High-Order Structure Exploration on Massive Graphs: A Local Graph Clustering Perspective

DAWEI ZHOU and SI ZHANG, University of Illinois at Urbana-Champaign
MEHMET YIGIT YILDIRIM, Emailage Corp.
SCOTT ALCORN, Early Warnings LLC
HANGHANG TONG, University of Illinois at Urbana-Champaign
HASAN DAVULCU, Arizona State University
JINGRUI HE, University of Illinois at Urbana-Champaign

Modeling and exploring high-order connectivity patterns, also called network motifs, are essential for understanding the fundamental structures that control and mediate the behavior of many complex systems. For example, in social networks, triangles have been proven to play the fundamental role in understanding social network communities; in online transaction networks, detecting directed looped transactions helps identify money laundering activities; in personally identifiable information networks, the star-shaped structures may correspond to a set of synthetic identities. Despite the ubiquity of such high-order structures, many existing graph clustering methods are either not designed for the high-order connectivity patterns, or suffer from the prohibitive computational cost when modeling high-order structures in the large-scale networks. This article generalizes the challenges in multiple dimensions. First (*Model*), we introduce the notion of high-order conductance, and define the high-order diffusion core, which is based on a high-order random walk induced by the *user-specified high-order* network structure. Second (*Algorithm*), we propose a novel high-order *structure-preserving* graph clustering framework named *HOSGRAP*, which partitions the graph into *structure-rich* clusters in polylogarithmic time with respect to the number of edges in the graph. Third (*Generalization*), we generalize our proposed algorithm to solve the real-world problems on various types of graphs, such as signed graphs, bipartite graphs, and multi-partite graphs. Experimental results on both synthetic and real graphs demonstrate the effectiveness and efficiency of the proposed algorithms.

CCS Concepts: • **Mathematics of computing → Graph algorithms**; • **Information systems → Clustering**; • **Computing methodologies →** *Cluster analysis*; *Motif discovery*;

Additional Key Words and Phrases: Local clustering algorithm, high-order network structure

Authors' addresses: D. Zhou and S. Zhang, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, Illinois, 61801-2302; emails: {dzhou21, sizhang2}@illinois.edu; M. Y. Yildirim, Emailage Corp, 25 S Arizona Pl #400, Chandler, Arizona, 85225; email: myigityildirim@gmail.com; S. Alcorn, Early Warnings LLC, 16552 N 90th St., Scottsdale, Arizona, 85260; email: scott.alcorn@earlywarning.com; H. Tong, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, Illinois, 61801-2302; email: htong@illinois.edu; H. Davulcu, Arizona State University; email: hdavulcu@asu.edu; J. He, University of Illinois at Urbana-Champaign, 501 E Daniel St, Champaign, Illinois, 61820-6211; email: jingrui@illinois.edu.

1556-4681/2020/12-ART18 $15.00
https://doi.org/10.1145/3425637

ACM Reference format:
Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2020. High-Order Structure Exploration on Massive Graphs: A Local Graph Clustering Perspective. *ACM Trans. Knowl. Discov. Data* 15, 2, Article 18 (December 2020), 26 pages.
https://doi.org/10.1145/3425637

1 INTRODUCTION

Graph analysis has gained in popularity in the past decade, due to the increasing prominence of network data in a variety of real-world applications, from social networks to collaboration networks, from biological systems to e-commerce systems. Graph clustering algorithms represent an important family of tools for studying the underlying structure of networks. While most existing graph clustering algorithms are inherently limited to lower-order connectivity patterns [12, 32, 42], i.e., vertices and edges. They fail to explore the higher-order network structures, which are of key importance in many high-impact domains. For example, triangles have been proven to play the fundamental roles in understanding community structures [29]; a multi-hop loop structure may indicate the existence of money laundering activities in financial networks [19]; a star-shaped structure may correspond to a set of synthetic identities in personally identifiable information (PII) networks of bank customers [28].

Despite its importance, a key challenge associated with finding structure-rich subgraphs is the prohibitive computational cost. Many existing works on high-order graph clustering are either based on spectral graph theory [7, 48], or estimating the frequency of the high-order connectivity patterns [2, 11]. These methods may not be scalable to large-scale networks especially when modeling various complex network structures, such as loop-shaped structures, star-shaped structures and cliques. In this article, we aim to answer the following open questions. First (*Q1. Model*), it is not clear that how to model various types of high-order connectivity patterns (e.g., triangles, loops, and stars) that exist in the given graphs. Some motif-based graph clustering algorithms [7, 34, 47] have been proposed recently, while they are mainly designed for the 3$^{\text{rd}}$-order network structures (e.g., triangle). Second (*Q2. Algorithm*), how should we design a fast graph clustering algorithm that produces *structure-preserving* graph partitions in the massive real-world networks? This question has been largely overlooked in the previous studies. Third (*Q3. Generalization*), how can we generalize our algorithm to solve real-world problems on various types of graphs such as signed graphs, bipartite graphs, and multi-partite graphs?

To address above challenges, we propose a novel high-order *structure-preserving* graph clustering framework named *HOSGRAP*, which partitions the graph into *structure-rich* clusters in polylogarithmic time with respect to the number of edges in the graph. In particular, we start with a generic definition of high-order conductance, and define the high-order diffusion core, which is based on a high-order random walk induced by *user-specified high-order* network structure. Then, inspired by the family of local graph clustering algorithms [4, 5, 45] for efficiently identifying a low-conductance cut without exploring the entire graph, we generalize the key idea to high-order network structures and propose our fast high-order graph clustering framework *HOSGRAP*, which runs in polylogarithmic time with respect to the number of edges in the graph. It starts with a seed vertex and iteratively conducts high-order random walks [25, 34] to explore its neighborhood until a subgraph with a small high-order conductance is found. Our algorithm operates on the tensor representation of graph data which allows the users to specify what kind of network structures to be preserved in the returned cluster. In addition, we provide analyses regarding the effectiveness and efficiency of the proposed algorithm. Furthermore, we generalize our proposed *HOSGRAP* algorithm to the scenarios when the given networks are signed networks, bipartite networks and multi-partite networks. At last, we perform extensive experiments to demonstrate the

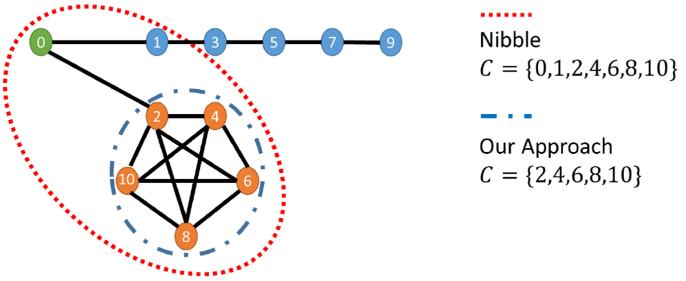


Fig. 1. A synthetic network where vertex 0 is connected with two kinds of network structures: clique and line. The local clusters found by our approach (within the blue dash-dot line) and the Nibble algorithm [45] (within the red dotted line) with the same initial vertex, i.e., vertex 0, where algorithm is conducted on the basis of three-node line (illustrated in Table 2).

effectiveness and the efficiency of the proposed methods. Figure 1 compares the clusters returned by our method and the Nibble algorithm [45], which shows that our method is better at partitioning a subgraph with the rich *user-specified high-order* network structure.

The main contributions of the article are summarized below.

(1) **Problem.** We formally define the problems of *Structure-Preserving Local Graph Cut* as well as *Structure-Preserving Graph Clustering*, and identify their unique challenges arising from real applications.
(2) **Algorithms and Analysis.** We propose a family of algorithms, i.e., *HOSPLOC* and *HOSGRAP*, to effectively identify structure-rich clusters with a *polylogarithmic* time complexity. Theoretical analyses show that our proposed algorithms can capture near-optimal structure-rich clusters under mild conditions.
(3) **Generalization and Application.** We generalize our proposed *HOSPLOC* and *HOSGRAP* algorithms from binary graphs to signed networks, bipartite networks, and multi-partite networks in real applications.
(4) **Evaluation.** Extensive experimental results on synthetic and real networks demonstrate the performance of the proposed *HOSPLOC* and *HOSGRAP* algorithms in terms of effectiveness, scalability, and parameter sensitivity.

The rest of our article is organized as follows. A brief overview of related literature is presented in Section 2, followed by the introduction of notation and preliminaries in Section 3. In Section 4, we present the proposed *HOSGRAP* algorithm as well as the analyses regarding its effectiveness and efficiency. Then, we introduce its generalizations and applications in Section 5. Experimental results are presented in Section 6 before we conclude the article in Section 7.

2 RELATED WORK

2.1 Local Spectral Clustering on Graphs

Nowadays, large-scale networks data appear in a broad spectrum of disciplines, from social networks [33, 36] to collaborative networks [16, 35], from rare category detection [3, 51–56] to community detection[13–15, 31, 50], and from data augmentation [9, 58, 59] to crowd-sourcing [60, 61]. Local spectral clustering techniques provide a simple, efficient time alternative to recursively identify a local sparse cut C with an upper-bounded conductance. In [45], the authors introduce an almost-linear Laplacian linear solver and a local clustering algorithm, i.e., Nibble, which conducts cuts that can be combined with balanced partitions. In [4, 5], the authors extend Nibble algorithm

[45] by using personalized PageRank vector to produce cuts with less running time on undirected and directed graphs. More recently, [24] proposes a local graph clustering algorithm with the same guarantee as the Cheeger inequalities, of which time complexity is slightly super linear in the size of the partition. In [6], the authors introduce randomized local partitioning algorithms that find sparse cuts by simulating the volume-biased evolving set process. To model the high-order connectivity patterns, [57] proposes a local graph clustering algorithm named *HOSPLOC* that identifies the structure-rich clusters by exploring the high-order structures in the neighborhood of the initial vertex in the given graph. Meanwhile, [49] develops a motif-based local graph clustering algorithm that approximately finds clusters with the minimal motif conductance,a generalization of the conductance metric for network motifs. Later on, in [43], the authors approach the problem the problem of discovering user-guided clustering in heterogeneous information networks, by transcribe the high-order interaction signals (i.e., network motifs) based on a non-negative tensor factorization methods. More recently, researchers aim to generalize *HOSPLOC* to the dynamic setting and develop a series of algorithms to compute [20] and track [23] "structure-rich" clusters in temporal networks. However, to my best of knowledge, this article is the first local clustering framework that focuses on modeling high-order network structures and partitions the graph into structure-rich clusters in polylogarithmic time with respect to the number of edges in the graph.

2.2 High-order Markov Chain Models

The o^{th} order Markov chain S describes a stochastic process that satisfies [25]

$$\begin{aligned} & Pr(S_{t+1} = i_1 | S_t = i_2, \ldots, S_{t-o+1} = i_{o+1}, \ldots, S_1 = i_{t+1}) \\ = & Pr(S_{t+1} = i_1 | S_t = i_2, \ldots, S_{t-o+1} = i_{o+1}) \end{aligned} \tag{1}$$

where $i_1, \ldots, i_{t+1}$ denote the set of states associated with different time stamps. Specifically, this means the future state only depends on the past o states. There are many cases that one would like to model observed data as a high-order Markov chain in different real-world problems, such as airport travel flows [41], web browsing behavior [17] and wind turbine design [39]. To solve these problems, many previous works [1, 18, 39] approximate the limiting probability distribution of high-order Markov chain as a linear combination of transition probability matrix.

More recently, in [34], the authors introduce a rank-1 approximation of high-order Markov chain limiting distribution and propose a recursive algorithm to compute it. Later on, [25] introduces a computationally tractable approximation of the high-order PageRank named multi-linear PageRank, where the underlying stochastic process is a vertex-reinforced random walk. In [8], the authors introduce a novel stochastic process, i.e., spacey random walk, whose stationary distribution is given by the tensor eigenvector, and show the convergence properties of these dynamics. In [7, 47], the authors propose the similar spectral clustering frameworks that allow for modeling third-order network structures and conduct partition while preserving such structures on the given graph. Followed by [7, 48] proposes a tensor spectral co-clustering method by modeling higher-order data with a novel variant of a higher-order Markov chain, i.e., the super-spacey random walk. Compared to the existing high-order Markov chain models, we propose a novel scalable local clustering algorithm that can identify clusters with a small conductance and also preserve the *user-specified high-order* network structures in a *polylogarithmic* time complexity. Moreover, we also provide provable theoretical bounds on the effectiveness and efficiency of the proposed high-order graph clustering framework.

3 PROBLEM DEFINITION

In this section, we formally define the structure-preserving graph cut and the structure-preserving graph clustering problems. Table 1 lists the main symbols used throughout this article. Given

Table 1. Symbols

Symbol	Definition and description
$G = (V, E)$	the undirected graph
A	the adjacency matrix of G
M	the transition probability matrix of G
$\mathbb{N}$	the k^{th}-order user-defined structure
$\boldsymbol{T}$	the k^{th}-order adjacency tensor of G
$\boldsymbol{P}$	the k^{th}-order transition tensor of G
$\bar{P}$	the unfolding matrix of the transition tensor $\boldsymbol{P}$
$q^{(t)}$	the t-step high-order random walk distribution vector
$r^{(t)}$	the truncated vector of $q^{(t)}$
C	the returned cluster
D	the returned graph partitions
$\mu(C)$	the edge volume of C
v	the initial vertex of local algorithm
$\Phi(C)$	the conductance of C
$\Phi(C, \mathbb{N})$	the k^{th}-order conductance of C regarding $\mathbb{N}$
k	the order of $\mathbb{N}$
m	the number edges of G
n	the number vertices of G
c	the number of graph partitions

an undirected graph $G = (V, E)$, where V consists of n vertices, and E consists of m edges, we let $A \in \mathbb{R}^{n \times n}$ denote the adjacency matrix of graph G, $D \in \mathbb{R}^{n \times n}$ denote the diagonal matrix of vertex degrees, and $d(v) = D(v, v)$ denote the degree of vertex $v \in V$. The transition matrix of a lazy random walk on graph G is $M = (A^T D^{-1} + I)/2$, where $I \in \mathbb{R}^{n \times n}$ is an identity matrix. For convenience, we define the indicator vector $\chi_C \in \{0, 1\}^n$ as follows.

$$\chi_C(v) = \begin{cases} 1 & v \in C \\ 0 & \text{Otherwise} \end{cases}.$$

In particular, the initial distribution of a random walk starting from vertex v could be denoted as χ_v.

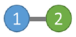
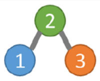
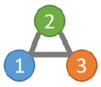
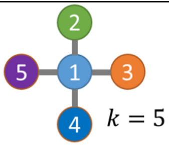
The volume of a subset $C \subseteq V$ is defined as the summation of vertex degrees in C, i.e., $\mu(C) = \sum_{v \in C} d(v)$. We let $\bar{C}$ be the complementary set of C, i.e., $\bar{C} = \{v \in \bar{C} | v \in V, v \notin C\}$. The conductance [10] of subset $C \subseteq V$ is therefore defined as

$$\Phi(C) = \frac{|E(C, \bar{C})|}{\min(\mu(C), \mu(\bar{C}))} \tag{2}$$

where $E(C, \bar{C}) = \{(u, v) | u \in C, v \in \bar{C}\}$, and $|E(C, \bar{C})|$ denotes the number of edges in $E(C, \bar{C})$. Besides, we represent the elements in a matrix or a tensor using the convention similar to Matlab, e.g., $M(i, j)$ is the element at the i^{th} row and j^{th} column of the matrix M, and $M(i, :)$ is the i^{th} row of M, and so on.

We let $\mathbb{N}$ denote the the k^{th}-order user-defined structure. Table 2 summarizes the examples of network structures $\mathbb{N}$ of different orders and the corresponding Markov chain. Notice that the order of the network structure is different from the order of the Markov chain (or random walk). For example, the edges in E are considered as the 2^{nd}-order network structures, and they correspond to the 1^{st}-order Markov Chain (random walk) due to the matrix representation of E. We use k to denote the order of the network structure $\mathbb{N}$. As what will be explained next, the k^{th}-order network structures correspond to the $(k-1)^{\text{th}}$-order Markov chain (random walk).

Table 2. Network Structures $\mathbb{N}$ and Markov Chains

$\mathbb{N}$	Illustration	Order of $\mathbb{N}$	Markov Chain	Random Walks	Graph Clustering Algorithms
Vertex		1st-order	0th-order	N/A	N/A
Edge		2nd-order	1st-order	1st-order	1st-order
3-node Line		3rd-order	2nd-order	2nd-order	2nd-order
Triangle					
k-node Star		kth-order	$(k-1)$th-order	$(k-1)$th-order	$(k-1)$th-order

With the above notion, our problems can be formally defined as follows:

Problem 1. *Structure-Preserving Local Graph Cut*

Input: *(i) an undirected graph $G = (V, E)$, (ii) a user-defined network structure $\mathbb{N}$, (iii) the initial vertex v.*
Output: *a local cluster C that largely preserves the user-defined structures $\mathbb{N}$.*

Problem 2. *Structure-Preserving Graph Clustering*

Input: *(i) an undirected graph $G = (V, E)$, (ii) a user-defined network structure $\mathbb{N}$, (iii) the number of clusters.*
Output: *a graph partition D that largely preserves the user-defined structures $\mathbb{N}$ in the returned clusters.*

4 PROPOSED ALGORITHMS

In the previous section, we introduced the notations and problem definitions. Now, we generalize the idea of truncated local clustering to produce clusters that preserve the *user-specified high-order* network structures. We start by reviewing the basics of the Nibble algorithm for local clustering on graphs [45], which pave the way for discussion of the proposed structure-preserving graph clustering algorithm. Then, we introduce the adjacency tensor and the associated transition tensor based on the *user-specified high-order* network structures, followed by the discussion on the stationary distribution of high-order random walk. After that, we introduce the definitions of high-order conductance and high-order diffusion core. Finally, we present our proposed *HOSPLOC* and *HOSGRAP* algorithms with theoretical analyses in terms of the effectiveness and efficiency.

4.1 Background: Truncated Local Graph Clustering Algorithm

Given an undirected graph G and a parameter $\phi > 0$, to find a cluster C from G such that $\Phi(C) \leq \phi$ or to determine no such C exists is an NP-complete problem [44]. Nibble algorithm [45] is one of the earliest attempts to partition a graph with a bounded conductance in polylogarithmic time. Starting from a given vertex, Nibble provably finds a local cluster in time $(O(2^b log^6 m)/\phi^4))$, where

b is a constant which controls the lower bound of the output volume. This is proportional to the size of the output cluster. The key idea behind Nibble is to conduct truncated random walks by using the following truncation operator

$$[q]_\epsilon(u) = \begin{cases} q(u) & \text{if } q(u) \geq d(u)\epsilon \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

where $q \in \mathbb{R}^n$ is the distribution vector over all the vertices in the graph, and ϵ is the truncation threshold that can be computed as follows [45]

$$\epsilon = \frac{1}{(1800 \cdot (l+2)t_{last}2^b)} \tag{4}$$

where l can be computed as $l = \lceil log_2(\mu(V)/2) \rceil$, and t_{last} can be computed as

$$t_{last} = (l+1)\left\lceil \frac{2}{\phi^2} ln\left(c_1(l+2)\sqrt{\mu(V)/2}\right)\right\rceil \tag{5}$$

Then, Nibble applies the vector-based partition method [37, 38, 45] that sorts the probable nodes based on the ratio of function I_x to produce a low conductance cut. To introduce function I_x mathematically, we first define $S_j(q)$ to be the set of top j vertices u that maximizes $q(u)/d(u)$. That is $S_j(q) = \{\pi(1), \ldots, \pi(j)\}$, where π is the permutation that follows

$$\frac{q(\pi(i))}{d(\pi(i))} \geq \frac{q(\pi(i+1))}{d(\pi(i+1))}.$$

In addition, we let $\lambda_j(q) = \sum_{u \in S_j(q)} d(u)$ denote the volume of the set $S_j(q)$. Finally, the function I_x is defined as follows

$$I_x(q, \lambda_j(q)) = \frac{q(\pi(j))}{d(\pi(j))}. \tag{6}$$

4.2 Adjacency Tensor and Transition Tensor

For an undirected graph G, the corresponding adjacency matrix A could be considered as a matrix representation of the existing edges on G. If each vertex in graph G corresponds to a distinct state, we can interpret the transition matrix M as the transition matrix of the 1st-order Markov chain. Specifically, the transition probability between vertex i and vertex j is given by $M(i,j) = Pr(S_{t+1} = i | S_t = j)$. Moreover, if M is stochastic, irreducible and aperiodic [40], we can compute a positive and unique vector $\bar{x} = M\bar{x}$, where $\bar{x} \in \mathbb{R}^n$ is the limiting or stationary probability distribution of the random walk.

However, in many real applications, we may want to explore and capture more complex and high-order network structures. To model the *user-specified* network structure $\mathbb{N}$, we introduce the definition of adjacency tensor $\boldsymbol{T}$ and the transition tensor $\boldsymbol{P}$ to represent the high-order random walk induced by the high-order network structures $\mathbb{N}$.

Definition 1 (Adjacency Tensor). Given a graph $G = (V, E)$, the k^{th}-order network structure $\mathbb{N}$ on G could be represented in a k-dimensional adjacency tensor $\boldsymbol{T}$ as follows

$$T(i_1, i_2, \ldots, i_k) = \begin{cases} 1 & \{i_1, i_2, \ldots, i_k\} \subseteq V \text{ and form } \mathbb{N}. \\ 0 & \text{Otherwise.} \end{cases} \tag{7}$$

Definition 2 (Transition Tensor). Given a graph $G = (V, E)$ and the adjacency tensor $\boldsymbol{T}$ for the k^{th}-order network structure $\mathbb{N}$, the corresponding transition tensor $\boldsymbol{P}$ could be computed as

$$P(i_1, i_2, \ldots, i_k) = \frac{T(i_1, i_2, \ldots, i_k)}{\sum_{i_1=1}^{n} T(i_1, i_2, \ldots, i_k)} \tag{8}$$

By the above definition, we have $\sum_{i_1} P(i_1, \dots, i_k) = 1$. Therefore, if each vertex in G is a distinguishable state, we can interpret the k^{th}-order transition tensor $\boldsymbol{P}$ as a $(k-1)^{\text{th}}$-order Markov chain (random walk), i.e.,

$$Pr(S_{t+1} = i_1 | S_t = i_2, \dots, S_{t-k+2} = i_k) = P(i_1, \dots, i_k).$$

Intuitively, if $i_1 \neq i_1'$, and they both form $\mathbb{N}$ together with $i_2, \dots, i_k$, then the probabilities of the next state being i_1 and being i_1' are the same given $S_t = i_2, \dots, S_{t-k+2} = i_k$. Notice that the transition matrix M of a lazy random walk defined in Subsection 4.1 can be considered as a special case of Definition 2 with the 2nd-order network structure $\mathbb{N}$, if we allow self-loops.

4.3 Stationary Distribution

For the k^{th}-order network structure $\mathbb{N}$ and the corresponding $(k-1)^{\text{th}}$-order random walk with transition tensor $\boldsymbol{P}$, if the stationary distribution $\boldsymbol{X}$ exists, where $\boldsymbol{X}$ is a $(k-1)$-dimensional tensor, then it satisfies [25]

$$X(i_1, i_2, \dots, i_{k-1}) = \sum_{i_k} P(i_1, i_2, \dots, i_k) X(i_2, \dots, i_k). \tag{9}$$

Where $X(i_1, \dots, i_{k-1})$ denotes the probability of being at states $i_1, \dots, i_{k-1}$ in consecutive time steps upon convergence of the random walk, and $\sum_{i_1, \dots, i_{k-1}} X(i_1, \dots, i_{k-1}) = 1$.

However, for this system, storing the stationary distribution requires $O(n^{(k-1)})$ space complexity. For the sake of computational scalability, in high-order random walks, a commonly held assumption is ‘rank-one approximation’ [7, 34], i.e.,

$$X(i_2, \dots, i_k) = q(i_2) \dots q(i_k) \tag{10}$$

where $q \in \mathbb{R}_+^{n \times 1}$ with $\sum_i q(i) = 1$. Then, we have

$$\sum_{i_2, \dots, i_k} P(i_1, \dots, i_k) q(i_2) \dots q(i_k) = q(i_1).$$

In this way, the space complexity of the stationary distribution of high-order random walk is reduced to $O(n)$. Although q is an approximation of the true stationary distribution of the high-order random walk, [34] theoretically demonstrates the convergence and effectiveness of the nonnegative vector q if $\boldsymbol{P}$ satisfies certain properties.

Following [7, 34], in this article, we also adopt “rank-one approximation” and assume the stationary distribution of the high-order random walk satisfies Equation (10). To further simplify the notation, we let $\bar{P}$ denote the $(k-2)$-mode unfolding matrix of the k-dimensional transition tensor P. Thus, the $(k-1)^{\text{th}}$-order random walk satisfies:

$$q = \bar{P}(q \otimes \dots \otimes q) \tag{11}$$

where $\otimes$ denotes the Kronecker product symbol. For example, for the third-order network structure $\mathbb{N}$ (e.g., triangle), the transition tensor $\boldsymbol{P} \in \mathbb{R}^{n \times n \times n}$ can be constructed based on Definition 2. Then, the 1-mode unfolding matrix $\bar{P}$ of $\boldsymbol{P}$ can be written as follows

$$\bar{P} = [P(:,:,1), P(:,:,2), \dots, P(:,:,n)]$$

where $\bar{P} \in \mathbb{R}^{n \times n^2}$. In this way, the associated second-order random walk with respect to the triangle network structure satisfies

$$q = \bar{P}(q \otimes q).$$

4.4 High-Order Conductance

Given a high-order network structure $\mathbb{N}$, it is usually the case that the user would like to find a local cluster C on the graph G such that: (1) C contains a rich set of network structures $\mathbb{N}$; and (2) by partitioning all the vertices into C and $\bar{C}$, we do not break many such network structures. For example, in financial fraud detection, directed loops may refer to money laundering activities. In this case, we would like to ensure the partition preserves rich directed loops inside the cluster and breaks such structure as less as possible. It is easy to see that the traditional definition of the conductance $\Phi(C)$ introduced in Section 4.1 does not serve this purpose. Therefore, we introduce the following generalized definition of conductance to preserve user-defined high-order network structure $\mathbb{N}$.

Definition 3 (k^{th}-order Conductance). For any cluster C in graph G and the k^{th}-order network structure $\mathbb{N}$, the k^{th}-order conductance $\Phi(C, \mathbb{N})$ is defined as

$$\Phi(C,\mathbb{N}) = \frac{cut(C,\mathbb{N})}{min\{\mu(C,\mathbb{N}), \mu(\bar{C},\mathbb{N})\}} \tag{12}$$

where $cut(C, \mathbb{N})$ denotes the number of network structures broken due to the partition of G into C and $\bar{C}$, i.e.,

$$\begin{aligned} cut(C,\mathbb{N}) = &\sum_{i_1,\ldots,i_k \in V} T(i_1,\ldots,i_k) - \sum_{i_1 i_2,\ldots,i_k \in C} T(i_1,\ldots,i_k) \\ &- \sum_{i_1,\ldots,i_k \in \bar{C}} T(i_1,\ldots,i_k) \end{aligned} \tag{13}$$

and $\mu(C, \mathbb{N})$ ($\mu(\bar{C}, \mathbb{N})$) denotes the total number of network structures $\mathbb{N}$ incident to the vertices within C ($\bar{C}$), i.e.,

$$\begin{aligned} \mu(C,\mathbb{N}) &= \sum_{i_1 \in C; i_2,\ldots,i_k \in V} T(i_1,i_2,\ldots,i_k) \\ \mu(\bar{C},\mathbb{N}) &= \sum_{i_1 \in \bar{C}; i_2,\ldots,i_k \in V} T(i_1,i_2,\ldots,i_k). \end{aligned} \tag{14}$$

Claim 1. *Definition 3 provides a generic definition of network conductance with respect to any network structure, and it subsumes existing measures of network conductance. In particular.*

- *When $\mathbb{N}$ represents edges, $\Phi(C, \mathbb{N})$ is twice the traditional conductance $\Phi(C)$ introduced in Section 4.1.*
- *When $\mathbb{N}$ represents triangles, $\Phi(C, \mathbb{N})$ is the same as the "high-order conductance" ϕ_3 introduced in [7].*

4.5 High-Order Diffusion Core

Similar to the Nibble algorithm, we are given a seed vertex v, and our goal is to find a cluster C containing or near v without looking at the whole graph. The main advantage of our proposed work is that, given the *user-specified high-order* network structure $\mathbb{N}$, we are able to produce a local cluster that preserves such structure within the cluster C and does not break many such structures by partitioning the graph into C and $\bar{C}$.

To this end, we perform high-order random walk with transition tensor $\boldsymbol{P}$ defined in Definition 2, starting from the seed vertex v. Let $q^{(t)}$ denote the distribution vector over all the vertices after the t^{th} iteration of the high-order random walk. Ideally, a seed vertex chosen within a cluster C with low conductance should lead to the discovery of this cluster. However, as pointed out in [45], for the 2nd-order network structure and the associated 1st-order random walk, if the vertices within the cluster are more strongly attached to vertices outside the cluster than inside it, they may not be good candidates for the seed, as the random walk will have a relatively high chance of escaping the cluster after a few iterations. Therefore, they propose the definition of the diffusion core to characterize the subset of vertices within the cluster, such that the random walks starting from such vertices stay inside the cluster for a long time. Here, we generalize the definition of a diffusion core to high-order network structures as follows.

Definition 4 (k^{th}-Order ξ-Diffusion Core). For any cluster C, we define $C^{k,\xi} \in C$ to be the k^{th}-order ξ-diffusion core of C, such that

$$\chi^T_{\bar{C}^{k,\xi}} q^{(t)} \le \xi \frac{cut(C, \mathbb{N})}{\mu(C, \mathbb{N})} \tag{15}$$

where $q^{(t)}$ denotes the diffusion distribution of t-step high-order random walks, and ξ is a positive constant that controls the compactness of the diffusion core.

Note that the left-hand side of Equation (15), $\chi^T_{\bar{C}^{k,\xi}} q^{(t)}$, represents the probability that a high-order random walk terminates outside the cluster C after t steps, which is also called the escaping probability of the cluster C. On the right-hand side of Equation (15), the numerator could be considered as the total number of the k^{th}-order random walk paths to escape cluster C, while the denominator could be regarded as the total number of the k^{th}-order random walk paths starting from C. It is easy to see that $\chi^T_{\bar{C}^{k,\xi}} q^{(t)}$ is positively correlated with $\frac{cut(C,\mathbb{N})}{\mu(C,\mathbb{N})}$. Since, for a given C, $\chi^T_{\bar{C}^{k,\xi}} q^{(t)}$ is a computable constant, we consider Equation (15) as the compactness constraint for the k^{th}-order ξ-diffusion core $C^{k,\xi} \in C$.

Proposition 1. *For any cluster C and the k^{th}-Order ξ-diffusion core $C^{k,\xi} \in C$, we have*

$$\chi^T_{\bar{C}^{k,\xi}} q^{(t)} \le \xi \Phi(C, \mathbb{N}). \tag{16}$$

Proof. Given a cluster $C \in V$ and a k^{th}-order network structure $\mathbb{N}$, the corresponding k^{th}-order conductance can be computed as

$$\Phi(C, \mathbb{N}) = \frac{cut(C, \mathbb{N})}{min\{\mu(C, \mathbb{N}), \mu(\bar{C}, \mathbb{N})\}}.$$

Obviously, we can divide the proof into the following two cases. Case 1 : when $\mu(C, \mathbb{N}) \ge \mu(\bar{C}, \mathbb{N})$, $\Phi(C, \mathbb{N}) = \frac{cut(C,\mathbb{N})}{\mu(\bar{C},\mathbb{N})} \ge \frac{cut(C,\mathbb{N})}{\mu(C,\mathbb{N})}$.

Case 2 : when $\mu(C, \mathbb{N}) < \mu(\bar{C}, \mathbb{N})$, $\Phi(C, \mathbb{N}) = \frac{cut(C,\mathbb{N})}{\mu(C,\mathbb{N})}$.

Thus, we have $\Phi(C, \mathbb{N}) \ge \frac{cut(C,\mathbb{N})}{\mu(C,\mathbb{N})}$. Meanwhile, by Definition 4, it turns out that

$$\chi^T_{\bar{C}^{k,\xi}} q^{(t)} \le \xi \frac{cut(C, \mathbb{N})}{\mu(C, \mathbb{N})} \le \xi \Phi(C, \mathbb{N}). \qquad \square$$

4.6 High-Order Structure-Preserving Graph Cut

Basically, the proposed *HOSPLOC* could be decomposed into three main steps: (1) approximately compute the distribution of high-order random walk starting at any vertex from which the walk does not mix rapidly; (2) truncate all small entries in $q^{(t)}$ to 0, thus we can limit the computation to the neighborhood of the seed; and (3) apply the vector-based graph partition method [37, 38, 45] to search for a *structure-rich* cut with a small conductance.

Now, we are ready to present our proposed *HOSPLOC* algorithm. The given inputs are the transition tensor $\boldsymbol{P}$, the transition matrix M, the seed vertex v, the conductance upper-bound ϕ, the maximum iteration number $t_{\max}$, and the constants b, c_1, ξ. Note that constant b controls the volume lower bound of the returned set C, i.e., $2^b \leq \mu(C)$, and c_1 is a constant which guarantees that the elements in C have a large probability of staying within C. Steps 1–4 are the initialization process. Step 1 constructs unfolding matrix $\bar{P}$ of the transition tensor $\boldsymbol{P}$. Steps 2–4 compute the truncation constant ϵ and the truncated initial distributions vectors $r^{(m)}$, $m = 1, \ldots, k-1$. The iterative process between Step 5 and Step 16 aims to identify the proper high-order local cluster C: Step 6 calculates the updated distribution over all the vertices in current iteration; Step 7 calculates the truncated local distribution $r^{(t)}$; the iterative process stops when it finds a proper cluster which satisfies the three constraints in Steps 9–11, where condition (a) guarantees that the conductance of C is upper-bounded by ϕ, condition (b) ensures that the volume of C is lower-bounded by 2^b, and condition (c) enforces that elements in C have a large probability mass.

ALGORITHM 1: High-Order Structure-Preserved Local Cut (*HOSPLOC*)

```
Input:
    (1) $k^{\text{th}}$-order transition tensor $\boldsymbol{P}$ and transition matrix $M$,
    (2) Initial vertex $v$,
    (3) Conductance upper bound $\phi$,
    (4) Maximum iteration number $t_{\max}$,
    (5) Parameters $b$, $c_1$, $\xi$.
Output:
    Local cluster $C$;
 1: Construct the unfolding matrix $\bar{P}$ of the transition tensor $\boldsymbol{P}$.
 2: Compute constant $\epsilon$ based on Equation (4).
 3: Set initial distribution vectors $q^{(t)} = M^{(t-1)}\chi_v$, where $t = 1, \ldots, k-1$.
 4: Compute truncated initial local distribution vectors $r^{(t)} = [q^{(t)}]_\epsilon$, $t = 1, \ldots, k-1$.
 5: for $t = k : t_{\max}$ do
 6:     Update distribution vector $q^{(t)} = \bar{P}(r^{(t-1)} \otimes \ldots \otimes r^{(t-k+1)})$.
 7:     Update truncated distribution vectors $r^{(t)} = [q^{(t)}]_\epsilon$.
 8:     if there exists a $j$ such that:
 9:     (a)$\Phi(S_j(q^{(t)})) <= \phi$,
10:     (b)$2^b <= \lambda_j(q^{(t)})$,
11:     (c)$I_x(q^{(t)}, 2^b) >= \frac{\xi}{c_1(l+2)2^b}$. then
12:         return $C = S_j(q^{(t)})$ and quit.
13:     else
14:         Return $C = \emptyset$.
15:     end if
16: end for
```

Next, we analyze the proposed *HOSPLOC* algorithm in terms of effectiveness and efficiency. Regarding the effectiveness, we will show that for any cluster C, if the seed vertex comes from the k^{th}-order ξ-diffusion core, i.e., $v \in C^{k,\xi}$, then the non-empty set C' returned by *HOSPLOC* has a large overlap with C. To be specific, we have the following theorem.

Theorem 1 (Effectiveness of HOSPLOC). *Let C be a cluster on graph G such that $\Phi(C, \mathbb{N}) \leq \frac{1}{c_2(l+2)}$, where $2c_1 \leq c_2$. If* HOSPLOC *runs with starting vertex $v \in C^{k,\xi}$ and returns a non-empty set C', then we have $\mu(C' \cap C) \geq 2^{b-1}$.*

Proof. Let $q^{(t)}$, $t \leq t_{\max}$, be the distribution of $t - step$ high-order random walk when the set $C' = S_j(q^{(t)})$ is obtained. Then, based on Proposition 1, we have the following inequality

$$\chi_{\bar{C}}^T q^{(t)} \leq \chi_{\bar{C}^{k,\xi}}^T q^{(t)} \leq \xi \Phi(C, \mathbb{N}) \leq \frac{\xi}{c_2(l+2)}. \tag{17}$$

In Step 11 of Algorithm 2, condition (c) guarantees that

$$I_x(u) = \frac{q^{(t)}(u)}{d(u)} \geq \frac{\xi}{c_1(l+2)2^b} \tag{18}$$

where $u \in S_j(q^{(t)})$. Since $d(u) \geq 0$ and $c_1(l+2)2^b \geq 0$, we can infer the following inequality from Equation (18)

$$d(u) \leq \frac{1}{\xi} c_1(l+2)2^b q^{(t)}(u). \tag{19}$$

Let j' be the smallest integer such that $\lambda_{j'}(q^{(t)}) \geq 2^b$. In Step 10 of Algorithm 1, condition (b) guarantees that $j' \leq j$. By Equations (17) and (19), we have

$$\begin{aligned} &\mu(S_{j'}(q^{(t)}) \cap \bar{C}) \\ &= \sum_{u \in S_{j'}(q^{(t)}) \cap \bar{C}} d(u) \\ &\leq \sum_{u \in S_{j'}(q^{(t)}) \cap \bar{C}} \frac{1}{\xi} c_1(l+2)2^b q^{(t)}(u) \\ &\leq \frac{1}{\xi} c_1(l+2)2^b (\chi_{\bar{C}}^T q^{(t)}) \\ &\leq \frac{\xi c_1(l+2)2^b}{\xi c_2(l+2)} \leq 2^{b-1}. \end{aligned} \tag{20}$$

Due to $2^b \leq \lambda_{j'}(q^{(t)})$, it turns out that $\mu(S_{j'}(q^{(t)}) \cap C) \geq 2^{b-1}$. Since $j \geq j'$, we have the final conclusion

$$\mu(S_j(q^{(t)}) \cap C) \geq \mu(S_{j'}(q^{(t)}) \cap C) \geq 2^{b-1}. \tag{21}$$

□

Regarding the efficiency of *HOSPLOC*, we provide the following lemma to show the *polylogarithmic* time complexity of *HOSPLOC* with respect to the number of edges in the graph.

Lemma 1 (Efficiency of HOSPLOC). *Given graph G and the k^{th}-order network structure $\mathbb{N}$, $k \geq 3$, the time complexity of* HOSPLOC *is bounded by $O(t_{max} \frac{2^{bk}}{\phi^{2k}} log^{3k} m)$.*

Proof. To bound the running time of *HOSPLOC*, we first show that each iteration in Algorithm 1 takes time $O(\frac{1}{\epsilon^k})$. Instead of conducting dense vector multiplication or Kronecker product, we track the nonzeros in both matrixes and vectors. Here, we let V^t denote the set of vertices such that $\{u \in V^{(t)} | r^{(t)}(u) > 0\}$, and $V^{(\hat{t})}$ be the set with the maximum number of nonzero elements in $\{V^{(t)} | 1 \leq t \leq t_{max}\}$. In Step 6, the Kronecker product chain $r^{(t-1)} \otimes \ldots \otimes r^{(t-k+1)}$ can be performed in time proportion to

$$|V^{(t-1)}| \ldots |V^{(t-k+1)}| \leq |V^{(\hat{t})}|^{(k-1)} \leq \mu(V^{(\hat{t})})^{(k-1)}.$$

Also, [45] shows that $\mu(V^{(t)}) \leq 1/\epsilon$ for all t. Therefore, to compute $r^{(t-1)} \otimes \ldots \otimes r^{(t-k+1)}$ takes $O(\mu(V^{(\hat{t})})^{(k-1)}) \leq O(1/\epsilon^{(k-1)})$ time. After that, the matrix vector product can be computed in

$$O\Big(\mu(V^{(t)}, \mathbb{N})\Big) \leq O\Big(\mu(V^{(\hat{t})}, \mathbb{N})\Big) \leq O\Big(\mu(V^{(\hat{t})})\Big)^k \leq O\left(\frac{1}{\epsilon^k}\right).$$

The truncation in Step 7 can be computed in time $O(|V^{(\hat{t})}|)$. Steps 8–15 require sorting the vertices in $|V^t|$ according to $r^{(t)}$, which takes time $O(|V^{(\hat{t})}| \log |V^{(\hat{t})}|)$. In sum, the time complexity of each iteration in *HOSPLOC* is $O(\frac{1}{\epsilon^k})$.

Since the algorithm runs at most t_{max} iterations, the overall time complexity of *HOSPLOC* is $O(\frac{t_{max}}{\epsilon^k})$. By Equation (4), we can expand $O(\frac{t_{\max}}{\epsilon^k})$ as follows

$$\begin{aligned} O\left(\frac{t_{\max}}{\epsilon^k}\right) &= O\left(t_{max}\left(\frac{2^b log^3 \mu(V)}{\phi^2}\right)^k\right) \\ &= O\left(t_{max}\frac{2^{bk}}{\phi^{2k}} log^{3k} m\right). \end{aligned}$$ □

Remark 1: The major computation overhead of Algorithm 1 comes from Step 6. Note that $O(t_{max}\frac{2^{bk}}{\phi^{2k}} log^{3k} m)$ is a strict upper-bound for considering extreme cases. While, due to the power law distribution in real networks, we may usually have $|V^{(t)}| \leq \sqrt{\mu(V^{(\hat{t})})}$. Then, the time complexity of Algorithm 1 can be reduced to $O(t_{max}/\epsilon^{k/2}) = O(t_{max}(2^b/\phi^2)^{k/2} log^{3k/2} m)$.

Remark 2: Suppose the maximum iteration number of Nibble and *HOSPLOC* are both upper-bounded by t_{max}, then the time complexity of Nibble is $O(\frac{t_{max} 2^b log^4 m}{\phi^2})$. Considering the $k = 3$ case, the time complexity of *HOSPLOC* is $O(t_{max}\frac{2^{3b}}{\phi^6} log^9 m)$. Without considering the impact from the other constants, we can see that similar to Nibble, *HOSPLOC* also runs in *polylogarithmic* time complexity with respect to the number of edges in the graph.

4.7 High-Order Structure-Preserving Graph Clustering

We now present the high-order structure-preserving graph clustering algorithm named *HOSGRAP* in Algorithm 2, that perform structure-preserving graph partitioning by routinely calling *HOSPLOC*. The inputs of Algorithm 2 are mostly the same as Algorithm 1, the only differences are that Algorithms 2 requires cluster number c and the vertices distribution ψ_V for sampling initial vertices in order to call *HOSPLOC*. Step 1 is the initialization step, while Steps 2–8 are the main loop of *HOSPLOC* that aims to partition the graph into c structure-rich subgraphs. Specifically, Steps 3 and 4 construct the subgraph $G^{(j)}$ and its corresponding transition tensor $\boldsymbol{P}^{(j)}$ by indexing G and $\boldsymbol{P}$; Step 5 samples the initial vertex from $G^{(j)}$ according to ψ_V, while Step 6 computes the value of b that controls the minimum volume of the returned cluster; in the end, Step 7 calls *HOSPLOC* to conduct graph cut by using the above computed parameters. If the returned cluster C in Step 6 is nonempty, we will update the partition $D = D \cup \{C, \bar{C}\}$, otherwise, we will return the current graph partition D. The algorithm stops when the graph is partitioned into c structure-rich subgraphs.

As *HOSGRAP* calls *HOSPLOC* via a subroutine, we analyze the complexity of Algorithm 2 based on the results from Lemma 1. Lemma 2 shows that the expected running time of *HOSGRAP* algorithm is bounded by $O(L\frac{t_{max} log^{3k+1} m}{\phi^{2k}})$, where L denotes the iteration number of Algorithm 2.

LEMMA 2 (EFFICIENCY OF HOSGRAP). *Given graph G and the k^{th}-order network structure $\mathbb{N}$, $k \geq 3$, the time complexity of* HOSGRAP *is bounded by* $O(L\frac{t_{max} log^{3k+1} m}{\phi^{2k}})$.

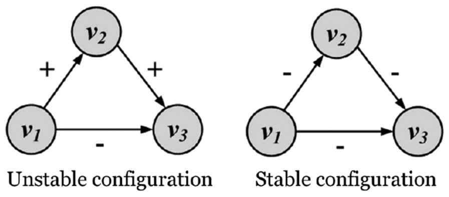


Fig. 2. Social status theory example: (Left) A directed "+" edge from node v_1 to node v_2 shows that v_2 has a higher status than v_1. (Right) A directed "-" edge from node v_1 to node v_2 shows vice versa.

Proof. Based on Lemma 1, the expected running time of inner loop (Steps 3–7) in Algorithm 2 can be bounded by

$$\begin{aligned} &O\left(\sum_{i=1}^{\lceil \log m^{(j)} \rceil} \frac{2^{-ik}}{1-2^{-\lceil \log m^{(j)} \rceil k}} t_{max} \frac{2^{ik}}{\phi^{2k}} log^{3k} m^{(j)}\right) \\ &\leq O\left(\sum_{i=1}^{\lceil \log m \rceil} \frac{1}{1-2^{-\lceil \log m \rceil k}} \frac{t_{max log^{3k} m}}{\phi^{2k}}\right) \\ &\leq O\left(t_{max} \frac{log^{3k+1} m}{\phi^{2k}}\right) \end{aligned} \tag{23}$$

where $m^{(j)}$ is the number of edges in the subgraph $G^{(j)}$.

Suppose the overall iterations of *HOSGRAP* is L, then the expected running time of *HOSGRAP* is upper bounded by $O(L\frac{t_{max} log^{3k+1} m}{\phi^{2k}})$. Note that the iteration number L is naturally larger than the number of clusters c. When L = c, it indicates the fact that *HOSGRAP* successfully identifies a cluster in each iteration before stopping the algorithm. □

5 GENERALIZATIONS AND APPLICATIONS

In this section, we introduce several generalizations and applications of our proposed *HOSPLOC* algorithm on signed networks, bipartite networks, and multi-partite networks.

5.1 Community Detection on Signed Networks

First, we extend our proposed framework, i.e., *HOSPLOC*, to solve problems on signed graphs. In many real applications, the high-order network structures of interest to us are presented with signed edges. For instance, Figure 2 presents an unstable three-node network structure and a stable three-node network structure based on social status theorem [27]. In community detection [22], we may want to ensure (1) the stable configurations to be rich within communities and sparse in-between different communities; (2) the unstable configurations to be sparse within communities and rich in-between different communities. For this purpose, the adjacency tensor can be constructed as follows

$$T(i_1, i_2, \ldots, i_k) = \begin{cases} 1 & \{i_1, i_2, \ldots, i_k\} \text{ is stable structure} \\ 0 & \{i_1, i_2, \ldots, i_k\} \text{ is unstable structure} \\ \alpha & \text{Otherwise} \end{cases} \tag{24}$$

where $\{i_1, i_2, \ldots, i_k\} \in V$ and constant $0 < \alpha < 1$. By this way, we can ensure: (1) the returned cluster of *HOSPLOC* contains rich stable structures; (2) the partition would most likely break unstable structures.

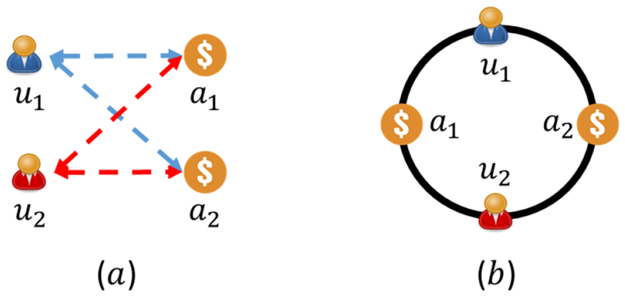


Fig. 3. The illustration of user-advertisement interaction. (a) An example of two users both participate in two advertisement campaigns. (b) An four-node loop induced from (a).

ALGORITHM 2: High-Order Structure-Preserved Graph Partitioning (*HOSGRAP*)

```
Input:
    (1) $k^{\text{th}}$-order transition tensor $\boldsymbol{P}$ and transition matrix $M$,
    (2) Vertex distribution $\psi_V$,
    (3) Conductance upper bound $\phi$,
    (4) Maximum iteration number $t_{\max}$,
    (5) Parameters $c_1$, $\xi$,
    (6) Partition number $c$
Output:
    Graph Partitioning $D = D_1 \cup \dots D_j$;
1: Set $G^{(1)} = G$, $\boldsymbol{P^{(1)}} = \boldsymbol{P}$, $M^{(1)} = M$ and $j = 1$.
2: while $j < c$ do
3:     Construct the subgraph $G^{(j)} = (V^{(j)}, E^{(j)})$ regarding the largest component in $D$.
4:     Compute the transition tensor $\boldsymbol{P^{(j)}}$ and the transition matrix $M^{(j)}$ of subgraph $G^{(j)}$ .
5:     Randomly sample a initial vertex in $G^{(j)}$ according to $\psi_V$.
6:     Choose a $b$ in $1, \dots, \lceil \log m \rceil$ according to
           $$Pr(b = i) = \frac{2^{-ki}}{1 - 2^{-k\lceil \log m \rceil}}.$$
7:     Partitioning $G^{(j)}$ into $C$ and $\bar{C}$ via HOSPLOC algorithm. If $C$ is nonempty, let $D = D \cup \{C, \bar{C}\}$,
       and $j = j + 1$; otherwise, return the current graph partitioning $D$.
8: end while
```

5.2 User Behavior Modeling on Bipartite Networks

We now turn our attention to the problem of user behavior modeling on the advertisement networks. Given an advertisement network $B = (V_B, E_B)$, the bipartite graph B contains two types of nodes, i.e., user nodes V_U and advertiser campaign nodes V_A, i.e., $V_B = \{V_U, V_A\}$. The edges E_B only exist between user nodes V_U and advertiser campaign nodes V_A. Intuitively, the customers with similar activities on the advertisement network should be included in the same cluster. For this reason, we choose four-node loop as the base network structure for *HOSPLOC* algorithm. Specifically, suppose both user nodes u_1, u_2 have user-campaign interactions with the advertiser campaign nodes a_1 and a_2, then we have a four-node loop, which is shown in Figure 3. In this problem, we consider the advertisement network as an undirected graph, and the adjacency tensor can be constructed as follows

$$T(i_1, i_2, i_3, i_4) = \begin{cases} 1 & \{i_1, i_2, i_3, i_4\} \text{ form a 4-nodes loop} \\ 0 & \text{Otherwise} \end{cases} \tag{25}$$

where $\{i_1, i_2, i_3, i_4\} \in V_B$. Starting from an initial vertex, the returned cluster C_B by *HOSPLOC* would represent a local user-campaign community, which consists of both similar users and the users' favorite advertiser campaigns.

5.3 Synthetic ID Detection on Multi-partite Networks

Here, we explain how to detect synthetic IDs on the PII network by using our proposed *HOSPLOC* algorithm. The PII network is a typical multi-partite network, where each partite set of nodes represents a particular type of PII, such as users' names, users' accounts, and email addresses, and the edges only exist between different partite sets of nodes. In synthetic ID fraud [28], criminals often use modified identity attributes, such as phone number, home address and email address, to combine with real users' information and create synthetic IDs to do malicious activities. Hence, for the synthetic IDs, there is a high possibility that their PIIs would be shared by multiple identities, which may compose rich star-shaped structures. In this case, the adjacency tensor can be constructed as

$$T(i_1, i_2, \ldots, i_k) = \begin{cases} 1 & \{i_1, i_2, \ldots, i_k\} \text{ form a } k\text{-node star} \\ 0 & \text{Otherwise} \end{cases} \tag{26}$$

where $\{i_1, i_2, \ldots, i_k\} \in V_B$. Note that the returned partition may consist of various types of nodes. However, it is viable to trace back from the extracted PII nodes and discover the set of synthetic identities.

6 EXPERIMENTAL RESULTS

In this section, we present the experimental evaluations. The experiments are designed to answer the following questions: In particular, we aim to answer the following questions:

— *Effectiveness*: How effective is the proposed *HOSPLOC* algorithm for conducting a local cut with preserving high-order network structures, and how effective is the proposed *HOSGRAP* algorithm for performing structure-preserving graph clustering?
— *Scalability*: How fast and scalable are the proposed *HOSPLOC* and *HOSGRAP* algorithms?
— *Parameter Sensitivity*: How robust are the proposed algorithms with changing parameters?
— *Case Study*: What's the performance of the proposed algorithms when we are solving problems on bipartite graph and multi-partite graph.

6.1 Experiment Setup

Datasets: We evaluate our proposed algorithm on both synthetic and real-world network graphs. The statistics of all real datasets are summarized in Table 2.

— *Collaboration Network:* We use two collaboration networks from Aminer.[1] In network (Author), the nodes are authors, and an edge only exists when two authors have a co-authored paper. In network (Paper), the nodes are distinct papers, and an edge only exists when one paper cites another paper.
— *Infrastructure Network:* In network (Airline),[2] the nodes represent 2,833 airports, and the edges represent the US flights in a 1-month interval. Network (Oregon) [30] is a network

[1] https://aminer.org/data.
[2] http://www.levmuchnik.net/Content/Networks/NetworkData.html.

Table 3. Statistics of the Networks

Category	Network	Type	Nodes	Edges
Citation	Author	Undirected	61,843	402,074
	Paper	Undirected	62,602	10,904
Infrastructure	Airline	Undirected	2,833	15,204
	Oregon	Undirected	7,352	15,665
	Power	Undirected	4,941	13,188
Social	Epinion	Undirected	75,879	508,837
Review	Rating	Bipartite	8,724	90,962
Financial	PII	Multi-partite	375	519

of routers in Autonomous Systems inferred from Oregon route-views between March 31, 2001 and May 26, 2001. Network (Power)[3] contains the information of the power grid of the western states of USA. A node represents a generator, a transformator or a substation, and an edge represents a power supply line.

— *Social Network:* Network (Epinion) [30] is a who-trust-whom online social network. Each node represents a user, and one edge exits if and only if when one user trusts another user.
— *Review Network:* Network (Rating) [26] is a bipartite graph, where one side of nodes represent 643 users, and another side of nodes represent 7,483 movies. Edges refer to the positive ratings, i.e., rating score larger than 2.5, on MovieLens website. Note that this network is a subgraph from the original one, due to storing the 4th-order transition tensor of the original graph, i.e., 100s K vertices and millions edges, requires too much memory.
— *Financial Network:* Network (PII) is a multi-partite graph, which consists of five types of vertices, i.e., 112 bank accounts, 71 names, 80 emails, 35 addresses, and 77 phone numbers. Edges only exist between account vertices and PII vertices.

Comparison Methods: In our experiments, we compare our methods with both local and global graph clustering methods. Specifically, the comparison algorithm includes three local algorithms, i.e., (1) Nibble [45], (2) NPR [4], and (3) LS-OQC [46], and two global clustering algorithms, i.e., (1) NMF [21] and (2) TSC [7]. Among these five baseline algorithms, TSC algorithm is designed based on high-order Markov chain, which can model high-order network structures, i.e., triangle.

Repeatability: Most of the datasets are publicly available. The code of the proposed algorithms is released on the authors' website. For all the results reported, we set $c_1 = 140$, $\xi = 1$, and vertex distribution ψ_V to be a uniform distribution. t_{last} can be directly computed from the given graph via Equation (5). The experiments are mainly performed on a Windows machine with four 3.5 GHz Intel Cores and 256 GB RAM.

6.2 Effectiveness Analysis

The effectiveness comparison results conducted on six real undirected graphs by the following three evaluation metrics. Among them, (1) *Conductance* [10] in Equation (2) measures the general quality of a cut on graph, which quantitatively indicates the compactness of a cut; (2) *The 3rd-Order Conductance* could be computed based on Equation (12) by treating triangle as the network structure $\mathbb{N}$, which estimates how well the network structure $\mathbb{N}$ is preserved in the returned cut from being broken by the partitions; and (3) *Triangle Density* [10] is defined as $\tau(C) = 3t(C)/w(C)$, where $t(G)$ is the number of triangles in C and $w(C)$ is the number of wedges in C. Conventionally,

[3]http://konect.uni-koblenz.de/networks/opsahl-powergrid.

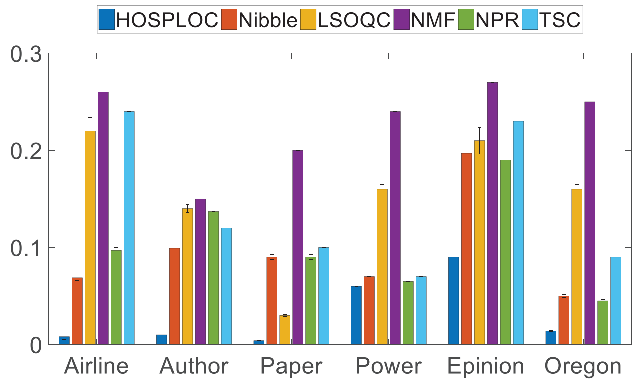


Fig. 4. The average conductance of the returned graph cut. Lower is better.

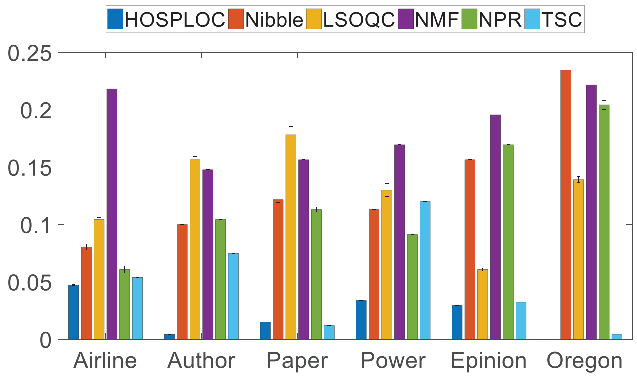


Fig. 5. The average 3rd-order conductance of the returned graph cut. Lower is better.

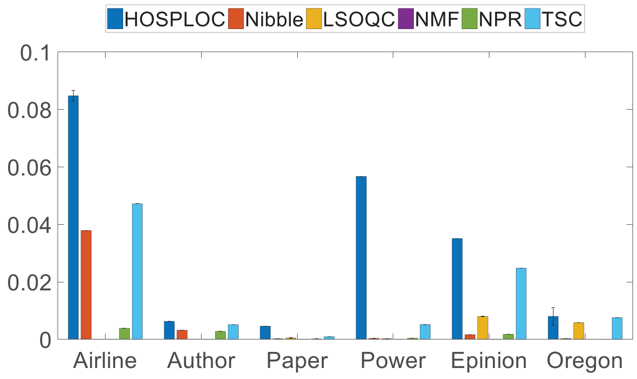


Fig. 6. The average triangle density of the returned graph cut. Higher is better.

we have $t(C) = 0$ if there is no wedge in the given C. Here we use Triangle Density to measure the ratio of how rich the triangle is included in the returned cluster C.

A. Quantitative Evaluations for Problem 1. The comparison results for the structure-preserving local graph cut problem are shown from Figure 4 to Figure 6. Moreover, to evaluate the convergence of local algorithms, we randomly select 30 vertices from one cluster on each testing

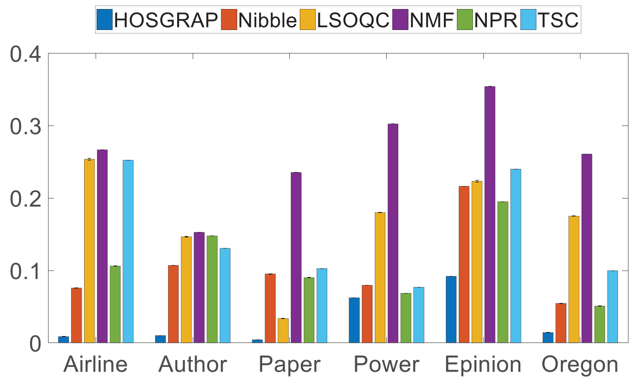


Fig. 7. The average conductance over the partitioned subgraphs. Lower is better.

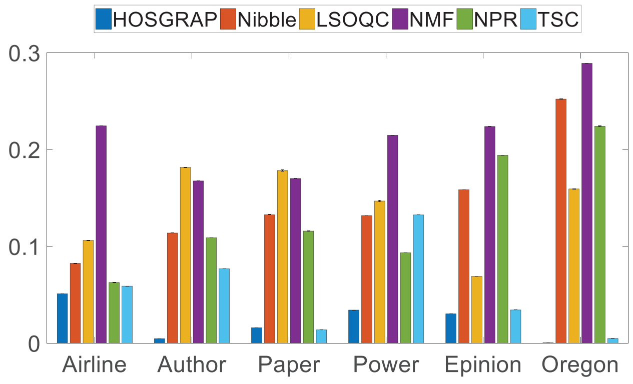


Fig. 8. The average 3rd-order conductance over the partitioned subgraphs. Lower is better.

graph and run all the local algorithms multiple times by treating each of these nodes as an initial vertex. In particular, the heights of bars indicate the average value of evaluation metrics, and the error bars (only for local algorithms) represent the standard deviation of evaluation metrics in multiple runs. We have the following observations: (1) In general, local algorithms perform better than the global algorithm, and our *HOSPLOC* algorithm consistently outperforms the others on all the evaluation metrics. For example, compared to the best competitor, i.e., TSC, on network (Airline), *HOSPLOC* algorithm is 97% smaller on conductance, 12.2% smaller on the 3rd-order conductance, 80% larger on triangle density. (2) High-order Markov chain models, i.e., *HOSPLOC* and TSC, could better preserve triangles in the returned cluster. For example, on network (Epinion), both *HOSPLOC* and TSC return a cluster with much higher triangle density and much lower the 3rd-order conductance. (3) *HOSPLOC* algorithm shows a more robust convergence property than the other local clustering algorithm by comparing the size of error bars. For example, among the three local algorithms, only *HOSPLOC* algorithm returns the identical cluster on network (Paper) with different initial vertexes.

B. Quantitative Evaluations for Problem 2. The comparison results for the Structure-Preserving Graph Partition problem are presented in Figures 7–9. For conducting *HOSGRAP*, we manually set the vertex distribution ψ_V following the degree distribution, and the partition number $c = 5$. In Figures 7–9, the height of the bars indicate the averaged value of the metrics of the partitioned subgraphs, and the error bars represent the standard deviation of evaluation metrics

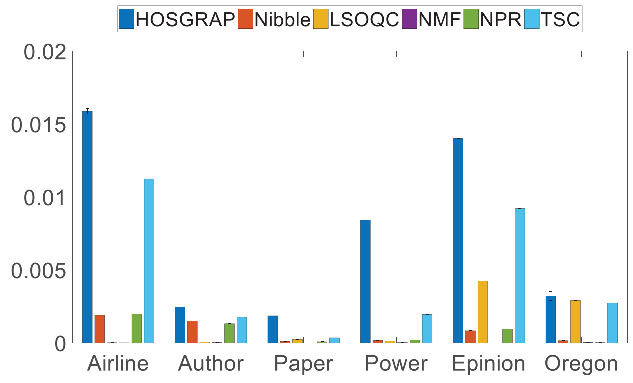


Fig. 9. The average triangle density over the partitioned subgraphs. Higher is better.

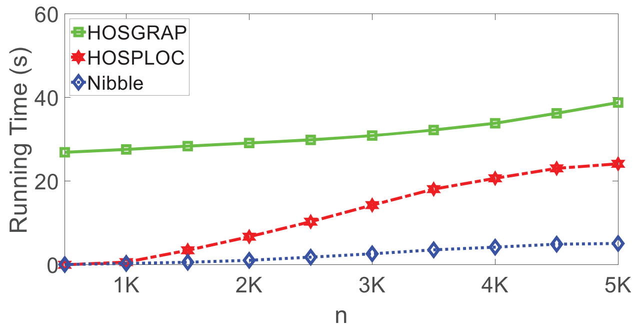
in 30-times runs. In general, we observe that our proposed *HOSGRAP* algorithm outperforms the baseline methods in all the six datasets across all the metrics.

6.3 Scalability Analysis

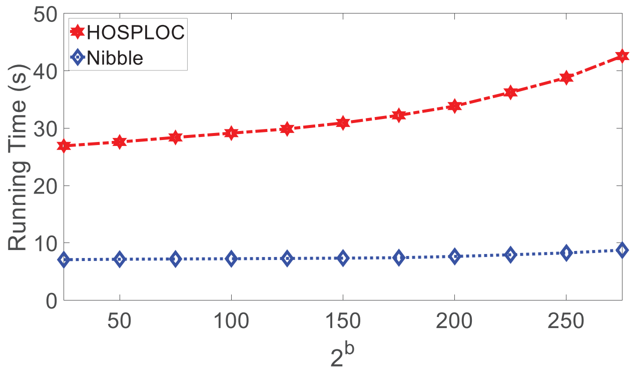
In this subsection, we study the scalability of our proposed framework. We let triangle as the user-defined network structure and the partition number $c = 3$. Since our method is built on higher order of random walk than Nibble, we consider Nibble as the running time lower bound of *HOSPLOC* algorithm. Notice that all the results in Figure 10 are the average values of multiple runs by using 30 different initial vertexes on the same graph. In Figure 10(a), we show the running time of *HOSGRAP*, *HOSPLOC*, and Nibble on a series of synthetic graphs with increasing number of vertices but fixed edge density of 0.5%. We observe that although *HOSGRAP* and *HOSPLOC* require more time than Nibble in each run, the running time of *HOSGRAP* and *HOSPLOC* increases *polylogarithmically* with the size of the graph $|V|$, which demonstrate our scalability analysis in Lemma 1 and Lemma 2. In Figure 10(b), we show the running time of *HOSPLOC* and Nibble versus the lower bound of output volume on the synthetic graph with 5,000 vertices and 0.5% edge density, by keeping the other parameters fixed. Note that *HOSGRAP* is not included in Figure 10(b), since the the lower bound of output volume 2^b is not an input variable of *HOSGRAP* algorithm. We can see that the running time of *HOSPLOC* is polynomial with respect to 2^b, which is consistent with our time complexity analysis.

6.4 Parameter Analysis

In this subsection, we analyze the parameter sensitivity of our proposed *HOSPLOC* algorithm with triangle as the specified network structure, by comparing with Nibble algorithm on the synthetic graph with 5,000 vertices and 0.5% edge density. Here, we mainly focused on *HOSPLOC* algorithm, as *HOSGRAP* could be considered as multiple runs of *HOSPLOC* algorithm with different initialization. In the experiments, we evaluate the conductance and the 3rd-order conductance of the returned cut with different values of input parameter ϕ. In Figure 11, we have the following observations: (1) *HOSPLOC* returns the optimal cut even with a very loose conductance upper bound ϕ. In Figure 11(a), we can see the output conductance of *HOSPLOC* converges to the minimum value when $\phi = 0.4$, while the output conductance of Nibble converges to its minimum value until $\phi = 0.1$. (2) Both the conductance and the 3rd-order conductance of *HOSPLOC*'s cut are always smaller than Nibble's cut with different ϕ.



(a) The number of vertices



(b) The lower bound of C's volume

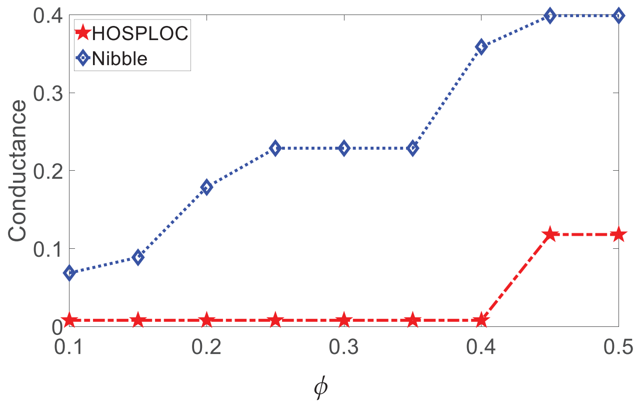
Fig. 10. Scalability Analysis w.r.t. the number of nodes n and the the lower bound volume of the returned clusters 2^b.

6.5 Case Study

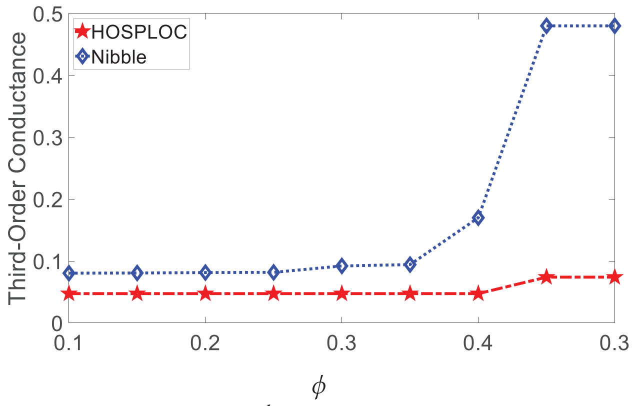
In this subsection, we will consider more complex network structures and perform our proposed *HOSPLOC* algorithm on bipartite and multi-partite networks.

Case Study on Bipartite Graph. We conduct a case study on the network (Rating) to find a local community consisting of similar taste users and their favorite movies. In this case study, we construct the transition tensor on the basis of four-node loop based on Equation (25). Figure 12(a) presents a miniature of the cluster identified by our proposed *HOSPLOC* algorithm regarding four-node loop that illustrated in Figure 12(b). For example, in Figure 12, the highlighted red loop shows that both of the third and the fourth users like the first and the fourth movies, while the highlighted blue loop represents that both of the third and the fifth users like the fifth and the last movies. It seems the fifth user does not like the first movie due to no direct connection between them. While the interesting part is the first, the fifth and the last movies are from the same series, i.e., Karate Kid I, II, III. Moreover, the fourth movie, i.e., Back to School, and Karate Kid I, II, III, are all from the category of comedy. It turns out that our *HOSPLOC* algorithm returns a community of comedy movies and their fans.

Case Study on Multi-partite Graph. Here, we conduct a case study on the network (PII) to identify suspicious systemic IDs. In this case, we treat five-node star as the underlying network structure, and the corresponding transition tensor could be generated by Equation (26). Figure 13(a) presents a subgraph of the returned cut by our proposed *HOSPLOC* algorithm regarding five-node star that illustrated in Figure 13(b). We can see that many PIIs are highly



(a) Conductance



(b) The 3rd-order conductance

Fig. 11. Parameter analysis w.r.t. conductance upper-bound ϕ. Lower is better.

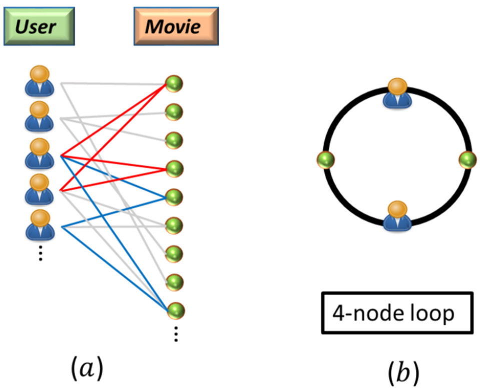


Fig. 12. Case study on bipartite network Rating. (a) An example of detected community by *HOSPLOC* on Rating. (b) An example of four-node loop on Rating.

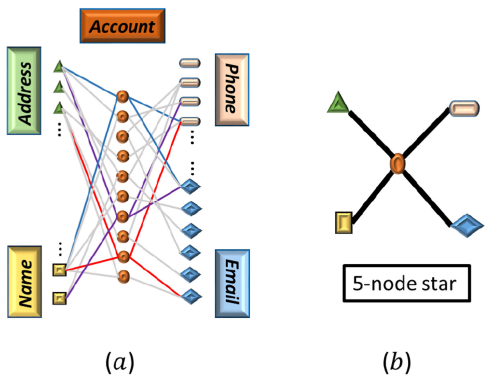


Fig. 13. Case study on multi-partite network PII. (a) An example of detected community by *HOSPLOC* on PII. (b) An example of five-node star on PII.

shared by different accounts. For example, the account connected with blue lines shares the home address and email address with the account connected with purple lines, while the account connected with red lines shares the holder's name and phone number with the account connected with blue lines. Comparing with the regular dense subgraph detection methods, our method can better identify the IDs who share their PIIs with others, by exploring the nature structure of PII, i.e., five-node star, on the given graph.

7 CONCLUSION

In this article, we propose a structure-preserving graph cut algorithm, i.e., *HOSPLOC*, that gives users the flexibility to model any high-order network structures and returns a small high-order conductance cluster which largely preserves the *user-specified* network structures. Based on *HOSPLOC*, we further develop a partitioning algorithm named *HOSGRAP* that largely preserves the user-defined structures in the returned clusters. Besides, we analyze its performance in terms of the optimality of the obtained cluster and the *polylogarithmic* time complexity on massive graphs. Furthermore, we generalize the proposed algorithms to solve real problems on signed networks, bipartite networks and multi-partite networks, by exploring the useful high-order network connectivity patterns, such as loops and stars. Finally, the extensive empirical evaluations on a diverse set of networks demonstrate the effectiveness and scalability of our proposed *HOSPLOC* and *HOSGRAP* algorithms.

REFERENCES

[1] S. R. Adke and S. R. Deshmukh. 1988. Limit distribution of a high order Markov chain. *Journal of the Royal Statistical Society: Series B (Methodological)* 50, 1 (1988), 105–108.

[2] Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick G. Duffield. 2015. Efficient graphlet counting for large networks. In *Proceedings of the 2015 IEEE International Conference on Data Mining*. Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu (Eds.). IEEE Computer Society, 1–10. DOI: https://doi.org/10.1109/ICDM.2015.141

[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery* 29, 3 (2015), 626–688.

[4] Reid Andersen, Fan R.K. Chung, and Kevin J. Lang. 2006. Local graph partitioning using PageRank vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE Computer Society, 475–486. DOI: https://doi.org/10.1109/FOCS.2006.44

[5] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2007. Local partitioning for directed graphs using PageRank. In *Proceedings of the 5th International Workshop on Algorithms and Models for the Web-Graph*. Anthony Bonato and Fan R. K. Chung (Eds.), Vol. 4863. Springer, 166–178. DOI: https://doi.org/10.1007/978-3-540-77004-6_13

[6] Reid Andersen, Shayan Oveis Gharan, Yuval Peres, and Luca Trevisan. 2016. Almost optimal local graph clustering using evolving sets. *Journal of the ACM* 63, 2 (2016), 15:1–15:31. DOI: https://doi.org/10.1145/2856030

[7] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2015. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. Suresh Venkatasubramanian and Jieping Ye (Eds.). SIAM, 118–126. DOI: https://doi.org/10.1137/1.9781611974010.14

[8] Austin R. Benson, David F. Gleich, and Lek-Heng Lim. 2017. The spacey random walk: A stochastic process for higher-order data. *SIAM Review* 59, 2 (2017), 321–345. DOI: https://doi.org/10.1137/16M1074023

[9] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. NetGAN: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning,*. Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 609–618. Retrieved from http://proceedings.mlr.press/v80/bojchevski18a.html.

[10] Béla Bollobás. 2002. *Modern Graph Theory*. Graduate Texts in Mathematics, Vol. 184. Springer. DOI: https://doi.org/10.1007/978-1-4612-0619-4

[11] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2017. Counting graphlets: Space vs time. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang (Eds.). ACM, 557–566. DOI: https://doi.org/10.1145/3018661.3018732

[12] Zhan Bu, Hui-Jia Li, Chengcui Zhang, Jie Cao, Aihua Li, and Yong Shi. 2020. Graph K-means based on leader identification, dynamic game, and opinion dynamics. *IEEE Transactions on Knowledge and Data Engineering* 32, 7 (2020), 1348–1361. DOI: https://doi.org/10.1109/TKDE.2019.2903712

[13] Zhan Bu, Hui-Jia Li, Jie Cao, Zhen Wang, and Guangliang Gao. 2017. Dynamic cluster formation game for attributed graph clustering. *IEEE Transactions on Cybernetics* 49, 1 (2017), 328–341.

[14] Zhan Bu, Zhiang Wu, Jie Cao, and Yichuan Jiang. 2015. Local community mining on distributed and dynamic networks from a multiagent perspective. *IEEE Transactions on Cybernetics* 46, 4 (2015), 986–999.

[15] Jie Cao, Zhan Bu, Yuyao Wang, Huan Yang, Jiuchuan Jiang, and Hui-Jia Li. 2019. Detecting prosumer-community groups in smart grids from the multiagent perspective. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49, 8 (2019), 1652–1664.

[16] Chen Chen, Jingrui He, Nadya Bliss, and Hanghang Tong. 2015. On the connectivity of multi-layered networks: Models, measures and optimal control. In *Proceedings of the 2015 IEEE International Conference on Data Mining*. Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu (Eds.). IEEE Computer Society, 715–720. DOI: https://doi.org/10.1109/ICDM.2015.104

[17] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamás Sarlós. 2012. Are web users really Markovian? In *Proceedings of the 21st World Wide Web Conference 2012*. Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab (Eds.). ACM, 609–618. DOI: https://doi.org/10.1145/2187836.2187919

[18] Wai-Ki Ching and Michael K. Ng. 2006. *Markov Chains: Models, Algorithms and Applications*. Springer.

[19] Kim-Kwang Raymond Choo. 2008. *Money laundering risks of prepaid stored value cards*. Retrieved from https://www.aic.gov.au/publications/tandi/tandi363.

[20] Dawei Zhou, Jingrui He, Hasan Davulcu, and Ross Maciejewski. 2018. Motif-preserving dynamic local graph cut. In *Proceedings of the IEEE International Conference on Big Data*. 1156–1161. DOI: https://doi.org/10.1109/BigData.2018.8622263

[21] Chris H. Q. Ding, Tao Li, and Michael I. Jordan. 2008. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08)*. IEEE Computer Society, 183–192. DOI: https://doi.org/10.1109/ICDM.2008.130

[22] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3–5 (2010), 75–174.

[23] Dongqi Fu, Dawei Zhou, and Jingrui He. 2020. Local motif clustering on time-evolving graphs. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 390–400. DOI: https://dl.acm.org/doi/10.1145/3394486.3403081

[24] Shayan Oveis Gharan and Luca Trevisan. 2012. Approximating the expansion profile and almost optimal local graph clustering. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 187–196. DOI: https://doi.org/10.1109/FOCS.2012.85

[25] David F. Gleich, Lek-Heng Lim, and Yongyang Yu. 2015. Multilinear PageRank. *SIAM Journal on Matrix Analysis and Applicationsl* 36, 4 (2015), 1507–1541. DOI: https://doi.org/10.1137/140985160

[26] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 19:1–19:19. DOI: https://doi.org/10.1145/2827872

[27] August B. Hollingshead. 1975. Four Factor Index of Social Status. New Haven, CT.

[28] Chris Jay Hoofnagle. 2007. Identity theft: Making the known unknowns known. *Harvard Journal of Law and Technology* 21 (2007), 97.

[29] Christine Klymko, David F. Gleich, and Tamara G. Kolda. 2014. Using triangles to improve community detection in directed networks. *CoRR* abs/1404.5874 (2014). arxiv:1404.5874 http://arxiv.org/abs/1404.5874.

[30] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. Retrieved from http://snap.stanford.edu/data.

[31] Hui-Jia Li, Zhan Bu, Zhen Wang, Jie Cao, and Yong Shi. 2018. Enhance the performance of network computation by a tunable weighting strategy. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2, 3 (2018), 214–223.

[32] Hui-Jia Li and Lin Wang. 2019. Multi-scale asynchronous belief percolation model on multiplex networks. *New Journal of Physics* 21, 1 (2019), 015005.

[33] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual analysis for anomaly detection in attributed networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. Carles Sierra (Ed.). ijcai.org, 2152–2158. DOI: https://doi.org/10.24963/ijcai.2017/299

[34] Wen Li and Michael K. Ng. 2014. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra* 62, 3 (2014), 362–385.

[35] Zhining Liu, Dawei Zhou, and Jingrui He. 2019. Towards explainable representation of time-evolving graphs via spatial-temporal graph attention networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 2137–2140. DOI: https://doi.org/10.1145/3357384.3358155

[36] Zhining Liu, Dawei Zhou, Yada Zhu, Jinjie Gu, and Jingrui He. 2020. Towards fine-grained temporal network representation via time-reinforced random walk. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, 4973–4980. Retrieved from https://aaai.org/ojs/index.php/AAAI/article/view/5936.

[37] László Lovász and Miklós Simonovits. 1990. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 346–354. DOI: https://doi.org/10.1109/FSCS.1990.89553

[38] László Lovász and Miklós Simonovits. 1993. Random walks in a convex body and an improved volume algorithm. *Random Structures & Algorithms* 4, 4 (1993), 359–412. DOI: https://doi.org/10.1002/rsa.3240040402

[39] Adrian E. Raftery. 1985. A model for high-order Markov chains. *Journal of the Royal Statistical Society: Series B (Methodological)* 47, 3 (1985), 528–539.

[40] Sheldon M. Ross. 2014. *Introduction to Probability Models*. Academic Press.

[41] Martin Rosvall, Alcides V. Esquivel, Andrea Lancichinetti, Jevin D. West, and Renaud Lambiotte. 2014. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications* 5, 1 (2014), 1–13.

[42] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64. DOI: https://doi.org/10.1016/j.cosrev.2007.05.001

[43] Yu Shi, Xinwei He, Naijing Zhang, Carl Yang, and Jiawei Han. 2019. User-guided clustering in heterogeneous information networks via motif-based comprehensive transcription. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 361–377. DOI: https://doi.org/10.1007/978-3-030-46150-8_22

[44] Jirí Síma and Satu Elisa Schaeffer. 2006. On the NP-completeness of some graph cluster measures. In *Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science*. Jirí Wiedermann, Gerard Tel, Jaroslav Pokorný, Mária Bieliková, and Julius Stuller (Eds.), Vol. 3831. Springer, 530–537. DOI: https://doi.org/10.1007/11611257_51

[45] Daniel A. Spielman and Shang-Hua Teng. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing* 42, 1 (2013), 1–26. DOI: https://doi.org/10.1137/080744888

[46] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. 2013. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy (Eds.). ACM, 104–112. DOI: https://doi.org/10.1145/2487575.2487645

[47] Charalampos E. Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*. Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 1451–1460. DOI: https://doi.org/10.1145/3038912.3052653

[48] Tao Wu, Austin R. Benson, and David F. Gleich. 2016. General tensor spectral co-clustering for higher-order data. In *Proceedings of the Advances in Neural Information Processing Systems*. Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 2559–2567. Retrieved from http://papers.nips.cc/paper/6376-general-tensor-spectral-co-clustering-for-higher-order-data.

[49] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 555–564. DOI: https://doi.org/10.1145/3097983.3098069

[50] Si Zhang, Dawei Zhou, Mehmet Yigit Yildirim, Scott Alcorn, Jingrui He, Hasan Davulcu, and Hanghang Tong. 2017. HiDDen: Hierarchical dense subgraph detection with application to financial fraud detection. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. Nitesh V. Chawla and Wei Wang (Eds.). SIAM, 570–578. DOI: https://doi.org/10.1137/1.9781611974973.64

[51] Dawei Zhou and Jingrui He. 2019. Gold panning from the mess: Rare category exploration, exposition, representation, and interpretation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 3213–3214. DOI: https://doi.org/10.1145/3292500.3332268

[52] Dawei Zhou, Jingrui He, K. Selçuk Candan, and Hasan Davulcu. 2015. MUVIR: Multi-view rare category detection. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 4098–4104. http://ijcai.org/Abstract/15/575

[53] Dawei Zhou, Jingrui He, Yu Cao, and Jae-sun Seo. 2016. Bi-level rare temporal pattern detection. In *Proceedings of the IEEE 16th International Conference on Data Mining*. Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu (Eds.). IEEE Computer Society, 719–728. DOI: https://doi.org/10.1109/ICDM.2016.0083

[54] Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. 2018. SPARC: Self-paced network representation for few-shot rare category characterization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Yike Guo and Faisal Farooq (Eds.). ACM, 2807–2816. DOI: https://doi.org/10.1145/3219819.3219968

[55] Dawei Zhou, Arun Karthikeyan, Kangyang Wang, Nan Cao, and Jingrui He. 2017. Discovering rare categories from graph streams. *Data Mining and Knowledge Discovery* 31, 2 (2017), 400–423. DOI: https://doi.org/10.1007/s10618-016-0478-6

[56] Dawei Zhou, Kangyang Wang, Nan Cao, and Jingrui He. 2015. Rare category detection on time-evolving graphs. In *Proceedings of the 2015 IEEE International Conference on Data Mining*. Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu (Eds.). IEEE Computer Society, 1135–1140. DOI: https://doi.org/10.1109/ICDM.2015.120

[57] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A local algorithm for structure-preserving graph cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 655–664. DOI: https://doi.org/10.1145/3097983.3098015

[58] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 401–411. DOI: https://dl.acm.org/doi/10.1145/3394486.3403082

[59] Dawei Zhou, Lecheng Zheng, Jiejun Xu, and Jingrui He. 2019. Misc-GAN: A multi-scale generative model for graphs. *Frontiers Big Data* 2 (2019), 3. DOI: https://doi.org/10.3389/fdata.2019.00003

[60] Yao Zhou and Jingrui He. 2016. Crowdsourcing via tensor augmentation and completion. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 2435–2441. Retrieved from http://www.ijcai.org/Abstract/16/347.

[61] Yao Zhou, Lei Ying, and Jingrui He. 2017. MultiC^2: An optimization framework for learning from task and worker dual heterogeneity. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. Nitesh V. Chawla and Wei Wang (Eds.). SIAM, 579–587. DOI: https://doi.org/10.1137/1.9781611974973.65

Received September 2019; revised May 2020; accepted September 2020