$), SC-Ng and SLIC. We observe that the superpixel

[11] https://www.mathworks.com/help/images/ref/superpixels.html

[12] https://github.com/neuwangmeng/Linear-Spectral-Clustering-Superpixel-Segmentation-Algorithm_Python



(a) SLIC (b) SLIC (c) SLIC (d) SLIC

(e) SC-Ng (f) SC-Ng (g) SC-Ng (h) SC-Ng

(i) MOSC (λ=0.5) (j) MOSC (λ=0.5) (k) MOSC (λ=0.5) (l) MOSC (λ=0.5)

Figure 7: Visual comparison among SLIC, SC-Ng and MOSC (λ=0.5). We observe that superpixels obtained by MOSC ($\lambda = 0.5$) can adhere well to boundaries of the head of a bird, and also can fit to the edges between a man's neck and head. SLIC generates the irregular and sharp superpixels.

boundaries obtained by proposed MOSC ($\lambda = 0.5$) can fit the object edges better than others. For example, MOSC ($\lambda = 0.5$) can adhere well to boundaries of the head of a bird, and also can fit to the edges between a man's neck and head. Additionally, compared with graph-based methods, superpixels obtained by SLIC are irregular and sharp.

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 simultaneously. 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. Moreover, we designed mixed-order cut criteria to enable existing single-order SC to preserve mixed-order structures, and new mixed-order evaluation metrics for structure evaluation. Experiments on community detection and superpixel segmentation tasks show that MOSC algorithms outperform existing SC methods in most cases and the proposed mixed-order approach has produced superior clustering of networks and superpixel segmentations of images. The future work can be developed for mixing more than two orders, and/or orders higher than three.

Acknowledgement

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

References

[1] A. R. Benson, D. F. Gleich, J. Leskovec, Higher-order organization of complex networks, Science 353 (6295) (2016) 163–166.

[2] A. Reihanian, M.-R. Feizi-Derakhshi, H. S. Aghdasi, Overlapping community detection in rating-based social networks through analyzing topics, ratings and links, Pattern Recognit 81 (2018) 370–387.

[3] H. Chang, Z. Feng, Z. Ren, Community detection using dual-representation chemical reaction optimization, IEEE Trans. Cybern. 47 (12) (2017) 4328–4341.

[4] F. Tung, A. Wong, D. A. Clausi, Enabling scalable spectral clustering for image segmentation, Pattern Recognit 43 (12) (2010) 4069–4076.

[5] Y. Kim, H. Do, S. B. Kim, Outer-points shaver: Robust graph-based clustering via node cutting, Pattern Recognit 97 (2020) 107001.

[6] J. V. Muñoz, M. A. Gonçalves, Z. Dias, R. d. S. Torres, Hierarchical clustering-based graphs for large scale approximate nearest neighbor search, Pattern Recognit 96 (2019) 106970.

[7] M. Meila, J. Shi, Learning segmentation by random walks, in: NeurIPS, 2001, pp. 873–879.

[8] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: NeurIPS, 2002, pp. 849–856.

[9] A. R. Benson, D. F. Gleich, J. Leskovec, Tensor spectral clustering for partitioning higher-order network structures, in: Proceedings of the 2015 SIAM International Conference on Data Mining, 2015, pp. 118–126.

[10] D. Zhou, S. Zhang, M. Y. Yildirim, S. Alcorn, H. Tong, H. Davulcu, J. He, A local algorithm for structure-preserving graph cut, in: KDD, ACM, 2017, pp. 655–664.

[11] O. Sporns, R. Kötter, Motifs in brain networks, PLoS Biology 2 (11) (2004) e369.

[12] M. S. Granovetter, The strength of weak ties, in: Social Networks, Elsevier, 1977, pp. 347–367.

[13] B. Serrour, A. Arenas, S. Gómez, Detecting communities of triangles in complex networks using spectral optimization, Computer Communications 34 (5) (2011) 629–634.

[14] D. Ghoshdastidar, A. Dukkipati, Consistency of spectral partitioning of uniform hypergraphs under planted partition model, in: NeurIPS, 2014, pp. 397–405.

[15] D. Ghoshdastidar, A. Dukkipati, Uniform hypergraph partitioning: Provable tensor methods and sampling techniques, JMLR 18 (50) (2017) 1–41.

[16] C. E. Tsourakakis, J. Pachocki, M. Mitzenmacher, Scalable motif-aware graph clustering, in: WWW, ACM, 2017, pp. 1451–1460.

[17] H. Yin, A. R. Benson, J. Leskovec, D. F. Gleich, Local higher-order graph clustering, in: KDD, ACM, 2017, pp. 555–564.

[18] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.

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

[20] S. E. Schaeffer, Graph clustering, Computer Science Review 1 (1) (2007) 27–64.

[21] P. Li, O. Milenkovic, Inhomogoenous hypergraph clustering with applications, in: NeurIPS, 2017, pp. 2305–2315.

[22] G. W. Flake, R. E. Tarjan, K. Tsioutsiouliklis, Graph clustering and minimum cut trees, Internet Mathematics 1 (4) (2004) 385–408.

[23] M. Fiedler, Algebraic connectivity of graphs, Czechoslovak Mathematical Journal 23 (2) (1973) 298–305.

[24] A. Abboud, V. V. Williams, Popular conjectures imply strong lower bounds for dynamic problems, in: FOCS, IEEE, 2014, pp. 434–443.

[25] J. Xu, T. L. Wickramarathne, N. V. Chawla, Representing higher-order dependencies in networks, Science Advances 2 (5) (2016) e1600028.

[26] J. R. Lee, S. O. Gharan, L. Trevisan, Multiway spectral partitioning and higher-order cheeger inequalities, Journal of the ACM (JACM) 61 (6) (2014) 1–30.

[27] J. Leskovec, K. J. Lang, M. Mahoney, Empirical comparison of algorithms for network community detection, in: WWW, ACM, 2010, pp. 631–640.

[28] T. Chakraborty, A. Dalmia, A. Mukherjee, N. Ganguly, Metrics for community analysis: A survey, ACM Computing Surveys (CSUR) (2017) 1–37.

[29] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowledge and Information Systems (2015) 181–213.

[30] L. Ana, A. K. Jain, Robust data clustering, in: CVPR, IEEE, 2003, pp. 128–136.

[31] W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropological Research 33 (4) (1977) 452–473.

[32] D. Lusseau, The emergent properties of a dolphin social network, Proceedings of the Royal Society of London B: Biological Sciences 270 (Suppl 2) (2003) S186–S188.

[33] M. E. Newman, Modularity and community structure in networks, Proceedings of the National Academy of Sciences 103 (23) (2006) 8577–8582.

[34] L. A. Adamic, N. Glance, The political blogosphere and the 2004 us election: divided they blog, in: Proceedings of the 3rd International Workshop on Link discovery, ACM, 2005, pp. 36–43.

[35] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding (2019). `arXiv:1909.13021`.

[36] D. F. Gleich, L.-H. Lim, Y. Yu, Multilinear pagerank, SIAM Journal on Matrix Analysis and Applications 36 (4) (2015) 1507–1541.

[37] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: KDD, 2014, pp. 701–710.

[38] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: CIKM, 2015, pp. 891–900.

[39] D. Stutz, A. Hermans, B. Leibe, Superpixels: An evaluation of the state-of-the-art, Computer Vision and Image Understanding 166 (2018) 1–27.

[40] G. Wu, W. Kang, Exploiting superpixel and hybrid hash for kernel-based visual tracking, Pattern Recognit 68 (2017) 175–190.

[41] J. Shen, Y. Du, W. Wang, X. Li, Lazy random walks for superpixel segmentation, IEEE Trans. Image Processing 23 (4) (2014) 1451–1462.

[42] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, Slic superpixels compared to state-of-the-art superpixel methods, IEEE Trans. Pattern Analysis and Machine Intelligence 34 (11) (2012) 2274–2282.

[43] J. Chen, Z. Li, B. Huang, Linear spectral clustering superpixel, IEEE Trans. Image Processing 26 (7) (2017) 3317–3330.

[44] M.-Y. Liu, O. Tuzel, S. Ramalingam, R. Chellappa, Entropy rate superpixel segmentation, in: CVPR, IEEE, 2011, pp. 2097–2104.