

# 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 the 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 star structures often correspond to a set of synthetic identities; in financial transaction networks, the loop structures may indicate the existence of money laundering. In this paper, we focus on mining *user-specified* high-order network structures and aim to find a *structure-rich* subgraph which does not break many such structures by separating the subgraph from the rest.

A key challenge associated with finding a *structure-rich* subgraph 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 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 algorithm named *HOSPLOC*, which runs in polylogarithmic time with respect to the number of edges in the graph. It starts with a seed vertex and iteratively explores its neighborhood until a subgraph with a small high-order conductance is found. 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.

## 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)*, 9 pages. 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

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

## 1 INTRODUCTION

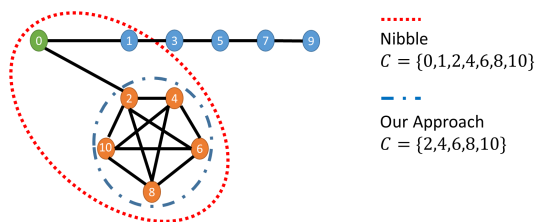
Given a massive graph and an initial vertex, to identify a good partition from the original graph - a subset of vertices with minimum conductance, is an *NP-complete problem* [18]. Local graph clustering algorithms provide an important class of tools to efficiently discover a dense subgraph that contains or is close to a given vertex without exploring the whole graph. Despite the elegant theoretical analysis, most existing works are inherently limited to simple network structures, i.e., vertices and edges, with the first-order Markov chain as the underlying model.

However, in many real-world applications, high-order network structures, e.g., triangles, loops, and cliques, are essential for exploring useful patterns on networks. For example, the multi-hop loop structure may indicate the existence of money laundering in financial networks [9]; the star structure may correspond to a set of synthetic identities in personally identifiable information (PII) networks of bank customers [13]. Therefore, an intriguing research question is whether local graph clustering algorithms can be generalized to model *any user-specified* network structures in an efficient way. Specifically, the traditional local clustering algorithms aim to find a cut by exploring the first-order connectivity patterns on the level of vertices and edges, while we aim to find the best cut based on high-order connectivity patterns on the level of network motifs.

Despite its key importance, it remains a challenge to generalize local graph clustering algorithms to model high-order network structures. Specifically, we need to answer the following questions. First (**Q1. Model**), it is not clear that how to conduct the generalized random walk with respect to any high-order network structures. Some existing works [5, 19] have studied second-order random walks based on  $3^{rd}$ -order network structures. However, it is unknown how to construct high-order random walks on the basis of *any user-specified* network structures. Second (**Q2. Algorithm**), how can we design a high-order local clustering algorithm that produces a graph cut rich in the *user-specified* network structures in an efficient way? This question has been largely overlooked in the previous research. Third (**Q3. Generalization**), how can we generalize our proposed algorithm to solve the real-world problems with different types of graphs, such as on bipartite and multipartite graphs?

To address these problems, in this paper, we propose a novel local algorithm for structure-preserving graph cut named *HOSPLOC*. The core of *HOSPLOC* is to approximately compute the distribution of high-order random walk [19] that is directly based on any *user-specified* high-order network structures, and then utilize the idea of vector-based graph partition methods [16, 17, 23] to find a cut with a small high-order conductance. Our algorithm operates on the tensor representation of graph data which allows the users to specify what kind of network structures should be preserved in the returned cluster. In addition, we provide analyses regarding the effectiveness

and efficiency of the proposed algorithm. Furthermore, we present how *HOSPLOC* can be applied to the applications with various types of networks, e.g., bipartite networks and multipartite networks. Finally, we evaluate the performance of *HOSPLOC* from multiple aspects using various real-world networks. Figure 1 compares the clusters returned by the proposed *HOSPLOC* algorithm and the Nibble algorithm [23], which shows that *HOSPLOC* is better at partitioning a subgraph with rich *user-specified high-order* network structures.



**Figure 1: A synthetic network where vertex 0 is connected with two kinds of network structures: clique and line. Local cut found by *HOSPLOC* (within the blue dash-dot line) and the Nibble algorithm [23] (within the red dotted line) with the same initial vertex, i.e., vertex 0, where *HOSPLOC* is based on the network structure of a 3-node line.**

The main contributions of the paper are summarized below.

- (1) Definition of adjacency tensor and transition tensor for high-order random walk induced by high-order network structures.
- (2) A local clustering algorithm named *HOSPLOC* for structure-preserving graph cut with *polylogarithmic* time complexity regarding the number of edges.
- (3) Theoretical analyses regarding the effectiveness and efficiency of *HOSPLOC*.
- (4) Generalizations and applications of *HOSPLOC* on bipartite and multipartite networks.
- (5) Extensive experimental results on both synthetic and real networks demonstrating the performance of the proposed *HOSPLOC* algorithm.

The rest of our paper is organized as follows. Related works are reviewed in Section 2, followed by the introduction of notation and preliminaries in Section 3. In Section 4, we present our proposed *HOSPLOC* algorithm as well as the analyses regarding its effectiveness and efficiency. Then we introduce its generalizations and applications on bipartite graph and multipartite graph in Section 5. Experimental results are presented in Section 6 before we conclude the paper in Section 7.

## 2 RELATED WORK

### 2.1 Local Spectral Clustering on Graphs

Local spectral clustering techniques provide a simple, efficient time alternative to recursively identify a local sparse cut  $C$  with an upper-bounded conductance. In [22, 23], 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 [1], the authors extend Nibble algorithm [23] by using personalized PageRank vector to produce cuts with less

running time on undirected graphs and directed graphs. More recently, [12] 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 [2, 3], the authors introduce randomized local partitioning algorithms that find sparse cuts by simulating the volume-biased evolving set process. However, to my best of knowledge, this paper is the first local clustering framework that focuses on modeling high-order network structures and aims to find a *structure-rich* subgraph with a *polylogarithmic* time complexity in the number of edges.

### 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 [21], web browsing behavior [8] and wind turbine design [20]. To solve these problems, many previous works [4, 20, 26] approximate the limiting probability distribution of high-order Markov chain as a linear combination of transition probability matrix. More recently, in [19], the authors introduce a rank-1 approximation of high-order Markov chain limiting distribution and propose a recursive algorithm to compute it. Later on, [11] 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 [6], 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. After that, [27] 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 any *user-specified* high-order network structures in a *polylogarithmic* time complexity. Besides, we also provide provable theoretical bounds on the effectiveness and efficiency of the proposed *HOSPLOC* algorithm.

## 3 NOTATION AND PRELIMINARIES

In this section, we review the basics of random walk with the Markov chain interpretation and the Nibble algorithm for local clustering on graphs [23], which pave 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. For convenience, we define the indicator vector  $\chi_C$  to represent the subset  $C \subseteq V$ , i.e.,  $\chi_C(v) = 1$  when  $v \in C$ , otherwise 0. In particular, the initial distribution of a random walk starting from vertex  $v$  could be denoted as  $\chi_v$ .

The volume of a subset  $C \subseteq V$  is defined as the summation of vertex degrees in  $C$ , i.e.,  $\mu(C) = \sum_{v \in C} d(v)$ . The conductance of subset  $C \subseteq V$  is therefore defined as  $\Phi(C) = \frac{|E(C, \bar{C})|}{\min(\mu(C), \mu(\bar{C}))}$  [7], 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  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of the matrix  $M$ , and  $M(i, :)$  is the  $i^{\text{th}}$  row of  $M$ , etc.

### 3.2 Markov Chain Interpretation

The  $o^{\text{th}}$  order Markov chain  $S$  describes a stochastic process that satisfies [11]

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

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  $1^{\text{st}}$ -order Markov chain. Specifically, the transition probability between vertex  $i$  and vertex  $j$  is given by  $M(i, j) = Pr(S_{t+1} = i | S_t = j)$ . In Section 4.1, we introduce the idea of adjacency tensor and transition tensor for modeling the high-order network structures, which will lead to the 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 to determine no such  $C$  exists is an NP-complete problem [24]. Nibble algorithm [23] 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

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

where  $q \in \mathbb{R}^n$  is the distribution vector over all the vertices in the graph, and  $\epsilon$  is the truncation threshold that can be computed as follows [23]

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

where  $l$  can be computed as  $l = \lceil \log_2(\mu(V)/2) \rceil$ , and  $t_{last}$  can be computed as  $t_{last} = (l+1) \left\lceil \frac{2}{\phi^2} \ln \left( c_1(l+2) \sqrt{\mu(V)/2} \right) \right\rceil$ .

Then, Nibble applies the vector-based partition method [16, 17, 23] 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  $S_j(q) = \{\pi(1), \dots, \pi(j)\}$ , where  $\pi$  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  $S_j(q)$ .





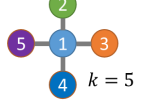
$\mathbb{N}$	Example	Illustration	Markov Chain
1 <sup>st</sup> -order	Vertex		0 <sup>th</sup> -order
2 <sup>nd</sup> -order	Edge		1 <sup>st</sup> -order
3 <sup>rd</sup> -order	3-node Line		2 <sup>nd</sup> -order
	Triangle		
$k^{\text{th}}$ -order	$k$ -node Star	 $k = 5$	$(k-1)^{\text{th}}$ -order

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

Finally, the function  $I_x$  is defined as follows

$$I_x(q, \lambda_j(q)) = \frac{q(\pi(i))}{d(\pi(i))}. \quad (4)$$

In the next section, we will introduce the high-order structure preserving local clustering framework, i.e., *HOSPLOC*. Compared to Nibble, *HOSPLOC* can model any *user-specified* network structure and conduct a *structure-rich* cut with a small conductance. Moreover, similar as Nibble, *HOSPLOC* runs in polylogarithmic time with respect to the number of edges in the graph.

## 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 [23]. 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 the *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 *HOSPLOC* with theoretical analyses on the effectiveness and efficiency.

### 4.1 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  $G$ . However, in many real applications, we may want to explore and capture more complex and high-order network structures. Table 1 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<sup>nd</sup>-order network structures, and they correspond to the 1<sup>st</sup>-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  $k^{\text{th}}$ -order network structures correspond to the  $(k-1)^{\text{th}}$ -order Markov chain (random walk).

To model any *user-specified* network structure  $\mathbb{N}$ , we introduce the definition of adjacency tensor  $\mathbf{T}$  and the transition tensor  $\mathbf{P}$  to represent the high-order random walk induced by the high-order network structures  $\mathbb{N}$ .

**Definition 4.1 (Adjacency Tensor).** Given a graph  $G = (V, E)$ , the  $k^{\text{th}}$ -order network structure  $\mathbb{N}$  on  $G$  could be represented in a  $k$ -dimensional adjacency tensor  $\mathbf{T}$  as follows

$$T(i_1, i_2, \dots, i_k) = \begin{cases} 1 & \{i_1, i_2, \dots, i_k\} \subseteq V \text{ and form } \mathbb{N}. \\ 0 & \text{Otherwise.} \end{cases} \quad (5)$$

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

$$P(i_1, i_2, \dots, i_k) = \frac{T(i_1, i_2, \dots, i_k)}{\sum_{i_1=1}^n T(i_1, i_2, \dots, i_k)} \quad (6)$$

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  $k^{\text{th}}$ -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 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 3.1 can be considered as a special case of Definition 4.2 with the 2<sup>nd</sup>-order network structure  $\mathbb{N}$ , if we allow self-loops.

## 4.2 Stationary Distribution

For the  $k^{\text{th}}$ -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 [11]

$$X(i_1, i_2, \dots, i_{k-1}) = \sum_{i_k} P(i_1, i_2, \dots, i_k) X(i_2, \dots, i_k). \quad (7)$$

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’ [5, 19], i.e.,

$$X(i_2, \dots, i_k) = q(i_2) \dots q(i_k) \quad (8)$$

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

Following [5] and [19], in this paper, we also adopt ‘rank-one approximation’ and assume the stationary distribution of the high-order random walk satisfies Eq. 8. To further simplify the notation, we let  $\bar{P}$  denote the  $(k-2)$ -mode unfolding matrix of the  $k$ -dimensional transition tensor  $\mathbf{P}$ . Thus, the  $(k-1)^{\text{th}}$ -order random walk satisfies:

$$q = \bar{P}(q \otimes \dots \otimes q) \quad (9)$$

where  $\otimes$  denotes the Kronecker product. For example, for the 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 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 Subsection 3.1 does not serve this purpose. Therefore, we introduce the following generalized definition of conductance to preserve any high-order network structure  $\mathbb{N}$ .

**Definition 4.3 ( $k^{\text{th}}$ -order Conductance).** For any cluster  $C$  in graph  $G$  and the  $k^{\text{th}}$ -order network structure  $\mathbb{N}$ , the  $k^{\text{th}}$ -order Conductance  $\Phi(C, \mathbb{N})$  is defined as

$$\Phi(C, \mathbb{N}) = \frac{\text{cut}(C, \mathbb{N})}{\min\{\mu(C, \mathbb{N}), \mu(\bar{C}, \mathbb{N})\}} \quad (10)$$

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.,

$$\begin{aligned} \text{cut}(C, \mathbb{N}) &= \sum_{i_1, \dots, i_k \in V} T(i_1, \dots, i_k) - \sum_{i_1, i_2, \dots, i_k \in C} T(i_1, \dots, i_k) \\ &\quad - \sum_{i_1, \dots, i_k \in \bar{C}} T(i_1, \dots, i_k) \end{aligned} \quad (11)$$

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.,

$$\begin{aligned} \mu(C, \mathbb{N}) &= \sum_{i_1 \in C; i_2, \dots, i_k \in V} T(i_1, i_2, \dots, i_k) \\ \mu(\bar{C}, \mathbb{N}) &= \sum_{i_1 \in \bar{C}; i_2, \dots, i_k \in V} T(i_1, i_2, \dots, i_k). \end{aligned} \quad (12)$$

**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 twice 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'  $\phi_3$  introduced in [5].

#### 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 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  $\mathbf{P}$  defined in Definition 4.2, starting from the seed vertex  $v$ . Let  $q^{(t)}$  denote the distribution vector over all the vertices after the  $t^{\text{th}}$  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 [23], for the 2<sup>nd</sup>-order network structure and the associated 1<sup>st</sup>-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 4.4 ( $k^{\text{th}}$ -Order  $\xi$ -Diffusion Core).** For any cluster  $C$ , we define  $C^{k, \xi} \in C$  to be the  $k^{\text{th}}$ -order  $\xi$ -diffusion core of  $C$ , such that

$$\chi_{C^{k, \xi}}^T q^{(t)} \leq \xi \frac{\text{cut}(C, \mathbb{N})}{\mu(C, \mathbb{N})} \quad (13)$$

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

Note that the left hand side of Eq. 13,  $\chi_C^T q^{(t)}$ , 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  $C$ . On the right hand side of Eq. 13, the numerator could be considered as the total number of the  $k^{\text{th}}$ -order random walk paths to escape cluster  $C$ , while the denominator could be regarded as the total number of the  $k^{\text{th}}$ -order random walk paths starting from  $C$ . It is easy to see that  $\chi_{C^{k, \xi}}^T q^{(t)}$  is positively correlated with  $\frac{\text{cut}(C, \mathbb{N})}{\mu(C, \mathbb{N})}$ . Since, for a given  $C$ ,  $\chi_{C^{k, \xi}}^T q^{(t)}$  is a computable constant, we consider Eq. 13 as the compactness constraint for the  $k^{\text{th}}$ -Order  $\xi$ -Diffusion Core  $C^{k, \xi} \in C$ .

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

$$\chi_{C^{k, \xi}}^T q^{(t)} \leq \xi \Phi(C, \mathbb{N}). \quad (14)$$

**PROOF.** Given a cluster  $C \in V$  and a  $k^{\text{th}}$ -order network structure  $\mathbb{N}$ , the corresponding  $k^{\text{th}}$ -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 divide the proof into the following two cases.

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

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

Thus, we have  $\Phi(C, \mathbb{N}) \geq \frac{\text{cut}(C, \mathbb{N})}{\mu(C, \mathbb{N})}$ . Meanwhile, by Definition. 4.4, it turns out that

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

#### 4.5 The Proposed HOSPLOC Algorithm

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; (3) apply the vector-based graph partition method [16, 17, 23] to search for a *structure-rich* cut with a small conductance.

Now, we are ready to present our proposed HOSPLOC algorithm. The given input are the transition tensor  $\mathbf{P}$ , the transition matrix  $M$ , the seed vertex  $v$ , the conductance upper-bound  $\phi$ , the maximum iteration number  $t_{\max}$ , and the constants  $b, c_1, \xi$ . 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  $\mathbf{P}$ . Step 2 to Step 4 compute the truncation constant  $\epsilon$  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 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 Step 9 to Step 11, where condition (a) guarantees that the conductance of  $C$  is upper-bounded by  $\phi$ , 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 effectiveness and efficiency. Regarding the effectiveness, we will show that for any cluster  $C$ , if the seed vertex comes from the  $k^{\text{th}}$ -order  $\xi$ -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  $\mu(C' \cap C) \geq 2^{b-1}$ .

**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

$$\chi_{C'}^T q^{(t)} \leq \xi \Phi(C, \mathbb{N}) \leq \frac{\xi}{c_2(l+2)}. \quad (15)$$

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

- (1) transition tensor  $\mathbf{P}$  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  $\mathbf{P}$ .
- 2: Compute constant  $\epsilon$  based on Eq. 3.
- 3: Set initial distribution vectors  $q^{(t)} = M^{(t-1)}\chi_v$ , where  $t = 1, \dots, k-1$ .
- 4: Compute truncated initial local distribution vectors  $r^{(t)} = [q^{(t)}]_\epsilon$ ,  $t = 1, \dots, k-1$ .
- 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 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)  $I_x(q^{(t)}, 2^b) \geq \frac{\xi}{c_1(l+2)2^b}$ .
- 12:    return  $C = S_j(q^{(t)})$  and quit.
- 13:   **else**
- 14:    Return  $C = \emptyset$ .
- 15:   **end if**
- 16: **end for**

In Step 11 of Algorithm 1, condition (c) guarantees that

$$I_x(u) = \frac{q^{(t)}(u)}{d(u)} \geq \frac{\xi}{c_1(l+2)2^b} \quad (16)$$

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. 16

$$d(u) \leq \frac{1}{\xi} c_1(l+2)2^b q^{(t)}(u). \quad (17)$$

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 Eq. 15 and Eq. 17, we have

$$\begin{aligned} \mu(S_{j'}(q^{(t)}) \cap \bar{C}) &= \sum_{u \in S_{j'}(q^{(t)}) \cap \bar{C}} d(u) \leq \sum_{u \in S_{j'}(q^{(t)}) \cap \bar{C}} \frac{1}{\xi} c_1(l+2)2^b q^{(t)}(u) \\ &\leq \frac{1}{\xi} c_1(l+2)2^b (\chi_{\bar{C}}^T q^{(t)}) \leq \frac{\xi c_1(l+2)2^b}{\xi c_2(l+2)} \leq 2^{b-1}. \end{aligned} \quad (18)$$

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

$$\mu(S_j(q^{(t)}) \cap C) \geq \mu(S_{j'}(q^{(t)}) \cap C) \geq 2^{b-1}. \quad (19)$$

□

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

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  $V^{(i)}$  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  $r^{(t-1)} \otimes \dots \otimes r^{(t-k+1)}$  can be performed in time proportion to

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

Also, [23] shows that  $\mu(V^{(t)}) \leq 1/\epsilon$  for all  $t$ . Therefore, to compute  $r^{(t-1)} \otimes \dots \otimes r^{(t-k+1)}$  takes  $O(\mu(V^{(i)})^{(k-1)}) \leq O(1/\epsilon^{(k-1)})$  time. After that, the matrix vector product can be computed in

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

The truncation in Step 7 can be computed in time  $O(|V^{(i)}|)$ . Step 8 to Step 15 require sorting the vertices in  $|V^t|$  according to  $r^{(t)}$ , which takes time  $O(|V^{(i)}| \log |V^{(i)}|)$ . 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 Eq. 3, we can expand  $O(\frac{t_{\max}}{\epsilon^k})$  as follows

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

□

*Remark 1:* The major computation overhead of Algorithm 1 comes from Step 6. Note that  $O\left(t_{\max} \frac{2^{bk}}{\phi^{2k}} \log^{3k} m\right)$  is a strict upper-bound for considering extreme cases. While, due to the power law distribution in real networks, we may usually have  $|V^{(t)}| \leq \sqrt{\mu(V^{(i)})}$ . 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  $O\left(\frac{t_{\max} 2^b \log^4 m}{\phi^2}\right)$ . Considering the  $k = 3$  case, the time complexity of HOSPLOC is  $O\left(t_{\max} \frac{2^{3b}}{\phi^6} \log^9 m\right)$ . Without considering the impact from the other constants, we can see that similar as Nibble, HOSPLOC also runs in *polylogarithmic* time complexity with respect to the number of edges in the graph.

## 5 GENERALIZATIONS AND APPLICATIONS

In this section, we will introduce several generalizations and applications of our proposed HOSPLOC algorithm on bipartite and multipartite Networks.

## 5.1 Adversarial Learning on Bipartite Network

We now turn our attention to the problem of user behavior modeling on the adversarial networks. Given an adversarial 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 adversarial network should be included in the same cluster. For this reason, we choose 4-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 4-node loop, i.e.,  $u_1 \rightarrow a_1 \rightarrow u_2 \rightarrow a_2 \rightarrow u_1$ . In this problem, we consider the adversarial network as an undirected graph, and the adjacency tensor used in *HOSPLOC* algorithm is

$$T(i_1, i_2, i_3, i_4) = \begin{cases} 1 & \{i_1, i_2, i_3, i_4\} \text{ form a 4-nodes loop} \\ 0 & \text{Otherwise} \end{cases} \quad (20)$$

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 both similar users and the users' favorite advertiser campaigns.

## 5.2 Synthetic ID Detection on Multipartite Network

Now we will explain how to detect synthetic IDs on PII network by using our proposed *HOSPLOC* algorithm. PII network is a typical multipartite 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 [13], 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 PII would be shared by multiple identities, which may compose rich star-shaped structures. In this case, the adjacency tensor can be constructed as

$$T(i_1, i_2, \dots, i_k) = \begin{cases} 1 & \{i_1, i_2, \dots, i_k\} \text{ form a } k\text{-node star} \\ 0 & \text{Otherwise} \end{cases} \quad (21)$$

where  $\{i_1, i_2, \dots, i_k\} \in V_B$ . 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

Now, we demonstrate the performance of our proposed *HOSPLOC* algorithm in the sense of effectiveness, efficiency and parameter sensitivity. Moreover, we also present two interesting case studies on bipartite graph and multipartite 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.

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	Multipartite	375	519

Table 2: Statistics of the Networks.

- **Collaboration Network:** We use two collaboration networks from<sup>‡</sup>. 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)<sup>§</sup>, the nodes represent 2,833 airports, and the edges represent the U.S. flights in a one-month interval. Network (Oregon) [15] is a network of routers in Autonomous Systems inferred from Oregon route-views between March 31, 2001, and May 26, 2001. Network (Power)<sup>¶</sup> 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) [15] is a who-trust-whom online social network. Each node represents a user, and one edge exists if and only if when one user trust another user.
- **Review Network:** Network (Rating) [14] 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 4<sup>th</sup>-order transition tensor of the original graph, i.e., 300,000 vertices and 20M edges, requires too much memory.
- **Financial Network:** Network (PII) is a multipartite 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 [23]; (2) Nibble-PageRank [1]; (3) LS-OQC [25], and two global clustering algorithms, i.e., (1) NMF [10]; (2) TSC [5]. 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 the 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.

<sup>‡</sup><https://aminer.org/data>

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

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



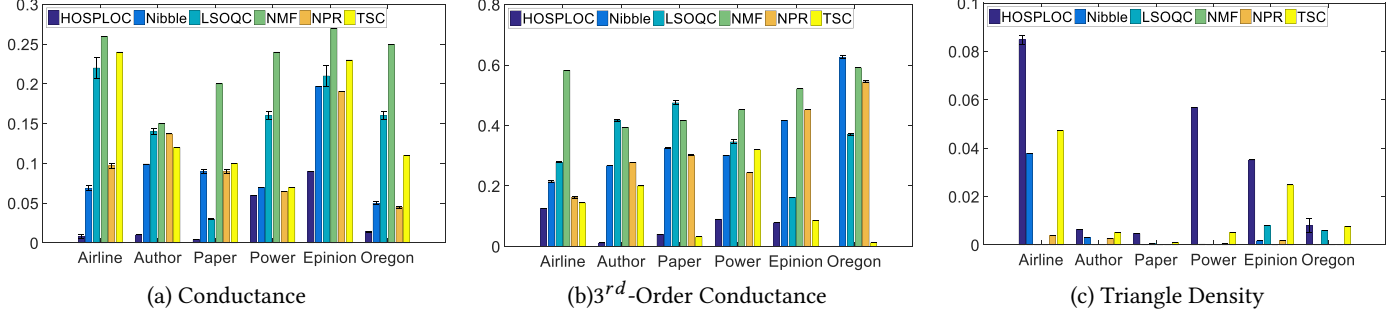


Figure 2: Effectiveness

6.2 Effectiveness Comparison

The effectiveness comparison results conducted on six real undirected graphs by the following three evaluation metrics are shown in Fig. 2. Among them, (1) **Conductance** [7] measures the general quality of a cut on graph, which quantitatively indicates the compactness of a cut; (2) the **3<sup>rd</sup>-Order Conductance** could be computed based on Eq. 10 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; (3) **Triangle Density** [7] computes the ratio of how rich the triangle is included in the returned cluster.

Moreover, to evaluate the convergence of local algorithms, we randomly select 30 vertices from one cluster on each testing graph and run all the local algorithms multiple times by treating each of these nodes as an initial vertex. In Fig. 2, the height of bars indicates the average value of evaluation metrics, and the error bars 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 3<sup>rd</sup>-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 3<sup>rd</sup>-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.

6.3 Efficiency Comparison

Here, we evaluate 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 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 Fig. 3 are the average values of multiple runs by using 30 different initial vertexes 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 of 1%. We observe that although *HOSPLOC* requires more time than Nibble in each run, the running time of *HOSPLOC* increases *polylogarithmically* with the size of the 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* is polynomial with respect to  $2^b$ , which is consistent with our time complexity analysis.

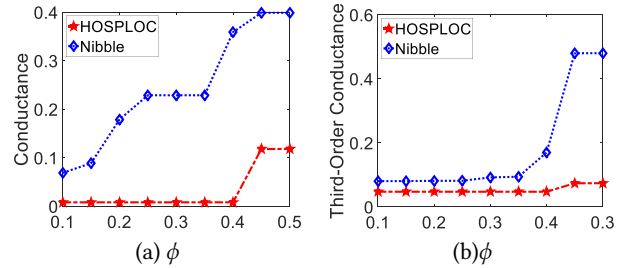


Figure 4: Parameter  $\phi$

6.4 Parameter Study

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 5000 vertices and 1% edge density. In the experiments, we evaluate the conductance and the 3<sup>rd</sup>-order conductance of the returned cut with different input parameter  $\phi$ . In Fig. 4, we have the 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* 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 3<sup>rd</sup>-order conductance of *HOSPLOC*'s cut are always smaller than Nibble's cut with different  $\phi$ .

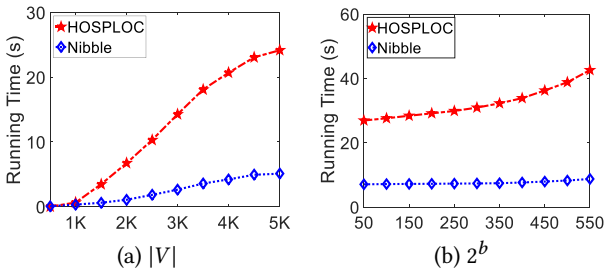
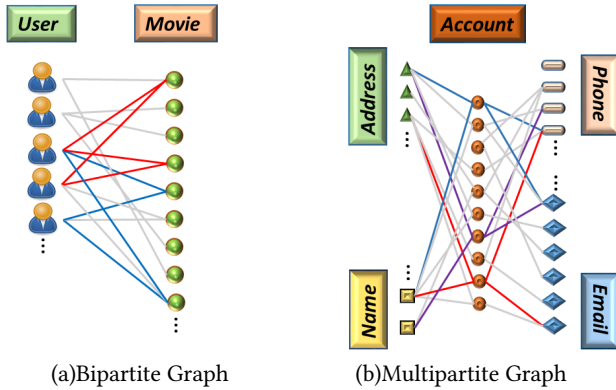


Figure 3: Efficiency Analysis





**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**

## 6.5 Case Study

In this subsection, we will consider more complex network structures and perform our proposed HOSPLOC algorithm on bipartite and multipartite 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 4-node loop based on Eq. 20. 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), 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 Multipartite 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. 21. Fig. 5(b) presents a subgraph of the returned cut by our proposed HOSPLOC algorithm. We can see that many PIIs are highly 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., 5-node star, on the given graph.

## 7 CONCLUSION

In this paper, we propose a local clustering framework, i.e., HOSPLOC, that gives users the flexibility to model any high-order network structures and returns a small high-order conductance cut

which largely preserves the *user-specified* network structures in the cut. 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 HOSPLOC algorithm and try to solve multiple real problems on bipartite and multipartite graphs, by exploring the useful high-order network connectivity patterns, such as loop, and star. Finally, the extensive empirical evaluations on a diverse set of networks demonstrate the effectiveness and scalability of our proposed HOSPLOC algorithm.

## REFERENCES

- [1] Reid A, Fan C, and Kevin L. 2006. Local graph partitioning using pagerank vectors. In *FOCS*. IEEE, 475–486.
- [2] Reid A, Shayan Oveis G, Yuval P, and Luca T. 2016. Almost Optimal Local Graph Clustering Using Evolving Sets. *JACM* 63, 2 (2016), 15.
- [3] Reid A and Yuval P. 2009. Finding sparse cuts locally using evolving sets. In *STOC*. ACM, 235–244.
- [4] SR A and SR D. 1988. Limit distribution of a high order Markov chain. *J R Stat Soc Series B Stat Methodol* (1988), 105–108.
- [5] Austin R B, David F G, and Jure L. 2015. Tensor spectral clustering for partitioning higher-order network structures. In *SDM*. SIAM, 118–126.
- [6] Austin R B, David F G, and Lek-Heng L. 2016. The Spacey Random Walk: A stochastic Process for Higher-Order Data. *arXiv preprint arXiv:1602.02102* (2016).
- [7] Béla Bollobás. 2013. *Modern graph theory*. Vol. 184. Springer Science & Business Media.
- [8] Flavio C, Ravi K, Prabhakar R, and Tamas S. 2012. Are web users really markovian?. In *WWW*. ACM, 609–618.
- [9] Kim-Kwang Raymond C. 2008. *Money laundering risks of prepaid stored value cards*. Australian Institute of Criminology.
- [10] Chris D, Tao L, and Michael I J. 2008. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *ICDM*. IEEE, 183–192.
- [11] David F G, Lek-Heng L, and Yongyang Y. 2015. Multilinear PageRank. *SIMAX* 36, 4 (2015), 1507–1541.
- [12] Shayan Oveis G and Luca T. 2012. Approximating the expansion profile and almost optimal local graph clustering. In *FOCS*. IEEE, 187–196.
- [13] Chris Joy H. 2007. Identity theft: Making the known unknowns known. *Harv. JL & Tech.* 21 (2007), 97.
- [14] F Maxwell H and Joseph A K. 2016. The movielens datasets: History and context. *TiIS* 5, 4 (2016), 19.
- [15] Jure L and Andrej K. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>. (June 2014).
- [16] László L and Miklós S. 1990. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *FOCS*. IEEE, 346–354.
- [17] László L and Miklós S. 1993. Random walks in a convex body and an improved volume algorithm. *Random structures & algorithms* 4, 4 (1993), 359–412.
- [18] Tom L and Satish R. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM* 46, 6 (1999), 787–832.
- [19] Wen L and Michael K N. 2014. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra* 62, 3 (2014), 362–385.
- [20] Adrian E R. 1985. A model for high-order Markov chains. *J R Stat Soc Series B Stat Methodol* (1985), 528–539.
- [21] Martin R, Alcides V E, Andrea L, Jevin D W, and Renaud L. 2014. Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications* 5 (2014).
- [22] Daniel A S and Shang-Hua T. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*. ACM, 81–90.
- [23] Daniel A S and Shang-Hua T. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SICOMP* 42, 1 (2013), 1–26.
- [24] Jiri S and Satu Elisa S. 2006. On the NP-completeness of some graph cluster measures. In *SOFSEM*. Springer, 530–537.
- [25] Charalampos T, Francesco B, Aristides G, Francesco G, and Maria T. 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *SIGKDD*. ACM, 104–112.
- [26] Jozef L T. 2008. Markov Chains: Models, Algorithms and Applications. *JASA* 103, 483 (2008), 1325–1325.
- [27] Tao W, Austin R B, and David F G. 2016. General tensor spectral co-clustering for higher-order data. In *NIPS*. 2559–2567.