

A Local Algorithm for Structure-Preserving Graph Cut

Dawei Zhou* Si Zhang* Mehmet Yigit Yildirim* Scott Alcorn[†]
Hanghang Tong* Hasan Davulcu* Jingrui He*

ABSTRACT

Nowadays, large-scale graph data is being generated in a variety of real-world applications, from social networks to co-authorship networks, from protein-protein interaction networks to road traffic networks. Many existing works on graph mining focus on the vertices and edges, with first-order Markov chain as the underlying model. They fail to explore the high-order network structures, which are of key importance in many high impact domains. For example, in bank customer personally identifiable information (PII) networks, the start structures often correspond to a set of synthetic identities; in financial transaction networks, the loop structures indicate the existence of money laundering; in signed networks, the triangle structures play an essential role in the balance theory for edge prediction. In this paper, we focus on mining user-specified high-order network structures, and aim to find a structure-rich sub-graph which does not break many such structures by separating the sub-graph from the rest.

A key challenge associated with finding a structure-rich sub-graph is the prohibitive computational cost. To address this problem, inspired by the family of local graph clustering algorithms for efficiently identifying a low-conductance cut without exploring the whole entire graph, we propose to generalize the key idea to model high-order network structures. 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 any user-specified high-order network structures. Then we propose a novel High-Order Structure-Preserving Local Clustering framework named *HOSPLOC*. It starts with a seed node, and iteratively explores its neighborhood until it finds a sub-graph with a small high-order conductance. Furthermore, we analyze its performance in terms of both effectiveness and efficiency. The experimental results on both synthetic graphs and real graphs demonstrate the effectiveness and efficiency of our proposed *HOSPLOC* algorithm.

KEYWORDS

Local Clustering Algorithm, High-Order Network Structure

ACM Reference format:

Dawei Zhou* Si Zhang[1] Mehmet Yigit Yildirim[1] Scott Alcorn[†]
Hanghang Tong[1] Hasan Davulcu[1] Jingrui He[1] . 2017. A Local Algorithm for Structure-Preserving Graph Cut. In *Proceedings of ACM KDD conference, Halifax, Nova Scotia - Canada, August 2017 (KDD'2017)*, 11 pages. DOI: 10.475/123_4

1 INTRODUCTION

Given a massive graph and an initial vertex, to identify a good partition from the original graph - a subset of vertices with minimum conductance, is an NP-complete problem [19]. Local graph clustering algorithms present an important class of tools for efficiently discovering a dense subgraph contains or closes to a given vertex without exploring the whole graph. They typically use a diffusion based on a first-order Markov chain to find a local cut with an upper-bounded conductance. While the simple matrix computations and promising theory guaranteed behind these methods, most existing works are inherently limited to stand on two-dimensional network structures, i.e., edge.

While, in many real world applications, high-order connectivity patterns, e.g., triangle, loop and clique, are always essential to understand the fundamental patterns and underlying problems on networks. For example, the network structure of triangle has been proved the fundamental role in edge prediction on signed networks [16, 18]; the network structure of multi-hop loop may indicate the existence of money laundering in financial networks [10]; the network structure of star may correspond to a set of synthetic IDs in personally identifiable information (PII) networks of banks [17]. Thus, it could be a nature question that whether local graph clustering algorithms can be generalized to model any order any type network structures in an efficient way. To be specifically, the traditional local clustering algorithms aim to find a cut by exploring lower-order connectivity patterns in the level of nodes and edges, while the generalized local clustering algorithms would try to find the best partition on the basis of high-order connectivity patterns at the level of network motifs.

Despite its key importance, it remains a challenging task in generalizing local graph clustering algorithms to high-order network structures. Specifically, the following questions remains largely unknown. First (**Q1. Model**), it is not clear that how to conduct the generalized random walk with respect to any order network structures. Some existing works [14, 21] studied $(m-1)^{th}$ order Markov chain model, which is used to fit the observed data through the calculation of the $(m-1)^{th}$ order transition probability p_{i_1, i_2, \dots, i_m} . Yet, limited by the simple definition, it is unknown how to model any order any type network structures into a high-order random walks. Second (**Q2. Algorithm**), how can we design a high-order local

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

KDD'2017, Halifax, Nova Scotia - Canada

© 2016 Copyright held by the owner/author(s). 123-4567-24-567/08/06...\$15.00
DOI: 10.475/123_4

*Arizona State University. Email:{dzhou23, szhan172, yigityildirim, hanghang.tong, HasanDavulcu, jingrui.he}@asu.edu

[†]Early Warnings LLC. Email: scott.alcorn@earlywarning.com

graph clustering algorithm with mathematical guarantees on the optimality of obtained cluster and the scalability on massive graphs. Third (**Q3. Generalization**), how can we generalize our proposed algorithm to solve the real applications on different types of graphs, such as signed network, bi-partite graph and multi-partite graph.

Here, we propose algorithm *HOSPLOC* to address above challenges. The core of *HOSPLOC* is to approximately compute the distribution of high-order random walks [14, 21] that is directly based on high-order network structures and utilizing the idea of vector-based graph partition methods [22, 23, 29] to find a small conductance cut on the basis of high-order connectivity patterns. Our framework operates on the tensor representation of graph data which allows the users to specify which kind of network structure should be preserved in the returned local cluster. In addition, we provide a provable theoretical bounds on the effectiveness and time complexity of proposed algorithm. Furthermore, we show how *HOSPLOC* can be applied to the applications on various types of networks, e.g., signed networks, multi-partite networks. Finally, we justify the effectiveness, scalability and parameter sensitivity of the proposed *HOSPLOC* by empirical evaluations on multiple real-world networks.

The main contributions of this paper can be summarized as follows:

- (1) Definitions of adjacency tensor and transition tensor for high-order random walk incorporated on the basis of high-order network structures.
- (2) Generalized conductance metric for evaluating high-order partitions incorporated with any network structures.
- (3) A scalable local algorithm for structure-preserving graph cut.
- (4) Mathematical guarantees on the optimality of obtained cluster and the scalability on massive graphs.
- (5) Generalization and applications of *HOSPLOC* on various types of networks, such as signed networks, multi-partite networks.
- (6) Extensive experimental case studies on several real networks.

The rest of our paper is organized as follows. Related work is reviewed in Section 2, followed by the discussion of notations and preliminaries in Section 3. In Section 4, we propose the definition of transition tensor and our *HOSPLOC* algorithm as well as its theoretical analysis. Then, we introduce the generalization and application on heterogeneous graphs, such as signed graph, bi-partite graph and multi-partite graph in Section 5. Experimental results are presented in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

2.1 Local Spectral Clustering on Graphs

Local spectral clustering techniques provide a simple, near-linear time alternative to recursively identify a local sparse cut C with an upper-bounded conductance. In [28, 29], the authors introduce an almost-linear Laplacian linear solver and a local clustering algorithm, i.e., Nibble, which discovered small cuts that can be combined into balanced cuts and multi-way partitions in nearly linear time complexity. In [2, 3], the authors extend Nibble algorithm [29]

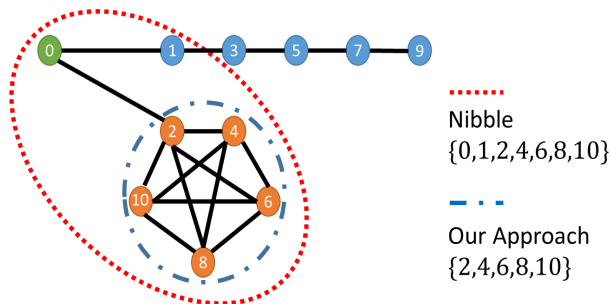


Figure 1: (Left) Network where node 0 connected with two kinds of network structures: clique and line. (Right) Local cuts founded by our method and the Nibble algorithm [29]. Both Nibble and our approach are given the same initial vertex, i.e., node 0. We observe that the cluster returned by Nibble contain the initial vertex, i.e., node 0, and all its direct neighbors, i.e., node 1 and node 2, while our method drop the initial vertex due to its redundancy in returned cluster. The reason is our proposed algorithm conducts the cut based on high-order random walk on the basis of 3-node line that introduced in Table 1, which largely helps preserving the high-order network structure and ensure the isolation of returned cluster. The cluster identified by our method is more compact than Nibble's cut.

by using personalized PageRank vector to produce cuts with less running time on undirected graphs and directed graphs. In [4, 5], the authors introduce a randomized local partitioning algorithm that find sparse cuts by simulating the volume-biased evolving set process. Moreover, to solve the problem that the approximation guarantee in local spectral clustering may be worse than the guarantee provided by Cheeger inequality, [13] 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. To my best of knowledge, it is the first local clustering framework that utilizes the diffusions distribution vector of high-order random walks to extract a graph cut which preserves rich user specified network structures.

2.2 High-order Markov Chain Models

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 [26], web browsing behavior [9] and wind turbine design [24]. To solve these problems, many previous works [1, 24, 30] approximate the limiting probability distribution of high-order Markov chain as a linear combination of transition probability matrix. More recently, in [21], the authors introduce a rank-1 approximation of high-order Markov chain limiting distribution and proposed a recursive algorithm for computing it. Later on, [14] introduce the problem of multi-linear PageRank and extended PageRank algorithm into the setting of high-order Markov chain, by using the rank-1 approximation in [21]. In [7], the authors introduce a novel non-Markovian 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 [6], the authors propose a tensor spectral clustering algorithm for the problem of partitioning sparse cuts with high-order network structures, by computing the second left eigenvector of the high-order Markov chain. [32] proposes a tensor spectral co-clustering method, by modeling higher-order data with a new stochastic process, i.e., the super-spacey random walk, which is a variant of a higher-order Markov chain. Compared with the existing high-order Markov chain models, we proposed a novel scalable local clustering algorithm that can identify clusters with a small conductance and also preserves any user specified high-order network structures in a nearly linear time complexity. Besides, we also provide a provable theoretical bounds on the effectiveness of proposed *HOSPLOC* algorithm when certain conditions are satisfied.

3 NOTATION AND PRELIMINARIES

In this section, we review the basics of random walk with Markov chain interpretation and the Nibble algorithm for local clustering on graphs [29], which paves the way for the proposed structure-preserving graph cut algorithm to be introduced in the next section.

3.1 Notation

Given 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. Also, 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 , etc. For convenience, we define the indicator vector χ_C to represent subset $C \subseteq V$ 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., $\mu(C) = \sum_{v \in C} d(v)$. Let $E(C, \bar{C})$ be the set of edges connecting vertices in C to vertices in the complementary set $\bar{C} = V - C$. The conductance of subset $C \subseteq V$ is therefore defined as $\Phi(C) = \frac{|E(C, \bar{C})|}{\min(\mu(C), \mu(\bar{C}))}$ [8], where $E(C, \bar{C}) = \{(u, v) | u \in C, v \in \bar{C}\}$.

3.2 Markov Chain Interpretation

The o^{th} order Markov chain S describes a stochastic process that satisfies [14]

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

where i_1, \dots, 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. If each vertex in graph G corresponds to a distinguishable state, we can interpret the transition matrix M as the transition matrix of the 1^{st} -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 irreducible [25], 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. In Section 4.1, we will introduce the idea of adjacency tensor and transition tensor for modeling high-order network structures, which will lead to high-order Markov chains and high-order random walks.

3.3 Nibble Algorithm

Given an undirected graph G and a parameter $\phi > 0$, to find a cut C from G such that $\Phi(C) \leq \phi$ or determine no such C exists is an NP-complete problem [27]. Nibble algorithm [29] is one of the earliest attempts to partition a graph with a bounded conductance with near linear time complexity. 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

$$[q]_\epsilon(u) = \begin{cases} q(u) & \text{if } q(u) \geq d(u)\epsilon \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

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 [29]

$$\epsilon = \frac{1}{(1800 \cdot (l+2)t_{last}2^b)}. \quad (3)$$

where

$$l = \lceil \log_2(\mu(V)/2) \rceil \quad (4)$$

and

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

Then Nibble applied the vector-based partition method [22, 23, 29] that utilizes the ordering of function I_x to produce a low conductance cut. For introducing function I_x mathematically, we firstly define $S_j(q)$ to be the set of top j vertices u that maximize $q(u)/d(u)$. That is $S_j(q) = \{\pi(1), \dots, \pi(j)\}$, 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

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

In the next section, we will introduce a novel high-order local clustering algorithm named *HOSPLOC* for producing dense local clusters. Both *HOSPLOC* and Nibble are local algorithms in the sense that their time complexity is related to the volume of the returned cluster. However, compared with Nibble, the clusters generated by *HOSPLOC* preserve any user specified high-order network structures, whereas the clusters generated by Nibble may not serve this benefit. By incorporating high-order network structure information, *HOSPLOC* can largely improve the understanding of the high-order connectivity patterns on real-world networks.





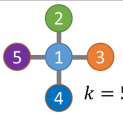
\mathbb{N}	Example	Illustration	Markov Chain
1 st -order	Vertex		0 th -order
2 nd -order	Edge		1 st -order
3 rd -order	3-node Line		2 nd -order
	Triangle		
k^{th} -order	k -node Star	 $k = 5$	$(k - 1)^{\text{th}}$ -order

Table 1: Network Structures \mathbb{N} and Markov Chains.

4 HIGH-ORDER NETWORK STRUCTURE AND THE HOSPLOC ALGORITHM

In the previous section, we introduced the basics of truncated local clustering, i.e., Nibble algorithm [29]. Now, we generalize the idea of truncated local clustering to produce clusters that preserve any user specified high-order network structures. We start by introducing the adjacency tensor and the associated transition tensor based on user specified high-order network structures, followed by the discussion on the stationary distribution of high-order random walk. Then, we introduce the definitions of high-order conductance and high-order diffusion core. Finally, we present the proposed high-order local clustering algorithm that preserves the user specified network structures with theoretical analysis on the effectiveness and efficiency.

4.1 Adjacency Tensor and Transition Tensor

For a binary undirected graph $G = (V, E)$, the corresponding adjacency matrix A is a binary matrix such that $A(i, j) = 1$ if $(i, j) \in E$, and $A(i, j) = 0$ otherwise. The adjacency matrix A could be considered as a matrix representation of the 2nd-order network structures on G , i.e., edges. In this paper, we use \mathbb{N} to denote the network structures on graph, where the order of network structure \mathbb{N} indicates the number of vertices forming \mathbb{N} . In real applications, the users might be interested in various network structures in different orders. For example, the triangle consisting of three nodes and three edges connecting each pair of the nodes in the triangle can be considered as a 3rd-order network structure \mathbb{N} . Table 1 summarizes 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 2nd-order network structures, and they correspond to the 1st-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).

Next, we extend the idea of adjacency matrix and introduce the definition of adjacency tensor \mathbf{T} and transition tensor \mathbf{P} for

representing the high-order random walk induced by high-order network structures \mathbb{N} .

Definition 4.1 (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 \mathbf{T} as follows

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

Definition 4.2 (Transition Tensor). Given a graph $G = (V, E)$ and the adjacency tensor \mathbf{T} for the k^{th} -order network structure \mathbb{N} , the corresponding transition tensor \mathbf{P} could be formed as

$$P(i_1, i_2, \dots, i_k) = \frac{T(i_1, i_2, \dots, i_k)}{\sum_{i_1=1}^n T(i_1, i_2, \dots, i_k)}. \quad (8)$$

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 \mathbf{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 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 3.1 can be considered as a special case of Definition 4.2 with 2nd-order network structure \mathbb{N} if we allow self-loops.

4.2 Stationary Distribution

For k^{th} -order network structure \mathbb{N} and the corresponding $(k - 1)^{\text{th}}$ -order random walk with transition tensor \mathbf{P} , if the stationary distribution \mathbf{X} exists, where \mathbf{X} is a $(k - 1)$ -dimensional tensor, then it satisfies [14]

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

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’ [6, 21], i.e.,

$$X(i_2, \dots, i_k) = q(i_2) \dots q(i_k) \quad (10)$$

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, [21] theoretically demonstrated the convergence and effectiveness of the nonnegative vector q if \mathbf{P} satisfies certain properties.

Following [6] and [21], in this paper, we also adopt ‘rank-one approximation’ and assume the stationary distribution of the high-order random walk satisfies Eq. 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 $(k-1)$ th-order random walk satisfies:

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

where \otimes denotes the Kronecker product. For example, for third-order network structure \mathbb{N} (e.g., triangle), the transition tensor $\mathbf{P} \in \mathbb{R}^{n \times n \times n}$ can be constructed based on Definition 4.2. Then, the 1-mode unfolding matrix \bar{P} of \mathbf{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

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

4.3 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} ; (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 want to ensure the partition preserves rich directed loops inside of the cluster and break such structure as less as possible. By this way, the returned cluster could include most of the suspicious identities involved in money laundering fraud. It is easy to see that the traditional definition of the conductance $\Phi(C)$ introduced in Subsection 3.1 does not serve this purpose. Therefore, we introduce the following generalized definition of conductance for preserving any high-order network structure \mathbb{N} .

Definition 4.3 (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

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

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

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

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.,

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

CLAIM 1. *Definition 4.3 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 two times of the traditional conductance $\Phi(C)$ introduced in Subsection 3.1.
- When \mathbb{N} represents triangles, $\Phi(C, \mathbb{N})$ is the same as the 'high-order conductance' ϕ_3 introduced in [6].

4.4 High-Order Diffusion Core

Similar as 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 **any user specified** high-order network structure, we are able to produce a local cluster that preserves such structures 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 \mathbf{P} defined in Definition 4.2, starting from the seed vertex v . Let $q^{(t)}$ denotes 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 [29], for 2^{nd} -nd order network structure and the associated 1^{st} -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 proposed the definition of the diffusion core to characterize the subset of vertices within the cluster, such that 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 4.4 (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

$$\chi_{C^{k, \xi}}^T q^{(t)} \leq \xi \frac{\text{cut}(C, \mathbb{N})}{\mu(C, \mathbb{N})} \quad (15)$$

where $q^{(t)}$ denotes the diffusion distribution of t -step high-order random walks and $\xi \geq 0$.

Note that the left hand side of Eq. 15, i.e., $\chi_{C^{k, \xi}}^T q^{(t)}$, 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 . While, on the right hand side of Eq. 15, the numerator could be considered as the total number of k^{th} -order random walk paths to escape cluster C , while the denominator could be thought as the total number of k^{th} -order random walk paths starting from C . We can see that $\chi_{C^{k, \xi}}^T q^{(t)}$ is positively correlated with $\frac{\text{cut}(C, \mathbb{N})}{\mu(C, \mathbb{N})}$, and ξ is a positive constant that controls the compactness of the diffusion core.

PROPOSITION 4.5. *For any cluster C and the k^{th} -order ξ -diffusion core $C^{k, \xi} \in C$, we have*

$$\chi_{C^{k, \xi}}^T q^{(t)} \leq \xi \Phi(C, \mathbb{N}) \quad (16)$$

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

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

Obviously, we can divided the proof into following two cases.

Case 1 : when $\mu(C, \mathbb{N}) \geq \mu(\bar{C}, \mathbb{N})$, we have $\Phi(C, \mathbb{N}) = \frac{\text{cut}(C, \mathbb{N})}{\mu(\bar{C}, \mathbb{N})} \geq \frac{\text{cut}(C, \mathbb{N})}{\mu(C, \mathbb{N})}$.

Case 2: when $\mu(C, \mathbb{N}) < \mu(\bar{C}, \mathbb{N})$, we have $\Phi(C, \mathbb{N}) = \frac{cut(C, \mathbb{N})}{\mu(C, \mathbb{N})}$. Thus, we have $\Phi(C, \mathbb{N}) \geq \frac{cut(C, \mathbb{N})}{\mu(C, \mathbb{N})}$. Meanwhile, by Definition. 4.4, it turns out that

$$\chi_{C^{\bar{k}, \xi}}^T q^{(t)} \leq \xi \frac{cut(C, \mathbb{N})}{\mu(C, \mathbb{N})} \leq \xi \Phi(C, \mathbb{N}).$$

□

4.5 The Proposed HOSPLOC Algorithm

The proposed algorithm *HOSPLOC* works by (1) approximately computing the distribution of high-order random walk starting at any vertex from which the walk does not mix rapidly, and (2) utilizing the idea of vector-based graph partition methods [22, 23, 29], in order to find a cut with a small conductance. Before we applying *HOSPLOC* algorithm, we need to generate the adjacency tensor \mathbf{T} and transition tensor \mathbf{P} by Definition 4.1 and Definition 4.2. Then, *HOSPLOC* algorithm is ready to run. Basically, *HOSPLOC* could be decomposed into three main steps. First of all, a high-order random walk is conducted starting at the seed vertex v to iteratively compute the diffusion distribution vector $q^{(t)}$. Then, we truncate all small entries in $q^{(t)}$ to 0. In this way, we limit the computation to the neighborhood of the seed. Finally, we apply the vector-based graph partition methods to search for the proper cut.

Now, we are ready to present our proposed *HOSPLOC* algorithm. It is given the transition tensor \mathbf{P} of the graph G taking into consideration of the high-order network structure \mathbb{N} , 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 . Step 1 to Step 4 are the initialization process. Step 1 constructs unfolding matrix \bar{P} of the transition tensor \mathbb{P} . Step 2 to Step 4 compute the truncation constant ϵ and the truncated initial distributions vectors $r^{(m)}$, $m = 1, \dots, k-1$. 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 time step; 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 Step 9 to Step 11, where condition (a) guarantees that the high-order 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.

Next, we analyze the proposed *HOSPLOC* algorithm in terms of both effectiveness and efficiency. Regarding 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 4.6 (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 $\mu(C' \cap C) \geq 2^{b-1}$.*

Algorithm 1 High-Order Structure-Preserved Local Clustering (HOSPLOC)

Require:

- (1) Transition tensor \mathbf{P} and transition matrix M ,
- (2) Initial vertex v ,
- (3) Conductance upper bound ϕ ,
- (4) Maximum iteration number t_{\max} ,
- (5) Parameters b, c_1, ξ .

Ensure:

Local cluster C ;

- 1: Construct the unfolding matrix \bar{P} of the transition tensor \mathbf{P} .
 - 2: Compute constant ϵ based on Eq. 3.
 - 3: Set initial distribution vectors $q^{(m)} = M^{(m-1)} \chi_v$, where $m = 1, \dots, k-1$.
 - 4: Compute truncated initial local distribution vectors $r^{(m)} = [q^{(m)}]_{\epsilon}$, $m = 1, \dots, k-1$.
 - 5: **for** $t = k : t_{\max}$ **do**
 - 6: Update distribution vector $q^{(t)} = \bar{P}(r^{(t-1)} \otimes \dots \otimes r^{(t-k+1)})$.
 - 7: Update truncated local distribution vectors $r^{(t)} = [q^{(t)}]_{\epsilon}$.
 - 8: **if** there exists a j such that: **then**
 - 9: (a) $\Phi(S_j(q^{(t)})) \leq \phi$,
 - 10: (b) $2^b \leq \lambda_j(q^{(t)})$,
 - 11: (c) $I_x(q^{(t)}, 2^b) \geq \xi / (c_1(l+2)2^b)$.
 - 12: **return** $C = S_j(q^{(t)})$ and **quit**.
 - 13: **else**
 - 14: **Return** $C = \emptyset$.
 - 15: **end if**
 - 16: **end for**
-

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 4.5, we have the following inequality

$$\chi_C^T q^{(t)} \leq \xi \Phi(C, \mathbb{N}) \leq \frac{\xi}{c_2(l+2)} \quad (17)$$

In Step 15 of Algorithm 1, condition (c) guarantees that

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

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 Eq. 18

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

Let j' be the smallest integer such that $\lambda_{j'}(q^{(t)}) \geq 2^b$. In Step 14 of Algorithm 1, condition (b) guarantees that $j' \leq j$. By Eq. 17 and Eq. 19, we have

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

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

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

□

Regarding the efficiency of *HOSPLOC*, we have the following lemma.

LEMMA 4.7 (EFFICIENCY OF *HOSPLOC*). *Given Graph G and k -order network structure \mathbb{N} , $k \geq 3$, the time complexity of *HOSPLOC* is bounded by $O\left(t_{\max} \left(\frac{2^b}{\phi^2}\right)^{\frac{k}{2}} \log^{\frac{3}{2}} k m\right)$.*

PROOF. The *HOSPLOC* algorithm runs at most t_{\max} iterations. We firstly analyze the time complexity of each for-loop in Algorithm 1. By taking advantage of the sparsity for large graphs, we will show that each iteration in Algorithm 1 takes time $O(\epsilon^{k/2})$. For computing the Kronecker product chain $r^{(t-1)} \otimes \dots \otimes r^{(t-k+1)}$, we will keep track of the non-zero elements in $r^{(t-1)} \dots r^{(t-k+1)}$. Here, we let V^t denote the set of vertices such that $\{u \in V^t | r^{(t)}(u) \geq 0\}$ and V^{t_m} be the set with the maximum number of vertices in $\{V^t | 1 \leq t_m \leq t_{\text{last}}\}$. For $1 \leq t \leq t_{\max}$, we have

$$O(r^{(t)} \otimes r^{(t-1)}) \leq O(r^{(t_m)} \otimes r^{(t_m)}) \leq O(\mu(V_{t_m}))$$

Since $\mu(V_{t_m}) = \sum_{u \in V_{t_m}} d(u)$ and [29] have proved that, for any $u \in V$, $d(u) \leq r^t(u)/\epsilon$. Thus, we have

$$O(r^{(t)} \otimes r^{(t-1)}) \leq O(\mu(V_{t_m})) \leq \sum_{u \in V_{t_m}} d(u) \leq \sum_{u \in V_{t_m}} r^{(t)}(u)/\epsilon \leq 1/\epsilon.$$

Similarly, we can prove that the Kronecker product chain $r^{(t-1)} \otimes \dots \otimes r^{(t-k+1)}$ can be computed in

$$O(\underbrace{r^{(t_m)} \otimes \dots \otimes r^{(t_m)}}_{k-1}) \leq O\left(\sqrt[k-1]{|V^{t_m}| \dots |V^{t_m}|}\right) \leq O\left(\frac{1}{\epsilon^{(k-1)/2}}\right).$$

After that, the multiplication between the unfolding matrix $\bar{P} \in \mathbb{R}^{n \times n^{(k-1)}}$ and the computed Kronecker product chain, i.e., $r^{(t-1)} \otimes \dots \otimes r^{(t-k+1)}$, requires time $O(1/\epsilon^{k/2})$. The truncation and normalization in Step 7 can be computed in time $O(|V^t|) \leq O(|V^{t_m}|)$. Step 8 to Step 15 require sorting the vertices in $|V^t|$ according to $r^{(t)}$, which takes time $O(|V^{t_m}| \log |V^{t_m}|)$.

Since we only consider network structure with order $k \geq 3$, we have

$$\frac{1}{\epsilon^{k/2}} \geq \frac{1}{\epsilon^{3/2}} \geq |\mu(V^{t_m})|^{3/2} \geq |\mu(V^t)|^{3/2} \geq |V^t|^{3/2} \geq |V^t| \log |V^t|.$$

It turns out that the running time of each for-loop iteration in Algorithm 1 is bounded by $O\left(\frac{1}{\epsilon^{k/2}}\right)$. Thus, the time complexity of Algorithm 1 is bounded by $O\left(\frac{t_{\max}}{\epsilon^{k/2}}\right)$. And, by Eq. 3, we can expand $O\left(\frac{t_{\max}}{\epsilon^{k/2}}\right)$ as follows

$$O\left(\frac{t_{\max}}{\epsilon^{k/2}}\right) = O\left(t_{\max} \left(\frac{2^b \log^3 \mu(V)}{\phi^2}\right)^{\frac{k}{2}}\right) = O\left(t_{\max} \left(\frac{2^b}{\phi^2}\right)^{\frac{k}{2}} \log^{\frac{3}{2}} k m\right).$$

□

For better illustrating the scalability of our proposed *HOSPLOC* algorithm, we show the comparison between our algorithm and another local clustering algorithm, i.e., Nibble, which is designed based on 1^{st} -order random walk and runs nearly linear in the number of edges of the graph. Suppose the maximum iteration number of Nibble and *HOSPLOC* are both upper-bounded by t_{\max} , then the

time complexity of Nibble is $O\left(\frac{t_{\max} 2^b \log^4 m}{\phi^2}\right)$. When we considering triangle as the specified network structure, i.e., $k = 3$, the time complexity of *HOSPLOC* algorithm is $O\left(t_{\max} \left(\frac{2^b}{\phi^2}\right)^{\frac{3}{2}} \log^{\frac{9}{2}} m\right)$.

Without considering the impact from various constants, we can see although our method need to conduct higher-order random walk in each iteration, it still runs nearly linear in the number of edges of the graph, which shows the high applicability of our method in real-world applications with sparse large-scaled graphs.

5 GENERALIZATION AND APPLICATIONS

In this section, we would like to introduce several generalities applications of our proposed algorithm on another three different types of graphs.

5.1 Community Detection on Signed Network

We firstly extend our proposed framework, i.e., *HOSPLOC*, for problems on signed graphs. In many real applications, the high-order network structures we want to study are presented with signed edges and play a significant role. In community detection [12], we may want to keep the social status of identities within the same community to be stable and keep the social status of identities from different communities to be unstable. In other words, we want to ensure stable configurations to be rich within communities and sparse in-between different communities; we want to ensure unstable configurations to be sparse within communities and rich in-between different communities. The adjacency tensor we use for this problem could be constructed as follows

$$T(i_1, i_2, \dots, i_k) = \begin{cases} 1 & \{i_1, i_2, \dots, i_k\} \text{ form stable structure} \\ 0 & \{i_1, i_2, \dots, i_k\} \text{ form unstable structure} \\ \alpha & \text{Otherwise} \end{cases} \quad (22)$$

where $\{i_1, i_2, \dots, i_k\} \in V$ and constant $0 < \alpha < 1$. Then, *HOSPLOC* could also be applied to identify communities with stable social status guarantee. Note that this method of constructing adjacency tensor could be also applied to detect communities with balanced network structure, by using social balanced theorem [18].

5.2 Adversarial Learning on Bi-partite Network

We now turn our attention to the applications on bi-partite graphs. Let us take the problem of user behavior modeling on adversarial network as an example. Given an adversarial network $B = (V_B, E_B)$, the bi-partite graph B contains two types of nodes, i.e., user nodes V_U and advertiser campaign nodes V_A , such that $V_B = \{V_U, V_A\}$. The edges E_B are only existing between user nodes V_U and advertiser campaign nodes V_A . Intuitively, similar customers tend to have similar purchase behaviors on adversarial network. Suppose we have two users u_1, u_2 and two adds a_1 and a_2 , both of u_1, u_2 show user-campaign interactions in the past, which forms a closed loop, ie, $u_1 \rightarrow a_1 \rightarrow u_2 \rightarrow a_2 \rightarrow u_1$. For modeling similar users' behavior, we would like to find a cluster of similar users and also their clicked advertiser campaigns. In this problem, we could consider the adversarial network as an undirected graph and the adjacency

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	8724	90962
Financial	PII	Multi-partite	375	519

Table 2: Statistics of the Networks.

tensor we use for *HOSPLOC* algorithm is

$$T(i_1, i_2, i_3, i_4) = \begin{cases} 1 & \{i_1, i_2, i_3, i_4\} \text{ form a 4-nodes loop} \\ 0 & \text{Otherwise} \end{cases} \quad (23)$$

where $\{i_1, i_2, i_3, i_4\} \in V_B$. By applying *HOSPLOC* algorithm, the returned partition C_B would represent a local user-campaign community, which consists similar users and users' frequently purchased advertiser campaigns.

5.3 Synthetic ID Detection on Multi-partite Network

We now show that how to solve the problems on multi-partite graphs, such as tripartite or four-partite graphs, by using our proposed *HOSPLOC* algorithm. Here, we take synthetic ID detection on personal identifiable information (PII) network as an example. PII network is a typical multi-partite network, where each partite set of nodes represent a certain type of PII, such as user names, user's accounts, and email addresses, and the edges are only existing between different partite sets of nodes. In synthetic ID fraud [17], criminals always use modified identify attributes, such as phone number, home address and email address, to combine with real users' information and create synthetic IDs to do malicious activities. For synthetic IDs, there is high possibilities that their PIIs would shared by multiple identities, which may compose a k -node star. Due to this reason, the adjacency tensor we use is

$$T(i_1, i_2, \dots, i_k) = \begin{cases} 1 & \{i_1, i_2, \dots, i_k\} \text{ form a } k\text{-node star} \\ 0 & \text{Otherwise} \end{cases} \quad (24)$$

where $\{i_1, i_2, \dots, i_k\} \in V_B$. Note that the returned partition may only contains part of the PII attributes for each included identities. But, it is viable to trace back from the extracted PIIs and discover the set of synthetic identities.

6 EXPERIMENTS

Now, we demonstrate the performance of the proposed *HOSPLOC* algorithm and its applications on bi-partite graph and multi-partite graph.

6.1 Experiment Setup

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

- **Collaboration Network:** We use two collaboration networks from[‡]. In network (Author), the nodes are authors, and an edge exists only when two authors have a co-authored paper. In network (Paper), the nodes are distinct papers, and an edge exists only when one paper cites another paper.
- **Infrastructure Network:** In network (Airline)[§], the nodes represent 2,833 airports, and the edges present the U.S. flights in one month interval. Network (Oregon) [20] is a network of routers in Autonomous Systems inferred from Oregon route-views between March 31, 2001 and May 26, 2001. Network (Power)[¶] contains the information of the power grid of the western states of U.S. A node represents a generator, a transformer or a substation, and an edge represents a power supply line.
- **Social Network:** Network (Epinion) [20] is a who-trust-whom online social network for consumer reviews. Each node represents a user and one edge exists if and only if when one user trust another user.
- **Review Network:** Network (Rating) [15] is a bipartite graph, where one side of nodes represent 643 users, and another side of nodes represent the 7,483 movies. Edges refer to the positive ratings, i.e., rating score larger than 2.5, on MovieLens website. Note that since we will model 4^{th} -order network structure, i.e., 4-node loop, on this network, to store a four dimensional transition tensor of original network, i.e., 300,000 vertices and 20M edges, requires to much memory. Thus, we extract a subgraph from the original network to perform the following experiments.
- **Financial Network:** Network (PII) is a multi-partite graph, which consisting five types of vertices, i.e., 112 bank accounts, 71 names, 80 emails, 35 addresses and 77 phone numbers. There is only edges between account vertices and PII vertices, which indicate the PIIs is perfectly matching the personal information of the accounts' holders.

Comparison Methods: In our experiments, we select both local and global graph clustering methods to compare with our methods. In specific, the comparison algorithm includes three local algorithms, i.e., (1) Nibble [29]; (2) Nibble-PageRank [2]; (3) LS-OQC [31], and two global clustering algorithm, i.e., (1) NMF [11]; (2) TSC [6]. 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 data sets are publicly available. The code of the proposed algorithms will be released on authors' website. For all the results reported, we set $c_1 = 140$ and $\xi = 1$. The experiments are mainly performed on a Windows machine with four 3.5GHz Intel Cores and 256GB RAM.

6.2 Effectiveness

The effectiveness comparison results are conducted on six real undirected graphs by following three evaluation metrics are shown in Fig. 2. Among them, (1) **Conductance** [8] measures the general quality of a cut on graph, which quantitatively indicate the compactness of a cut; (2) **3^{rd} -Order Conductance** could be computed base on Eq. 12 by treating triangle as the network structure \mathbb{N} , which

[‡]<https://aminer.org/data>

[§]<http://www.levmuchnik.net/Content/Networks/NetworkData.html>

[¶]<http://konect.uni-koblenz.de/networks/opsahl-powergrid>

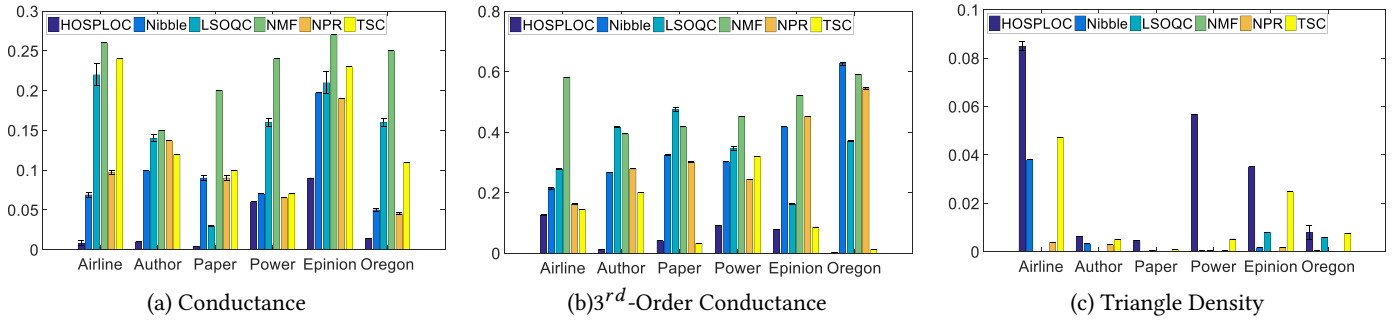


Figure 2: Effectiveness

estimate how well the network structure \mathbb{N} is preserved in returned cut from being broken by partition; (3) **Triangle Density** [8] computes the ratio of how rich the triangle is included in the returned cluster.

Moreover, for evaluating the convergence of local algorithms, we randomly select 30 vertices from the one cluster on each testing graph and run all the local algorithms multiples times by treating each of these nodes as initial vertex. In Fig. 2, the height of bars indicate the average value of evaluation metrics and the error bars represent the standard deviation of evaluations metrics in multiple runs. We have following observations: (1) In general, local algorithms perform better than the global algorithm, and our *HOSPLOC* algorithm consistently outperform the others on all the evaluation metrics. For example, compared with the best competitor, i.e., *TSC*, on network (Airline), *HOSPLOC* algorithm is 97% smaller on conductance, 12.2% smaller on 3^{rd} -Order Conductance, 80% larger on triangle density. (2) High-order Markov chain models, i.e., *HOSPLOC* and *TSC*, are better preserving triangles in the returned cluster. For example, on network (Epinion), both *HOSPLOC* and *TSC* returned a cluster with much higher triangle density and much lower 3^{rd} -Order Conductance. (2) *HOSPLOC* algorithm show a more robust convergence property than the other local clustering algorithm by comparing the size of error bar. For example, among the three local algorithms, only *HOSPLOC* algorithm returns the identical cluster on network (Paper) with different initial vertex.

6.3 Efficiency Analysis

Here, we evaluate the the efficiency of our proposed *HOSPLOC* algorithm with triangle as the specified network structure, by comparing with *Nibble* algorithm on synthetic graphs. Since our method is build 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 Fig. 3 is the average value of multiple runs by using 30 different initial vertex on the same graph. In Fig. 3(a), we show the running time of *HOSPLOC* and *Nibble* on a series of synthetic graphs with increasing number of vertices but fixed edge density 1%. We observe that although *HOSPLOC* requires more time than *Nibble* in each run, the running time of *HOSPLOC* increases nearly linear with respect to the size of graph $|V|$. In Fig. 3(b), we show the running time of *HOSPLOC* and *Nibble* versus the lower bound of output volume on the synthetic graph with 5000 vertices and 1% edge density, by keeping the other parameters fixed. We can see that the running time of *HOSPLOC* algorithm increases

exponentially with respect to 2^b , which is in accordance with our time complexity analysis, i.e., $O((2^b)^{\frac{3}{2}})$.

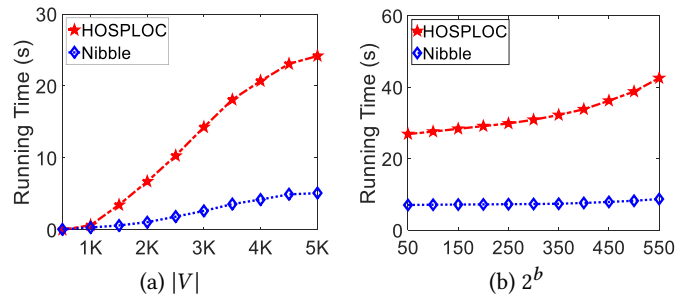


Figure 3: Efficiency Analysis

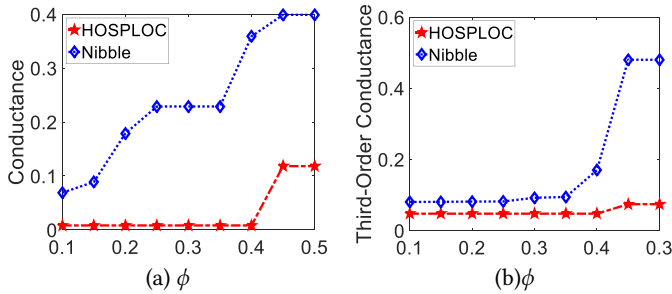
6.4 Parameter Analysis

In this subsection, we analysis the parameter sensitivity of our proposed *HOSPLOC* algorithm with triangle as the specified network structure, by comparing with *Nibble* algorithm on synthetic graph with 5000 vertices and 1% edge density. In the experiments, we evaluate the conductance and the 3^{rd} -order conductance of returned cut with different input parameter ϕ . In Fig. 4, we have following observations: (1) *HOSPLOC* shows a better convergence property than *Nibble*. For example, in Fig. 4 (a), we can see the output conductance of *HOSPLOC* converge to the minimum value when $\phi = 0.4$, while the output conductance of *Nibble* converge to its minimum value until $\phi = 0.1$. (2) Both the conductance and the 3^{rd} -order conductance of *HOSPLOC*'s cut are always smaller than *Nibble*'s cut with different input ϕ .

6.5 Case Study

Considering the above experiments of *HOSPLOC* algorithm are all conducted on the basis of 3^{rd} -order network structure, i.e., triangle. In this subsection, we will perform the following case studies to demonstrate the generality and effectiveness of our proposed *HOSPLOC* algorithm on multipartite networks with the problems incorporated with more complex and higher order network structures, such as 4-node loop and 5-node star.

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. Here, we treat 4-node

Figure 4: Parameter ϕ

loop, i.e., $user1 \rightarrow movie1 \rightarrow user2 \rightarrow movie2 \rightarrow user1$, as the underlying network structure, and the corresponding transition tensor could be generated by Eq. 23(a). Taking the advantages of network structure 4-node loop, our method could easily model the users' history action, i.e., ratings and reviews, and conduct a local partition with less connectivity to the complementary part of the network. Fig. 5(a) presents a miniature of the cluster identified by our proposed *HOSPLOC* algorithm, which reveals some interesting patterns. For example, in Fig. 5(a), we highlight a red loop and a blue loop. The red loop shows that both the third and the fourth users like the first movie and the fourth movies, while the blue loop represents that both the third and fifth user 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 movies, i.e., Back to School, and Karate Kid I,II,II, are all in the category of comedy movies. It turns out that our *HOSPLOC* algorithm returned 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 5-node star as the underlying network structure, and the corresponding transition tensor could be generated by Eq. 24. Typically, synthetic ID frauds always use the faked PII to open the synthetic bank accounts. This means that some PII would be shared by multiple suspicious bank accounts. Fig. 5(b) presents a subgraph of the returned cut by our proposed *HOSPLOC* algorithm. In Fig. 5(b), we can see that many PII is highly occupied 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 name and phone number with the account connected with blue lines. Comparing with the regular dense subgraph detection methods, our method can better preserve the structure of PII, i.e., 5-node star, in the returned cluster, which may better identify the set of suspicious accounts with highly shared PII.

7 CONCLUSIONS

In this paper, we proposed a local clustering framework, i.e., *HOSPLOC* that gives the user the flexibility to modeling any high-order network structures and returns a small conductance cut which largely preserves the user specified network structures in the cut. In addition, we provided mathematical guarantees on the optimality

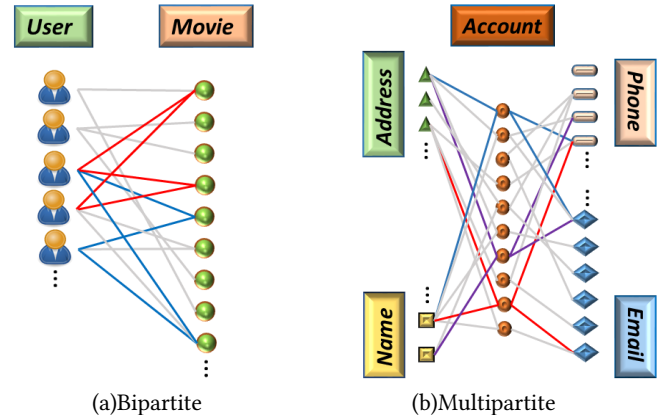


Figure 5: Case Study; (a) an example of detected community by *HOSPLOC* on network Rating; (b) an example of suspicious synthetic ID fraud revealed by our algorithms

of obtained cluster and also the time complexity on massive graphs. Furthermore, we generalize the proposed *HOSPLOC* algorithm and try to solve multiple real problems on signed networks, bipartite graphs and multi-partite graphs, by exploring the useful high-order network connectivity patterns, such as balanced triangle, loop and star. In the end, the extensive empirical evaluations on a diverse set of networks demonstrate the effectiveness and scalability of our proposed *HOSPLOC* algorithm.

REFERENCES

- [1] SR Adke and SR Deshmukh. 1988. Limit distribution of a high order Markov chain. *Journal of the Royal Statistical Society. Series B (Methodological)* (1988), 105–108.
- [2] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.
- [3] Reid Andersen, Fan Chung, and Kevin Lang. 2007. Local partitioning for directed graphs using PageRank. In *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 166–178.
- [4] Reid Andersen, Shayan Oveis Gharan, Yuval Peres, and Luca Trevisan. 2016. Almost Optimal Local Graph Clustering Using Evolving Sets. *Journal of the ACM (JACM)* 63, 2 (2016), 15.
- [5] Reid Andersen and Yuval Peres. 2009. Finding sparse cuts locally using evolving sets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 235–244.
- [6] 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, Vancouver, BC*. SIAM, 118–126.
- [7] Austin R Benson, David F Gleich, and Lek-Heng Lim. 2016. The Spacey Random Walk: A stochastic Process for Higher-Order Data. *arXiv preprint arXiv:1602.02102* (2016).
- [8] Béla Bollobás. 2013. *Modern graph theory*. Vol. 184. Springer Science & Business Media.
- [9] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamas Sarlos. 2012. Are web users really markovian?. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 609–618.
- [10] Kim-Kwang Raymond Choo. 2008. *Money laundering risks of prepaid stored value cards*. Australian Institute of Criminology.
- [11] Chris Ding, Tao Li, and Michael I Jordan. 2008. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 183–192.
- [12] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3 (2010), 75–174.
- [13] Shayan Oveis Gharan and Luca Trevisan. 2012. Approximating the expansion profile and almost optimal local graph clustering. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE, 187–196.

- [14] David F Gleich, Lek-Heng Lim, and Yongyang Yu. 2015. Multilinear PageRank. *SIAM J. Matrix Anal. Appl.* 36, 4 (2015), 1507–1541.
- [15] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.
- [16] August B Hollingshead and others. 1975. Four factor index of social status. (1975).
- [17] Chris Joy Hoofnagle. 2007. Identity theft: Making the known unknowns known. *Harv. J.L. & Tech.* 21 (2007), 97.
- [18] Deni Khanafiah and Hokky Situngkir. 2004. Social balance theory: revisiting Heiderfis balance theory for many agents. (2004).
- [19] Tom Leighton and Satish Rao. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)* 46, 6 (1999), 787–832.
- [20] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>. (June 2014).
- [21] 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.
- [22] László Lovász and Miklós Simonovits. 1990. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*. IEEE, 346–354.
- [23] 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.
- [24] Adrian E Raftery. 1985. A model for high-order Markov chains. *Journal of the Royal Statistical Society. Series B (Methodological)* (1985), 528–539.
- [25] Sheldon M Ross. 2014. *Introduction to probability models*. Academic press.
- [26] 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 (2014).
- [27] Jiří Šima and Satu Elisa Schaeffer. 2006. On the NP-completeness of some graph cluster measures. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 530–537.
- [28] Daniel A Spielman and Shang-Hua Teng. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 81–90.
- [29] 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 J. Comput.* 42, 1 (2013), 1–26.
- [30] Jozef L Teugels. 2008. Markov Chains: Models, Algorithms and Applications. *J. Amer. Statist. Assoc.* 103, 483 (2008), 1325–1325.
- [31] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria 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*. ACM, 104–112.
- [32] Tao Wu, Austin R Benson, and David F Gleich. 2016. General tensor spectral co-clustering for higher-order data. In *Advances in Neural Information Processing Systems*. 2559–2567.