Local Motif Clustering on Time-Evolving Graphs

Dongqi Fu
University of Illinois at Urbana-Champaign
dongqif2@illinois.edu

Dawei Zhou
University of Illinois at Urbana-Champaign
dzhou21@illinois.edu

Jingrui He
University of Illinois at Urbana-Champaign
jingrui@illinois.edu

ABSTRACT

Graph motifs are subgraph patterns that occur in complex networks, which are of key importance for gaining deep insights into the structure and functionality of the graph. Motif clustering aims at finding clusters consisting of dense motif patterns. It is commonly used in various application domains, ranging from social networks to collaboration networks, from market-basket analysis to neuroscience applications. More recently, local clustering techniques have been proposed for motif-aware clustering, which focuses on a small neighborhood of the input seed node instead of the entire graph. However, most of these techniques are designed for static graphs and may render sub-optimal results when applied to large time-evolving graphs. To bridge this gap, in this paper, we propose a novel framework, Local Motif Clustering on Time-Evolving Graphs (L-MEGA), which provides the evolution pattern of the local motif cluster in an effective and efficient way. The core of L-MEGA is approximately tracking the temporal evolution of the local motif cluster via novel techniques such as edge filtering, motif push operation, and incremental sweep cut. Furthermore, we theoretically analyze the efficiency and effectiveness of these techniques on time-evolving graphs. Finally, we evaluate the L-MEGA framework via extensive experiments on both synthetic and real-world temporal networks.

CCS CONCEPTS

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

KEYWORDS

Local Clustering, High-Order Structure, Time-Evolving Graph

ACM Reference Format:
Dongqi Fu, Dawei Zhou, and Jingrui He. 2020. Local Motif Clustering on Time-Evolving Graphs. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403081

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403081

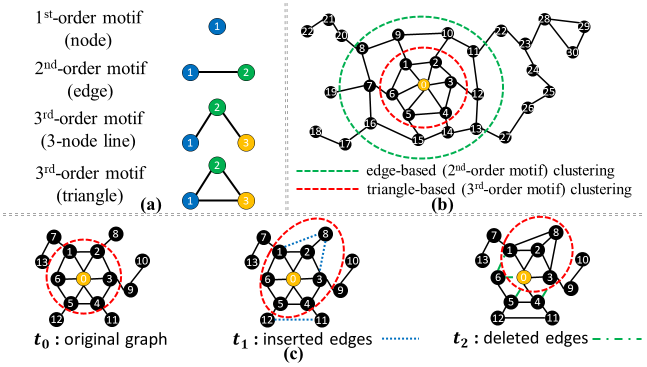


Figure 1: An overview of local motif clustering on time-evolving graphs. Part (a) illustrates the different order motif structures; Part (b) shows the difference between second-order and third-order local clustering on a network example given the yellow seed node; Part (c) shows the evolution pattern of the triangle-based local cluster (red dash line) on the time-evolving graphs for inserted edges and deleted edges.

1 INTRODUCTION

Local clustering aims to identify a qualified local cluster near a given seed node. Previously, most existing local clustering methods [1, 28] take the connectivity of low-order structures (i.e., edges) as the clustering metric. In other words, they largely ignore the connection of high-order subgraph patterns, i.e., motifs [23] (Fig. 1(a) illustrates a few motifs), which might be of key importance in real applications [18]. For example, triangle motifs may help identify frequent flight patterns in air traffic networks [26]; star motifs may indicate synthetic identities in social networks [12]; loop motifs may be associated with money laundering in financial networks [36]; user-specified complex protein-pair motifs can be helpful for better understanding the functional organization of the proteome in protein-protein interaction networks [31]. To address this problem, local motif clustering algorithms [33, 36] have been proposed to find a cluster with densely connected motifs in the neighborhood of the seed node. Fig. 1(b) illustrates the difference between the traditional edge-based local clustering method [28] and the emerging local motif clustering method [36], by setting the triangle as the target motif being preserved from cut, the local cluster identified by the local motif clustering method is much more densely connected by the triangle motifs.

However, most existing local motif clustering techniques [33, 36] are designed for the static setting. On time-evolving graphs, repeatedly applying these techniques at each time stamp from scratch would be computationally prohibitive. A closely related work is [35], where the authors extended local motif clustering to dynamic graphs by integrating the time-respecting information

into one snapshot. However, their approach cannot be used with regular motifs without the temporal information (e.g., a regular triangle), and it does not take into consideration the evolution pattern between time stamps of dynamic graphs.

To bridge this gap, in this paper, we propose a novel local motif clustering framework for time-evolving graphs named L-MEGA. Fig. 1(c) illustrates the triangle-based local cluster at each time stamp identified by L-MEGA as the graph evolves over time. Given a user-specified high-order motif, we formally define local motif clustering problems on time-evolving graphs and propose L-MEGA framework to solve the problem effectively and efficiently. The core of L-MEGA is approximately tracking the stationary distribution of high-order random walk [10] among time stamps by leveraging temporal information, then input the tracked stationary distribution into a vector-based sweep cut procedure [1] to get a local cluster with a small motif conductance score. L-MEGA consists of three parts, and we design speeding up strategies in each part. The first part is updating transition tensors for representing the transition probability for high-order random walk process [6], the second part is tracking the stationary probability distribution at each time stamp, and the third part is using the tracked stationary distribution to incrementally get a qualified local cluster. We give the theoretical efficiency and effectiveness analysis for each part and make experiments to demonstrate L-MEGA with self-ablation and other state-of-art algorithms on real-world temporal networks.

Our main contributions are summarized as follows.

- We define the local motif clustering problem on time-evolving graphs, we then propose a novel model, L-MEGA, to solve the dynamic local motif clustering problem effectively and efficiently.
- We theoretically analyze the effectiveness and efficiency of the proposed L-MEGA model and practically evaluate it on real-word temporal networks compared with self-ablation and other state-of-art algorithms. We also study the scalability analysis and parameter sensitivity of L-MEGA.

The rest of the paper is organized as follows. The problem of local motif clustering on time-evolving graphs is formally defined in Section 2. In Section 3, we introduce the preliminaries and the proposed model with the analysis on effectiveness and efficiency. Then we present the L-MEGA algorithm in Section 4. The experimental results on multiple real and synthetic data sets are presented in Section 5. Then we review the related work in Section 6 before concluding the paper in Section 7.

2 PROBLEM DEFINITION

The main symbols are summarized in Table 3 of Appendix A. We use lowercase letters (e.g., α) for scalars, bold lowercase letters for column vectors (e.g., $\boldsymbol{x}$), bold capital letters for matrices (e.g., $\boldsymbol{P}$), and parenthesized superscript to denote the time stamp (e.g., $G^{(t)}$). We index the elements in a matrix using a convention similar to Matlab, $P(i,:)$ is the i-th row of $\boldsymbol{P}$, etc. We use overlined and bold capital letters for tensors (e.g., $\bar{\boldsymbol{P}}$).

A time-evolving graph can be naturally presented as a collection of graph snapshots over time, i.e., $\widetilde{G} = \{G^{(0)}, G^{(1)}, \ldots, G^{(T)}\}$, where $G^{(t)} = (V^{(t)}, E^{(t)})$, $t = 1, \ldots, T$. We let $V^{(t)}$ and $E^{(t)}$ denote the sets of nodes and edges in $G^{(t)}$, $\boldsymbol{d}^{(t)} \in \mathbb{R}^{n^{(t)}}$ denote the degree vector of $G^{(t)}$, and $d^{(t)}(v)$ denote the specific degree of vertex $v \in V^{(t)}$. Following [30], we assume the number of nodes are fixed, i.e., an inserted (or deleted) node at time stamp t is regarded as an existed dangling node at all previous time stamps (or all future time stamps), such that the size of adjacency matrix $A^{(t)}$ of each snapshot $G^{(t)}$ are consistent over time. On top of that, we introduce the notion of updated edge set $\Delta E^{(t)} = \Delta E^{(t)}_{+} \cup \Delta E^{(t)}_{-}$ to simplify the description of our algorithm, where $\Delta E^{(t)}_{+} = \{E^{(t+1)} \setminus E^{(t)}\}$ and $\Delta E^{(t)}_{-} = \{E^{(t)} \setminus E^{(t+1)}\}$ are the sets of inserted edges and deleted edges at time stamp t to transform graph $G^{(t)}$ to $G^{(t+1)}$.

In the context of classic local motif clustering [33, 36], we aim to identify a local dense cluster C which sits near the given seed node v and preserves rich user-defined motifs $\mathbb{N}$. We let k denote the order of the user-defined motif $\mathbb{N}$, and k refers to the number of vertices involved in the motif $\mathbb{N}$. For example, an edge is a second-order motif, a three-node line and a triangle can be classified as the third-order motif. The quality of the identified motif cluster is measured by motif conductance [33, 36], small motif conductance indicates a good partition. For any local cluster C and user-defined motif $\mathbb{N}$, the corresponding motif conductance $\Phi(C, \mathbb{N})$ is defined as

$$\Phi(C, \mathbb{N}) = \frac{cut(C, \mathbb{N})}{min\{\mu(C, \mathbb{N}), \mu(\bar{C}, \mathbb{N})\}} \tag{1}$$

where $cut(C, \mathbb{N})$ denotes the number of motif structures broken due to the partition of the graph G into local cluster C and its complement $\bar{C}$, $\mu(C, \mathbb{N})$ and $\mu(\bar{C}, \mathbb{N})$ denote the number of motif structures $\mathbb{N}$ within C and $\bar{C}$, respectively. Here we generalize the local motif clustering problem to the dynamic setting. Given a time-evolving graph $\widetilde{G}$, our goal is minimizing motif conductance to obtain a good local cluster $C^{(t)}$ over time.

$$\min_{C^{(1)}, \ldots, C^{(T)}} \sum_{t=1}^{T} \Phi(C^{(t)}, \mathbb{N}) \tag{2}$$

With above notations, we formally define our problem as follows:

Problem. *Local Motif Clustering on Time-Evolving Graphs*

Input: *(i) a user-defined motif $\mathbb{N}$, (ii) a sequence of snapshots of the time-evolving graph $\widetilde{G} = \{G^{(0)}, G^{(1)}, \ldots, G^{(T)}\}$, (iii) the seed node v, and (iv) the motif conductance upper bound ϕ.*

Output: *a local cluster $C^{(t)}$ near seed node v such that $\Phi(C^{(t)}, \mathbb{N}) \leq \phi$ at each time stamp $t \in \{1, 2, \ldots, T\}$.*

3 PROPOSED MODEL

In this section, we introduce our proposed L-MEGA algorithm for the problem local motif clustering on time-evolving graphs. We start with the background and preliminaries on the local clustering (Subsection 3.1) and discuss the technical details regarding how we model the graph evolution over time (Subsections 3.2 to 3.4). We then discuss how we identify the local cluster for the next time stamp (Subsection 3.5). Due to space limit, we leave the proof of all lemmas and theorems in Appendix B.

3.1 Preliminaries

PageRank and Multilinear PageRank. Given a random walk on the graph, the PageRank framework [22] models a stochastic process with a unique stationary distribution, which is the PageRank

vector solving the following linear system:

$$\boldsymbol{x} = \alpha \boldsymbol{P}\boldsymbol{x} + (1-\alpha)\boldsymbol{u} \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the PageRank vector, $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is a stochastic column matrix encoding the transition probability for the random walk, and $\boldsymbol{u} \in \mathbb{R}^n$ is the stochastic vector.

Multilinear PageRank framework [10] is proposed based on the spacey random walk [6] which is a high-order random walk developed on the high-order Markov Chain stochastic process. For example, in the second-order Markov Chain, the next state of the discrete time stochastic process only depends on the past two states, the probability is expressed $Prob(S_{t+1}|S_t, S_{t-1})$. With the rank-1 approximation [16] of high-order Markov Chain, authors in [10] encode $(k-1)^{th}$-order Markov Chain into a k^{th}-order probability transition tensor, and then propose the multilinear PageRank. For example, modelling third-order data, the stationary distribution (i.e., multilinear PageRank vector) solves the following equation:

$$\boldsymbol{x} = \alpha \bar{\boldsymbol{P}}(\boldsymbol{x} \otimes \boldsymbol{x}) + (1-\alpha)\boldsymbol{u} \tag{4}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the multilinear PageRank vector, $\otimes$ denotes the Kronecker product, the third-order transition tensor $\bar{\boldsymbol{P}} \in \mathbb{R}^{n\times n\times n}$ encodes the transition probability for second-order Markov Chain (random walk), and $\boldsymbol{u} \in \mathbb{R}^n$ is the stochastic vector.

Note that, the order of data is different from the order of Markov Chain (random walk). In general, k^{th}-order motif data correspond to the $(k-1)^{th}$-order Markov chain. Encoding k^{th}-order motif data into the k^{th}-order probability transition tensor (i.e., $(k-1)^{th}$-order Markov Chain) will be introduced in Subsection 3.2. Moreover, for the tensor-vector multiplication operation, we define the one-mode unfolding operation of tensors. For example, a three-mode (or third-order) tensor $\bar{\boldsymbol{P}} \in \mathbb{R}^{n\times n\times n}$ is unfolded as the matrix $\boldsymbol{P} = [\bar{P}(:,:,1), \bar{P}(:,:,2), \ldots, \bar{P}(:,:,n)]$, where $\boldsymbol{P} \in \mathbb{R}^{n\times n^2}$ and $\bar{P}(:,:,i)$ is the i-th $n \times n$ block of the tensor $\bar{\boldsymbol{P}}$.

Approximated PageRank for Local Clustering. The problem of identifying a local cluster with minimum conductance has been proven to be NP-complete [27]. To mitigate the intractable complexity, a series of PageRank-based approximation methods [1, 28] have been developed in the past. The key idea behinds these approaches is to model the stochastic process of random walks on graphs and then conduct vector-based sweep cut procedure on the obtained PageRank vector to identify local clusters.

Solving the PageRank vector (i.e., stationary distribution) through the power iteration method [22] is time-consuming, which iterates Eq. 3 until convergence requiring time complexity $O(m)$, where m is the number of edges of the graph. Instead, using the approximated PageRank vector, PageRank-Nibble [1] could obtain a local cluster with conductance ϕ in time complexity $O(2^b \log^3 m/\phi^2)$, where b is the constant parameter. First, PageRank-Nibble [1] solves the approximated PageRank vector through the push operation. Push operation maintains a residual vector to record the divergence between the approximated PageRank vector and the PageRank vector. After sufficient iterations of moving probability mass to the approximated PageRank vector from the residual vector, push operation obtains the converged approximated stationary distribution as the clustering indicator vector. Second, PageRank-Nibble feeds the approximated PageRank into the sweep cut procedure to obtain the qualified local cluster satisfying statistical conditions.

More recently, a surge of research interest on motif clustering has been observed in the data mining community [4, 5, 32, 33, 36]. To indicate a local cluster preserving rich user-defined high-order motifs, (approximated) multilinear PageRank vector [10] has been proven to be effective in many high-order clustering applications [4, 32, 36] as the clustering indicator vector.

Inspired by local motif clustering method [36], we extend the statistical conditions of sweep cut [1] to high-order and dynamic settings, such that we can identify a qualified motif cluster on the approximated multilinear PageRank vector at each time stamp. For notation simplicity, we ignore the superscript of time stamps below.

(C.1) $\Phi(S_j(\boldsymbol{x}), \mathbb{N}) \leq \phi$ /*motif conductance check*/
(C.2) $\lambda_j(\boldsymbol{x}) \leq (2/3)\mu(V)$ /*upper-bound volume check*/
(C.3) $2^b \leq \lambda_j(\boldsymbol{x})$ /*lower-bound volume check*/

Condition (C.1) guarantees that the returned local cluster $C = S_j(\boldsymbol{x})$ has a low motif conductance. $S_j(\boldsymbol{x})$ denotes the sweep cut (or sweep set) of $\boldsymbol{x}$ where $\boldsymbol{x}$ is the approximated multilinear PageRank vector and $S_j(\boldsymbol{x}) = \{\pi(1), \ldots, \pi(j)\}$ is the set of top j vertices v that maximize $x(v)/d(v)$, where $\boldsymbol{d}$ is the degree vector, π is the permutation of nodes as $\frac{x(\pi(i))}{d(\pi(i))} \geq \frac{x(\pi(i+1))}{d(\pi(i+1))}$.

Condition (C.2) and Condition (C.3) limit the volume of the returned local cluster $S_j(\boldsymbol{x})$. The volume is defined as the summation of vertex degrees, $\lambda_j(\boldsymbol{x}) = \sum_{v\in S_j(\boldsymbol{x})} d(v)$ and $\mu(V) = \sum_{v\in V} d(v)$. $b \in [1, \log_2 m]$ is the parameter that controls the lower bound of volume, and m is the number of edges.

During the sweep cut procedure, j starts from 1 to the number of nodes, if a sweep cut $S_j(\boldsymbol{x})$ satisfies the conditions (C.1) to (C.3), then we suppose $S_j(\boldsymbol{x})$ is a qualified motif-preserving local cluster.

3.2 Time-Evolving Motif Representation

Nonzero entries in the adjacency matrix indicate the existence of edges (second-order motif), and the probability transition matrix derived from the adjacency matrix provides matrix representation of the first-order Markov Chain stochastic process. To explore high-order motif data, we follow the indicator and transition tensor to represent the high-order motif existence and the transition probability of high-order Markov Chain [4, 32, 36]. Furthermore, we generalize the idea to the dynamic setting and propose the following *dynamic indicator tensor* and *dynamic transition tensor*.

Definition 3.1 (Dynamic Indicator Tensor). At each time stamp t, given an undirected and unweighted graph $G^{(t)} = (V^{(t)}, E^{(t)})$, and the user-defined k^{th}-order structure $\mathbb{N}$, the k-mode dynamic indicator tensor $\bar{\boldsymbol{I}}^{(t)}$ is defined as follows.

$$\bar{I}^{(t)}(v_1, v_2, \ldots, v_k) = \begin{cases} 1 & \{v_1, v_2, \ldots, v_k\} \subseteq V^{(t)} \text{ and form } \mathbb{N} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Definition 3.2 (Dynamic Transition Tensor). At each time stamp t, given an undirected and unweighted graph $G^{(t)} = (V^{(t)}, E^{(t)})$, the k^{th}-order motif $\mathbb{N}$, and the k-mode dynamic indicator tensor $\bar{\boldsymbol{I}}^{(t)}$, the corresponding k-mode dynamic transition tensor $\bar{\boldsymbol{P}}^{(t)}$ can be computed as follows.

$$\bar{P}^{(t)}(v_1, v_2, \ldots, v_k) = \frac{\bar{I}^{(t)}(v_1, v_2, \ldots, v_k)}{\sum_{v_1=1}^{n^{(t)}} \bar{I}^{(t)}(v_1, v_2, \ldots, v_k)} \tag{6}$$

At each time stamp t, the dynamic transition tensor $\bar{\boldsymbol{P}}^{(t)}$ encodes the transition probability with respect to k^{th}-order motif $\mathbb{N}$ based on $(k-1)^{th}$-order Markov Chain, such that $\sum_{v_1=1}^{n^{(t)}} \bar{P}^{(t)}(v_1, v_2, \ldots, v_k) = 1$ and $\bar{P}^{(t)}(v_1, v_2, \ldots, v_k) = Prob(S_{t+1} = v_1 | S_t = v_2, \ldots, S_{t-k+2} = v_k)$ [16], where S_t denotes a discrete time stochastic process on the state space. With the k-mode transition tensor $\bar{\boldsymbol{P}}^{(t)}$ modelling k^{th}-order motif structure, at time t, the multilinear PageRank [10] vector $\boldsymbol{x}^{(t)} \in \mathbb{R}^{n^{(t)}}$ satisfies the following equation:

$$\boldsymbol{x}^{(t)} = \alpha \bar{\boldsymbol{P}}^{(t)} \underbrace{(\boldsymbol{x}^{(t)} \otimes \ldots \otimes \boldsymbol{x}^{(t)})}_{(k-1 \text{ terms})} + (1-\alpha)\boldsymbol{u} \tag{7}$$

where $\otimes$ denotes the Kronecker product, and $\boldsymbol{u} \in \mathbb{R}^{n^{(t)}}$ is the stochastic vector inherited from last time stamp $t-1$. Stochastic vector $\boldsymbol{u}$ encodes a probability distribution over $n^{(t)}$ nodes, where $\sum_{i=1}^{n^{(t)}} u(i) = 1$ and its element associated with the seed node v_{seed} has a large personalized value, e.g., 1.

Push operation [1] is an efficient method for solving the approximated PageRank vector for local clustering problems. It is widely adopted for tracking the PageRank vector evolution on dynamic graphs [21, 34]: first, the approximated PageRank vector and the corresponding residual vector from the last time stamp are usually used as the initial vectors for the current time stamp; then an iterative procedure of reducing the probability distribution of the initial residual vector is performed till convergence.

We extend the push operation to obtain the multilinear PageRank vector (Subsection 3.4). Here, we first define the residual vector $\boldsymbol{r}^{(t)}$ of the multilinear PageRank vector $\boldsymbol{x}^{(t)}$ at time stamp t as follows.

$$\boldsymbol{r}^{(t)} = \alpha \bar{\boldsymbol{P}}^{(t)} \underbrace{(\boldsymbol{x}^{(t)} \otimes \ldots \otimes \boldsymbol{x}^{(t)})}_{(k-1 \text{ terms})} + (1-\alpha)\boldsymbol{u} - \boldsymbol{x}^{(t)} \tag{8}$$

Dynamic graphs change over time due to updated edges. With the vectors $\boldsymbol{x}^{(t)}$ and $\boldsymbol{r}^{(t)}$, tensor $\bar{\boldsymbol{P}}^{(t)}$, and local cluster $C^{(t)}$ from time stamp t, we aim to efficiently obtain $\boldsymbol{x}^{(t+1)}$ to feed into the sweep cut procedure (i.e., Conditions (C.1) to (C.3)), in order to get local cluster $C^{(t+1)}$. To obtain multilinear PageRank vector $\boldsymbol{x}^{(t+1)}$ for time stamp $t+1$, we need to solve the following two problems efficiently, which will be elaborated in the next two subsections.

- First, transition tensor $\bar{\boldsymbol{P}}^{(t)}$ changes due to inserted (or deleted) edges. Instead of building a new tensor $\bar{\boldsymbol{P}}^{(t+1)}$ from the scratch, we need to leverage the change in structural information between time stamps to update the transition tensor $\bar{\boldsymbol{P}}^{(t)}$ efficiently (Subsection 3.3).
- Second, multilinear PageRank vector $\boldsymbol{x}^{(t)}$ changes with the new tensor $\bar{\boldsymbol{P}}^{(t+1)}$. We need to track $\boldsymbol{x}^{(t+1)}$ and $\boldsymbol{r}^{(t+1)}$ with the updates in the transition tensor $\bar{\boldsymbol{P}}^{(t+1)}$ (Subsection 3.4).

3.3 Transition Tensor Update via Edge Filtering

To obtain the transition tensor $\bar{\boldsymbol{P}}^{(t+1)}$, we start with $\bar{\boldsymbol{P}}^{(t)}$ from the previous time stamp, and update it based on the updated edge set $\Delta E^{(t)}$. Furthermore, we propose an *edge filtering model* to identify the subset of edges in $\Delta E^{(t)}$ that have little or no impacts on the resulting local cluster, which further speeds up the tensor computation. For example, some updated edges (inserted or deleted in the current time stamp) are "far-away" from the seed node. Therefore, their appearance (or disappearance) will not affect the local cluster identified at the last time stamp. The proposed edge filtering model is able to identify and filter these edges without affecting the generated local cluster at the current time stamp. To this end, we define the "far-away" updated edges as follows.

Definition 3.3 ("Far-away" Updated Edge). At time t, given a k^{th}-order motif $\mathbb{N}$, a multilinear PageRank vector $\boldsymbol{x}^{(t)}$, a degree vector $\boldsymbol{d}^{(t)}$ and an identified local cluster $C^{(t)} = S_j(\boldsymbol{x}^{(t)})$, an updated edge $e = (v_1, v_2)$ is "far-away" if it satisfies the following conditions:

$$\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)} < \frac{x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))},\ \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)} < \frac{x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))} \tag{9}$$

$$\frac{\gamma + x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))} < \frac{x^{(t)}(\pi(j))}{d^{(t)}(\pi(j))} \tag{10}$$

$$D_{ist}(v_1, C^{(t)}) > k-1,\ D_{ist}(v_2, C^{(t)}) > k-1 \tag{11}$$

where π is the permutation that follows $\frac{x^{(t)}(\pi(i))}{d^{(t)}(\pi(i))} \geq \frac{x^{(t)}(\pi(i+1))}{d^{(t)}(\pi(i+1))}$, as stated in condition (C.1); the local cluster $C^{(t)}$ is formed by vertices $S_j(\boldsymbol{x}^{(t)}) = \{\pi(1), \ldots, \pi(j)\}$ from the sweep cut; $D_{ist}(v, C^{(t)})$ denotes the shortest distance from the node v to reach any node within the local cluster $C^{(t)}$; and γ denotes the largest probability mass increment that updated edge $e = (v_1, v_2)$ adds to node $\pi(j+1)$, which can potentially change the ranking of node $\pi(j+1)$ at time stamp $t+1$. Depending on whether the updated edge $e = (v_1, v_2)$ is an inserted edge or a deleted edge, it can be shown that γ has the following values:

$$\gamma = \begin{cases} \frac{x^{(t)}(v_1) + x^{(t)}(v_2)}{\min(d^{(t)}(v_1),\ d^{(t)}(v_2))} & \text{if } (v_1, v_2) \text{ is an inserted edge} \\ \frac{x^{(t)}(v_1)}{d^{(t)}(v_1)-1} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)-1} & \text{if } (v_1, v_2) \text{ is a deleted edge} \end{cases} \tag{12}$$

We provide Lemma 3.4 which implies that even considering one "far-away" updated edge, the ranking of the first j entries $\{\pi(1), \ldots, \pi(j)\}$ of permutation π at time t remains still at time $t+1$; and sweep cut procedure will still terminate at the same index j at time $t+1$ to return sweep set $S_j(\boldsymbol{x}^{(t+1)})$ as the local cluster $C^{(t+1)}$. Therefore, one "far-away" updated edge is neglectable.

Lemma 3.4. *If updated edge $e = (v_1, v_2)$ is a "far-away" updated edge, $\boldsymbol{x}^{(t)}$ is the multilinear PageRank vector at time t, $\boldsymbol{x}^{(t+1)}$ is the approximated multilinear PageRank vector after sufficient spacey random walk steps starting from vector $\boldsymbol{x}^{(t)}$ on the new edge set $E^{(t+1)} = \{E^{(t)} \cup e\}$, then $S_j(\boldsymbol{x}^{(t)}) = S_j(\boldsymbol{x}^{(t+1)})$; and if the local cluster $C^{(t)} = S_j(\boldsymbol{x}^{(t)})$ is returned by sweep cut procedure through conditions (C.1) to (C.3) regrading the parameter set $B = \{\phi, b\}$, $\mu(C^{(t)}, \mathbb{N}) < \mu(\bar{C}^{(t)}, \mathbb{N})$ when the updated edge e is inserted (or $\mu(C^{(t)}, \mathbb{N}) > \mu(\bar{C}^{(t)}, \mathbb{N})$ when the updated edge e is deleted), then sweep cut procedure will still terminate at the same j index to return local cluster $C^{(t+1)} = S_j(\boldsymbol{x}^{(t+1)})$ at time $t+1$ regarding the same parameter set B.*

Based on the above analysis, we are now ready to present the main theorem with respect to the properties of multiple "far-away" updated edges.

Theorem 3.5. *Let Γ denote the summation of probability mass increments of sampled "far-away" inserted (or deleted) edges $\{e_1, \ldots, e_m\}$, $\Gamma = \sum_{i=1}^{m} \gamma_i$, where γ_i of e_i is obtained by Eq. 12. If*

$$\frac{\Gamma + x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))} < \frac{x^{(t)}(\pi(j))}{d^{(t)}(\pi(j))} \tag{13}$$

and $\mu(C^{(t)}, \mathbb{N}) < \mu(\bar{C}^{(t)}, \mathbb{N})$ (or $\mu(C^{(t)}, \mathbb{N}) > \mu(\bar{C}^{(t)}, \mathbb{N})$), then $C^{(t+1)}$ for time stamp $t+1$ is the same as $C^{(t)}$.

According to Theorem 3.5, "far-away" updated edges can be safely filtered before updating transition tensor $\bar{\boldsymbol{P}}^{(t)}$. This greatly saves the running time, as shown below.

Claim 1. *For a k^{th}-order motif $\mathbb{N}$ ($k \geq 3$), the time complexity of updating transition tensor $\bar{\boldsymbol{P}}^{(t)}$ in terms of one single updated edge is $O(\frac{k(k-1)}{2} \cdot d_{max}^{k-2})$, where d_{max} is the max degree of the graph $G^{(t)}$.*

Let n and m denote the numbers of nodes and edges of $G^{(t)}$, respectively. Based on Claim 1, in the worst case (node degree $d_{max} \approx n$), the time complexity of updating transition tensor $\bar{\boldsymbol{P}}^{(t)}$ of one updated edge is $O(n^{k-2})$. However, the edge filtering operations Eq. 9 and 10 cost $O(1)$, and Eq. 11 costs $O(n+m\log n)$ [8]. Therefore, our proposed edge filtering model identifies a "far-away" updated edge in $O(n + m\log n)$ time instead of taking it into consideration for updating the transition tenor in $O(n^{k-2})$ time.

We summarize the proposed edge filtering model within the updating process of the transition tensor in Alg. 1. Step 1 separates the updated edges into inserted edges and deleted edges; Step 2 checks whether these edges are "far-away" edges according to Theorem 3.5; and Steps 3-5 (or Steps 7-9) filter "far-away" inserted (or deleted) edges before updating the transition tensor.

Edge filtering model depends on the multilinear PageRank vector (i.e., stationary distribution). As we solve the approximated multilinear PageRank vector at each time stamp, therefore, after the long-term updating transition tensor via edge filtering model based on the approximated multilinear PageRank vector, there may be accumulated errors in the proposed L-MEGA framework. To ensure the accuracy of the transition tensor, we need to construct the transition tensor from the scratch after certain time stamps.

3.4 Motif Push Operation

With the updated transition tensor $\bar{\boldsymbol{P}}^{(t+1)}$, we also need to update the multilinear PageRank vector $\boldsymbol{x}^{(t+1)}$ accordingly. To this end, we initialize $\boldsymbol{x}^{(t+1)}$ with $\boldsymbol{x}^{(t)}$, and compute the initial residual vector $\boldsymbol{r}^{(t+1)}$ through Eq. 7 and 8 as follows.

$$\boldsymbol{r}^{(t+1)} = \boldsymbol{r}^{(t)} + \alpha(\bar{\boldsymbol{P}}^{(t+1)} - \bar{\boldsymbol{P}}^{(t)})(\boldsymbol{x}^{(t)} \otimes \ldots \otimes \boldsymbol{x}^{(t)}) \tag{14}$$

Next, we aim to gradually decrease the residual vector in order to obtain the updated multilinear PageRank vector. To this end, given a user-defined high-order motif, we propose an iterative *motif push operation* to track the multilinear PageRank vector. We use subscript as iteration index. Therefore, $\boldsymbol{x}_0^{(t+1)} = \boldsymbol{x}^{(t)}$, and $\boldsymbol{r}_0^{(t+1)}$ is set to Eq. 14. Motif push operation repeatedly updates the multilinear PageRank vector $\boldsymbol{x}_{v+1}^{(t+1)}$ and its residual vector $\boldsymbol{r}_{v+1}^{(t+1)}$ as follows.

$$\boldsymbol{x}_{v+1}^{(t+1)} = \boldsymbol{x}_v^{(t+1)} + r_v^{(t+1)}(i)\boldsymbol{e} \tag{15}$$

Algorithm 1 Transition Tensor Update with Edge Filtering

Input:
local cluster $C^{(t)}$, transition tensor $\bar{\boldsymbol{P}}^{(t)}$, updated edge set $\Delta E^{(t)}$.
Output:
transition tensor $\bar{\boldsymbol{P}}^{(t+1)}$.

1: Separate updated edge set $\Delta E^{(t)}$ into inserted edge set ΔE_+ and deleted edge set ΔE_-.
2: **if** $\mu(C^{(t)}, \mathbb{N}) < \mu(\bar{C}^{(t)}, \mathbb{N})$ **then**
3: **for** edge e in ΔE_+ **do**
4: Sample "far-away" edge set $E_F = \{e_1, \ldots, e_m\}$ which satisfies the conditions of Eq. 9, 10, 11, and 13.
5: **end for**
6: **else**
7: **for** edge e in ΔE_- **do**
8: Sample "far-away" edge set $E_F = \{e_1, \ldots, e_m\}$ which satisfies the conditions of Eq. 9, 10, 11, and 13.
9: **end for**
10: Update $\bar{\boldsymbol{P}}^{(t+1)}$ from $\bar{\boldsymbol{P}}^{(t)}$ based on edge set $(\Delta E^{(t)} \setminus E_F)$.
11: Save filtered edge set E_F for next time stamp updates.
12: **end if**

$$\boldsymbol{r}_{v+1}^{(t+1)} = \boldsymbol{r}_v^{(t+1)} - r_v^{(t+1)}(i)\boldsymbol{e} + \alpha r_v^{(t+1)}(i)\bar{\boldsymbol{P}}^{(t+1)}(\boldsymbol{e} \otimes \boldsymbol{x}_{v-1}^{(t+1)} \ldots \otimes \boldsymbol{x}_{v-k+2}^{(t+1)}) \tag{16}$$

where $r_v^{(t+1)}(i)$ is the largest entry of $\boldsymbol{r}_v^{(t+1)}$, $\boldsymbol{e} \in \mathbb{R}^{n^{(t)}}$ is the basis vector with the i-th entry equals to 1, and the other entries equal to 0.

Intuitively, motif push operation iteratively extracts the largest probability mass $r_v^{(t+1)}(i)$ from the residual vector $\boldsymbol{r}_v^{(t+1)}$, and adds it to the approximated multilinear PageRank vector $\boldsymbol{x}_v^{(t+1)}$, such that the residual vector $\boldsymbol{r}^{(t+1)} \to \boldsymbol{0}$ after sufficient iterations. During each iteration of the motif push operation, the effect of increasing the i-th entry of the vector $\boldsymbol{x}_v^{(t+1)}$ should also be considered, as it produces the additive divergence between $\boldsymbol{r}_{v+1}^{(t+1)}$ and $\boldsymbol{x}_{v+1}^{(t+1)}$ according to Eq. 7. Therefore, we adopt the lazy updating rule of HOSPLOC [36] (i.e., Eq. 17) to measure such additive divergence, which works as follows.

$$\boldsymbol{x}_{v+1}^{(t+1)} = \boldsymbol{P}^{(t+1)}(\boldsymbol{x}_v^{(t+1)} \otimes \boldsymbol{x}_{v-1}^{(t+1)} \otimes \ldots \otimes \boldsymbol{x}_{v-k+2}^{(t+1)}) \tag{17}$$

Adding vector $r_v^{(t+1)}(i)\boldsymbol{e}$ on vector $\boldsymbol{x}_v^{(t+1)}$ (i.e., Eq. 15) produces additive divergence $\alpha r_v^{(t+1)}(i)\boldsymbol{P}^{(t+1)}(\boldsymbol{e} \otimes \boldsymbol{x}_{v-1}^{(t+1)} \ldots \otimes \boldsymbol{x}_{v-k+2}^{(t+1)})$ derived from Eq. 7 and 17, which should be added back to residual vector $\boldsymbol{r}_v^{(t+1)}$ as the third item of the right-hand-side of Eq. 16.

Motif push operation terminates until $\boldsymbol{r}^{(t+1)} = \boldsymbol{0}$ at time stamp $t+1$. In practice, we define a threshold ϵ such that at the v-th iteration, if the largest entry $r_v^{(t+1)}(i) \leq \epsilon$, then motif push operation terminates. The following theorem shows the efficiency of the proposed push operation with such a threshold.

Theorem 3.6 (Efficiency of motif push operation). *At time stamp $t+1$, given a motif push operation threshold ϵ, the motif push operation will terminate in $\frac{1}{(1-\alpha)\epsilon}$ number of iterations with $\|\boldsymbol{r}^{(t+1)}\|_\infty \leq \epsilon$, and each iteration has polylogarithmic time complexity $O(\frac{1}{\xi^k})$ with respect to the number of edges, where $\xi \propto \frac{1}{\log_2(\mu(V^{(t+1)}))}$, $\mu(V^{(t+1)})$ is the volume of graph $G^{(t+1)}$, k is the order of motif $\mathbb{N}$.*

3.5 Incremental Sweep Cut

After we have obtained the approximated multilinear PageRank vector $\boldsymbol{x}^{(t+1)}$ for time stamp $t+1$, we then use the sweep cut procedure to compute permutation π, which leads to sweep set $S_j(\boldsymbol{x}^{(t+1)})$ as local cluster $C^{(t+1)}$. However, in the sweep cut procedure, checking condition (C.1) is costly: each iteration of the sweep cut algorithm requires $O(n^2)$ for computing $\Phi(S_j(\boldsymbol{x}^{(t+1)}), \mathbb{N})$ with the support of the motif weighted matrix from [5, 33], where n is the number of nodes of graph $G^{(t+1)}$. Therefore, we aim to leverage the temporal information for identifying shared computation between two consecutive time stamps. In this way, we can reduce the number of iterations required in the sweep cut procedure. To this end, we first introduce some properties of the local cluster as follows.

Claim 2. *With the local cluster $C^{(t)}$ identified by the sweep cut procedure through conditions (C.1) to (C.3), if $\mu(C^{(t)}, \mathbb{N}) < \mu(\bar{C}^{(t)}, \mathbb{N})$, and updated edge set $\Delta E^{(t)}$ only contains inserted edges on $\bar{C}^{(t)}$ after edge filtering, then $|C^{(t+1)}| \geq |C^{(t)}|$.*

From Claim 2, when the updated edge set only consists of inserted edges on the complement of the local cluster identified by the sweep cut procedure, and the motif volume of the local cluster is smaller than its complement, then the local cluster can only expand or remain the same at the next time stamp. Hence, it is safe to conclude that the first q shared nodes of $\pi^{(t)}$ and $\pi^{(t+1)}$ are included in the same local cluster.

Based on the above analysis, we propose incremental sweep cut to speed up the computation. With $S_j(\boldsymbol{x}^{(t+1)})$, we need to determine the value of j to return the local cluster $C^{(t+1)}$ in the sweep cut procedure. Here we use $j^{(t)}$ and $j^{(t+1)}$ to denote the values of j at time stamps t and $t+1$ respectively. In the original sweep cut procedure, $j^{(t+1)}$ increases from 1. However, in incremental sweep cut, if the condition of Claim 2 is satisfied, we first compare permutation $\pi^{(t)}$ and $\pi^{(t+1)}$ to find the largest index q such that

$$\pi^{(t)}(1) = \pi^{(t+1)}(1)\,,\ \ldots\ ,\ \pi^{(t)}(q) = \pi^{(t+1)}(q)\,,\ \text{and}\ q \leq j^{(t)} \quad (18)$$

then $j^{(t+1)}$ would start from $q+1$ instead of 1. In this way, incremental sweep cut reduces q iterations where each iteration costs polynomial time complexity $O(n^2)$. However, incremental sweep cut cannot deal with deletion updates. Because deleted edges may keep some shared ranking but the local cluster shrinks, which means the length of the shared ranking is larger than the size of the optimal local cluster. If $j^{(t+1)}$ still iterates from $q+1$, we may miss the denser local cluster.

4 L-MEGA ALGORITHM

In this section, we introduce the L-MEGA algorithm in Alg. 2. It receives as input the high-order motif $\mathbb{N}$, the seed node v_{seed}, and the updated edge set ΔE. Notice that some inputs (e.g., $\boldsymbol{x}^{(0)}$, $\bar{\boldsymbol{P}}^{(0)}$, $C^{(0)}$) could be obtained from existing static algorithms, e.g., [36]. Then it incrementally tracks the local cluster as the graph evolves over time, and outputs the local clusters associated with each time stamp. First, Step 2 updates the transition tensor $\bar{\boldsymbol{P}}^{(t+1)}$ with the edge filtering model (Alg.1), which paves the way for calculating the initial approximate multilinear PageRank vector $\boldsymbol{x}_0^{(t+1)}$ and residual vector $\boldsymbol{r}_0^{(t+1)}$ in Step 3. Then Steps 6-8 call motif push operation

Algorithm 2 Local Motif Clustering on Time-Evolving Graphs (L-MEGA)

Input:
the k^{th}-order motif $\mathbb{N}$, the transition tensor $\bar{\boldsymbol{P}}^{(0)}$, the initial multilinear PageRank vector $\boldsymbol{x}^{(0)}$ and initial residual $\boldsymbol{r}^{(0)}$, the initial local cluster $C^{(0)}$, the updated edge set $\{\Delta E^{(0)}, \ldots, \Delta E^{(T-1)}\}$, the motif push operation threshold ϵ, the motif conductance upperbound ϕ, and parameters b, c_1.
Output:
clusters $C^{(1)}, \ldots, C^{(T)}$.

```
1:  for t = 0 : T − 1 do
2:    Update transition tensor P̄^(t+1) using Alg. 1.
3:    Initialize x_0^(t+1) = x^(t), and r_0^(t+1) through Eq. 14.
4:    Initialize x_v^(t+1) = x_0^(t+1), r_v^(t+1) = r_0^(t+1), v = 1, ..., k.
5:    Set v = k.
6:    while the largest entry r_v^(t+1)(i) of r_v^(t+1) ≥ ε do
7:      Get pushed x_{v+1}^(t+1), r_{v+1}^(t+1) through Eq. 15 and 16.
8:    end while
9:    if ΔE^(t) only contains inserted edges on C̄^(t) then
10:     Find the largest index q through Eq. 18.
11:   else
12:     Let q = 0.
13:   end if
      /*Incremental Sweep Cut*/
14:   for j = q + 1 : n^(t+1) do
15:     if there exists j such that:
16:       (C.1) Φ(S_j(x^(t+1)), ℕ) ≤ φ
17:       (C.2) λ_j(x^(t+1)) ≤ (2/3)μ(V)
18:       (C.3) 2^b ≤ λ_j(x^(t+1)) then
19:       Return C^(t+1) = S_j(x^(t+1)) and quit.
20:     end if
21:   end for
22:   Return C^(t+1) = S_q(x^(t+1)).
23: end for
```

to converge the initial solution fit the probability distribution of graph $G^{(t+1)}$ to obtain approximated multilinear PageRank vector $\boldsymbol{x}^{(t+1)}$. Steps 9-13 check whether updated edge set $\Delta E^{(t)}$ meets the conditions of the incremental sweep cut then set q with different values for incremental sweep cut algorithm (i.e. Steps 14-22).

L-MEGA algorithm includes three major proposed techniques for speeding up the computation. First, in updating transition tensor $\bar{\boldsymbol{P}}^{(t+1)}$, instead of considering every updated edge which costs $O(|V^{(t+1)}|^{k-2})$, the edge filtering model filters out some "far-away" edges in $O(|V^{(t+1)}| + |E^{(t+1)}| \log |V^{(t+1)}|)$ time. Second, the motif push operation approximates multilinear PageRank vector in constant number of iterations and each iteration costs polylogarithmic time with respect to the number of edges (Theorem 3.6). Third, incremental sweep cut reduces q iterations ($O(|V^{(t+1)}|^2)$ time complexity each iteration) compared with the original sweep cut procedure. Moreover, the returned cluster $C^{(t+1)}$ is bounded, which means if there is a cluster $\hat{C}^{(t+1)}$ whose motif conductance $\Phi(\hat{C}^{(t+1)}, \mathbb{N}) \leq \frac{1}{2c_1(l+2)}$ and $\frac{x^{(t+1)}(\pi(j))}{d^{(t+1)}(\pi(j))} \geq \frac{1}{c_1(l+2)2^b}$, then $\mu(\hat{C}^{(t+1)} \cap C^{(t+1)}) \geq 2^{(b-1)}$ [36], b and c_1 are two constants, l is proportional to the volume of the graph.

5 EXPERIMENTAL RESULTS

In this section, we demonstrate the effectiveness and scalability of L-MEGA on real-world and synthetic temporal networks. We provide additional results of parameter sensitivity analysis in Appendix C.

5.1 Experiment Setup

Datasets: The real-world temporal networks are summarized in Table 1, which are used to compare the effectiveness of the proposed L-MEGA algorithm with other state-of-the-art methods. We employ four real-world temporal networks. Alpha and OTC networks [14, 15] are rating networks from BitcoinAlpha and BitcoinOTC platforms, where an edge is a rating record between two Bitcoin traders. Call network [25] records human mobile phone call events among a set of core users. Contact network [25] is a human contact network where nodes represent humans and edges between them represent contacts in the physical world. Moreover, to compare the efficiency scalability on different density level temporal networks, we generate two kinds of synthetic networks: control the edge density (i.e., 0.6%) and increase the number of vertices (i.e., from 1,000 to 5,000); control the number of vertices (i.e., 2,000) and increase the edge density (i.e., from 0.5% to 0.9%). The advantage of using synthetic networks is that we can investigate the scalability of efficiency of the proposed L-MEGA framework between the growth of time consumed and the growth of the size of networks.

Table 1: Statistics of Real-world Networks

Network	Category	$\|V\|$	$\|E\|$	Time Span
Alpha	Rating	3,783	14,124	62 months
OTC	Rating	5,881	21,492	62 months
Call	Communication	6,809	7,967	4 months
Contact	Interaction	10,972	44,517	3 months

Prepossessing: We process all time-evolving networks as undirected and unweighted. Due to different time spans shown in Table 1, for each dataset we unify different granularities of time stamps into 10 super time stamps, in each super time stamp, the updated edge set $\Delta E^{(t)}$ occupies 1 percent volume of the whole graph, the initial graph occupies 90 percents volume of the entire graph.

Comparison Methods: We compare L-MEGA with different categories of state-of-the-art clustering algorithms. Static edge-based clustering algorithms (Nibble [28]), dynamic edge-based clustering algorithms (TPPR [21], ISC [20]), static motif-based clustering algorithms (HOSPLOC [36], MAPPR [33]), and our L-MEGA stands for the dynamic motif-based clustering method. Moreover, we provide the self-ablation comparison. L-MEGA-1 replaces the updating transition tensor of L-MEGA with the re-building the tensor from the scratch, L-MEGA-2 replaces the tracking approximated multilinear PageRank vector of L-MEGA with solving it by power iteration, and L-MEGA-3 replaces the incremental sweep cut with the original sweep cut procedure. We illustrate the reproducibility of the proposed L-MEGA framework in Appendix D.

5.2 Effectiveness Comparison

Evaluation Metrics: We set the triangle as the motif $\mathbb{N}$, triangle is the widely adopted motif for high-order clustering in existing work [4, 5, 32, 33, 36] for balancing the validity of experiments with the heavy load of time complexity $O(n^{(k)})$, also triangles have the fundamental role in understanding the social network and community structures [9, 11, 24]. Moreover, we use three metrics: (1) **conductance**, which measures the general quality of a cut on the graph to indicate the compactness of the cut, and it is computed by setting the undirected edge as the motif $\mathbb{N}$ in Eq. 1; (2) **third-order conductance**, which could be computed with Eq. 1 by setting the undirected triangles as the motif $\mathbb{N}$; (3) **triangle density**, which computes the ratio of triangles included in the returned local cluster.

Quantitive Results: The clustering effectiveness and consumed time of each baseline are shown in Table 2, where we select different seed nodes for representing the average and the standard deviation of the first local minimum conductance score and its associated triangle density under the same parameter ϕ of the sweep cut. For the static algorithm, the clustering result is solely obtained at the tenth time stamp, and for the dynamic clustering algorithms, the clustering result is tracked from the first time stamp to the tenth time stamp. Because the updating and tracking process of the L-MEGA framework have the accumulated error, if L-MEGA performs well at the tenth time stamp, it also performs well at previous time stamps. From Table 2, we observe that our proposed L-MEGA achieves comparable performance with the baseline methods across real-world dynamic networks in terms of three aforementioned metrics. For example, L-MEGA performs slightly better than static algorithms like HOSPLOC in both conductance and third-order conductance. An intuitive explanation is that, the last time tracked approximated multilinear PageRank is closer to the stationary distribution than the approximated multilinear PageRank solved by HOSPLOC from the scratch. Comparing L-MEGA with L-MEGA-1, we know that building transition tensor occupies major time consumption of the motif-based clustering algorithms, and our updating transition tensor algorithm provides acceptable accuracy and reduces time complexity to a large extent. Comparing L-MEGA with L-MEGA-2, we know that tracking multilinear PageRank algorithm also reduces the consumed time and provides adequate accuracy. Comparing L-MEGA with L-MEGA-3, we observe that incremental sweep cut also saves time and ensures preciseness. Observing the time consumption of L-MEGA in Alpha network and Call Network, we find running L-MEGA on Call is faster. We suppose that the growth of the consumed time of L-MEGA may not relate with the size of the graph but with the dynamics between two consecutive time stamps. To prove our guess, we design the following scalability analysis.

5.3 Scalability Analysis

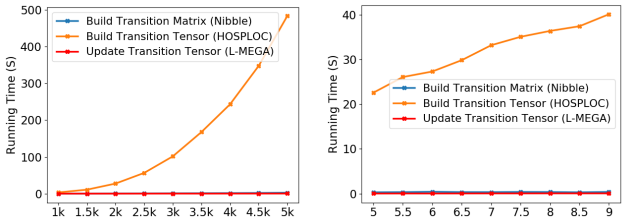
Here, we use the synthetic networks mentioned above, and the scalability analysis is divided into the following three parts comparing with PageRank-based clustering algorithms: Nibble and HOSPLOC.

First, we analyze the efficiency of constructing matrices/tensors in Fig. 2. With the raw graph data, Nibble builds the two-dimensional transition matrix, HOSPLOC builds the three-mode transition tensor, and our L-MEGA tracks the three-mode transition tensor. We plot their consumed time during one time stamp in Fig. 2. We test the running time of processing the raw graph in terms of increasing number of vertices (Fig. 2a) and increasing number of edges (Fig. 2b). From Fig. 2a and Fig. 2b, when the number of vertices and

Table 2: Comparison of Local Motif Clustering Effectiveness and Efficiency

Methods	Alpha				OTC			
	conductance	third-order conductance	triangle density	time	conductance	third-order conductance	triangle density	time
Nibble	0.4909 ± 0.0060	0.4555 ± 0.0454	0.2355 ± 0.1033	18.4073 ± 5.9853	0.4963 ± 0.0045	0.5091 ± 0.0941	0.1582 ± 0.1076	63.1869 ± 34.2154
TPPR	0.4923 ± 0.0089	0.4994 ± 0.1188	0.1613 ± 0.0934	12.4094 ± 5.7653	0.4970 ± 0.0021	0.5751 ± 0.1106	0.1524 ± 0.1320	39.1307 ± 19.3550
ISC	0.3334 ± 0.0000	1.0000 ± 0.0000	0.0000 ± 0.0000	56.6376 ± 0.0000	0.5999 ± 0.0000	0.5656 ± 0.0000	0.1908 ± 0.0000	195.5490 ± 0.0000
MAPPR	0.4947 ± 0.0008	0.5852 ± 0.0104	0.0712 ± 0.0030	43.0597 ± 2.9107	0.4890 ± 0.0015	0.5404 ± 0.0023	0.0904 ± 0.0001	207.5004 ± 1.1757
HOSPLOC	0.4915 ± 0.0080	0.4816 ± 0.0576	0.1891 ± 0.0859	237.6121 ± 12.5513	0.4957 ± 0.0041	0.5080 ± 0.0722	0.2000 ± 0.1172	753.3742 ± 51.6812
L-MEGA	0.4712 ± 0.0586	**0.4097 ± 0.0278**	0.2561 ± 0.1008	**8.2032 ± 4.8534**	0.4652 ± 0.0074	**0.4102 ± 0.0620**	0.2946 ± 0.0719	**32.4308 ± 46.8278**
L-MEGA-1	0.4728 ± 0.0102	0.4676 ± 0.0344	0.2490 ± 0.0736	241.4762 ± 13.3320	0.4733 ± 0.0074	0.4622 ± 0.0547	0.2578 ± 0.0961	778.5583 ± 33.4156
L-MEGA-2	0.4944 ± 0.0036	0.4369 ± 0.0428	0.3819 ± 0.0737	14.8578 ± 4.0788	0.4860 ± 0.0013	0.4750 ± 0.0175	0.5318 ± 0.0141	32.6827 ± 1.6759
L-MEGA-3	0.4712 ± 0.0586	0.4097 ± 0.0278	0.2561 ± 0.1008	11.4955 ± 4.2939	0.4652 ± 0.0074	0.4102 ± 0.0620	0.2946 ± 0.0719	45.5937 ± 45.6706
Methods	Call				Contact			
	conductance	third-order conductance	triangle density	time	conductance	third-order conductance	triangle density	time
Nibble	0.0792 ± 0.0309	0.5675 ± 0.4809	0.0249 ± 0.0384	13.5155 ± 2.7236	0.3536 ± 0.0925	0.2878 ± 0.1857	0.0017 ± 0.0015	33.7139 ± 0.1147
TPPR	0.1910 ± 0.1399	0.5589 ± 0.4442	0.0274 ± 0.0420	8.3268 ± 2.1605	0.2643 ± 0.1323	0.2221 ± 0.1382	0.0025 ± 0.0023	25.0759 ± 0.1416
ISC	0.5893 ± 0.0000	0.5270 ± 0.0000	0.1700 ± 0.0000	27.3982 ± 0.0000	0.4765 ± 0.0000	0.5252 ± 0.0000	0.0035 ± 0.0000	1351.1732 ± 0.0000
MAPPR	0.5957 ± 0.0042	0.4401 ± 0.0291	0.2219 ± 0.1869	2938.3853 ± 81.2163	0.3317 ± 0.0573	0.2790 ± 0.0753	0.0006 ± 0.0003	88.6153 ± 0.2981
HOSPLOC	0.1652 ± 0.0485	0.2981 ± 0.3721	0.0296 ± 0.0416	768.4879 ± 1.1554	0.2646 ± 0.1346	0.2308 ± 0.1559	0.0034 ± 0.0051	3443.8829 ± 0.2193
L-MEGA	0.1542 ± 0.0544	0.2866 ± 0.3823	0.0395 ± 0.0448	**1.1316 ± 0.9816**	0.2438 ± 0.1676	**0.1614 ± 0.1443**	0.0042 ± 0.0052	**3.4496 ± 3.0660**
L-MEGA-1	0.1542 ± 0.0544	0.2866 ± 0.3823	0.0395 ± 0.0448	754.2537 ± 0.8341	0.2028 ± 0.1349	0.2172 ± 0.1921	0.0004 ± 0.0005	3492.1936 ± 3.2966
L-MEGA-2	0.2333 ± 0.1839	**0.2713 ± 0.3883**	0.0239 ± 0.0180	17.2451 ± 0.9250	0.2511 ± 0.1121	0.2166 ± 0.1304	0.0003 ± 0.0004	48.1245 ± 1.1528
L-MEGA-3	0.1542 ± 0.0544	0.2866 ± 0.3823	0.0395 ± 0.0448	1.2502 ± 1.0725	0.2438 ± 0.1676	0.1614 ± 0.1443	0.0042 ± 0.0052	3.5800 ± 3.1314

the number of edges are increasing, the running time of building matrices/tensors is increasing, that is because more vertices are involved in the formation of the matrix (or tensor). However, in both settings, L-MEGA method is near constantly increasing, because L-MEGA takes advantage of the information from the last time stamp, and updates it by locating the sphere which may be influenced by updated edges. Hence, L-MEGA reduces much unnecessary computation.

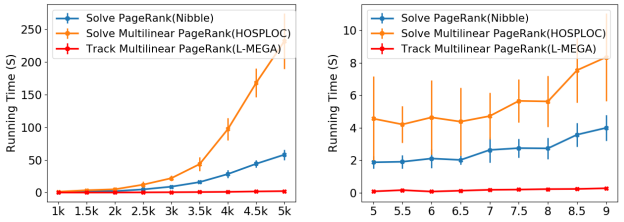


(a) The number of vertices **(b) The edge density**

Figure 2: Efficiency of constructing matrices/tensors.

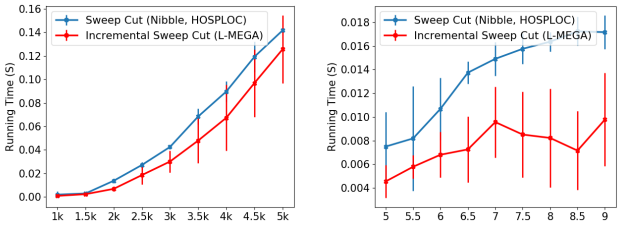
Second, we test the running time of solving the approximated (multilinear) PageRank and tracking the approximated multilinear PageRank under the same converge condition. In Fig. 3a, solving the multilinear PageRank from the scratch is increasing quadratically with the number of vertices and linearly the number of edges; solving the PageRank is increasing linearly in both settings; while our tracking method is increasing constantly in Fig. 3a and Fig. 3b. Also, there is an interesting observation: in both settings, Nibble and HOSPLOC have an observable standard deviation due to their intrinsic randomness for different selected seed nodes. However, L-MEGA performs consistently for different given seed nodes.

Third, we test the efficiency of identifying a local cluster from the cluster indicator vector, i.e, multilinear PageRank vector. In Fig. 4, we compare the proposed incremental sweep cut with the original



(a) The number of vertices **(b) The edge density**

Figure 3: Efficiency of getting the cluster indicator vector.



(a) The number of vertices **(b) The edge density**

Figure 4: Efficiency of identifying the graph cut.

sweep cut, which is widely used by many clustering algorithms like HOSPLOC and Nibble. In general, the incremental sweep cut outperforms the original sweep cut method in running time, and both of them are increasing linearly with the number of nodes and edges. It is interesting that the standard deviation of incremental sweep cut is larger than the original static sweep cut in Fig. 4. It is because the random choice of seed node and the updated edges can result in different value of q in increment sweep cut, which in turn affects the number of iterations of incremental sweep cut. When q is large, incremental sweep cut will reduce a considerable number of iterations; on the contrary, when q is small, incremental sweep cut will perform similarly as the original sweep cut.

6 RELATED WORK

The informative and representative nature of graph structures can be extended to many domains like anomaly detection [3], missing information imputation [17] and crowdsourcing [37, 38]. Graph clustering is the task of grouping various graph connectivity patterns (e.g., edges and motifs) into clusters, where rich connectivity patterns are within each cluster while few are between clusters. Many existing graph clustering approaches are known to be designed for the edge-based connectivity patterns, such as the PageRank-based methods [1, 28], the spectral-based clustering method [19], and the kernel-based method [13]. The traditional dynamic method [7] assumes that clustering results should not shift dramatically from time stamp to time stamp, therefore, the clustering result produced by that framework balances the history cost. Latter, some works propose different techniques for the dynamic clustering like the parameter-free clustering method [29], the incremental spectral clustering method [20], and the matrix factorization method [2]. More recently, the high-order organization of complex networks has received a surge of research interest in the graph mining area [5, 18]. For example, in [33], the authors come up with the motif-based approximated personalized PageRank algorithm for the local motif clustering; in [36], the authors propose a high-order structure-preserving local cut framework, which defines the adjacency tensor and transition tensor regarding the given high-order substructure then applies the high-order Markov Chain technique for computing the proximity vector in an unfolded matrix of the transition tensor. Different from the aforementioned static methods, in this paper, our proposed dynamic local motif clustering algorithm is designed for time-evolving graphs, which aims to update the motif clusters over time in an effective and efficient way.

7 CONCLUSION

In this paper, we propose a novel framework named L-MEGA, which performs dynamic local motif clustering effectively and efficiently. In particular, we designed novel techniques such as edge filtering, motif push operation, and incremental sweep cut to speed up the computation. We theoretically analyze the efficiency and effectiveness of these techniques and evaluate L-MEGA via extensive experiments on synthetic and real-world temporal networks.

ACKNOWLEDGEMENT

This work is supported by the United States Air Force and DARPA under contract number FA8750-17-C-0153, National Science Foundation under Grant No. IIS-1947203 and Grant No. IIS-2002540, the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-03-03 and Ordering Agreement Number HSHQDC-16-A-B0001. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2006. Local Graph Partitioning using PageRank Vectors. In *IEEE FOCS 2006.*
[2] Ana Paula Appel, Renato Luiz de Freitas Cunha, Charu C. Aggarwal, and Marcela Megumi Terakado. 2018. Temporally Evolving Community Detection and Prediction in Content-Centric Networks. In *ECML PKDD 2018.*
[3] Yikun Ban, Xin Liu, Ling Huang, Yitao Duan, Xue Liu, and Wei Xu. 2019. No Place to Hide: Catching Fraudulent Entities in Tensors. In *WWW 2019.*
[4] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2015. Tensor Spectral Clustering for Partitioning Higher-order Network Structures. In *SIAM SDM 2015.*
[5] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* (2016).
[6] Austin R. Benson, David F. Gleich, and Lek-Heng Lim. 2017. The Spacey Random Walk: A Stochastic Process for Higher-Order Data. *SIAM Rev.* (2017).
[7] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. 2006. Evolutionary Clustering. In *ACM SIGKDD 2006.*
[8] Edsger W. Dijkstra. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik* (1959).
[9] Nurcan Durak, Ali Pinar, Tamara G. Kolda, and C. Seshadhri. 2012. Degree Relations of Triangles in Real-world Networks and Graph Models. In *ACM CIKM 2012.*
[10] David F. Gleich, Lek-Heng Lim, and Yongyang Yu. 2015. Multilinear PageRank. *SIAM J. Matrix Anal. Appl.* (2015).
[11] Mark S. Granovetter. 1973. The Strength of Weak Ties. *Amer. J. Sociology* (1973).
[12] Chris Jay Hoofnagle. 2007. Identity theft: Making the known unknowns known. *Harv. JL & Tech.* (2007).
[13] Brian Kulis, Sugato Basu, Inderjit S. Dhillon, and Raymond J. Mooney. 2005. Semi-supervised Graph Clustering: A Kernel Approach. In *ICML 2005.*
[14] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2018. REV2: Fraudulent User Prediction in Rating Platforms. In *ACM WSDM 2018.*
[15] Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. 2016. Edge Weight Prediction in Weighted Signed Networks. In *IEEE ICDM 2016.*
[16] Wen Li and Michael K. Ng. 2014. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra* (2014).
[17] Xu Liu, Jingrui He, Sam Duddy, and Liz O'Sullivan. 2019. Convolution-Consistent Collective Matrix Completion. In *ACM CIKM 2019.*
[18] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* (2002).
[19] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On Spectral Clustering: Analysis and an algorithm. In *NeurIPS 2001.*
[20] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S. Huang. 2010. Incremental Spectral Clustering by Efficiently Updating the Eigen-System. *Pattern Recognition* (2010).
[21] Naoto Ohsaka, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Efficient PageRank Tracking in Evolving Networks. In *ACM SIGKDD 2015.*
[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report.
[23] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In *ACM WSDM 2017.*
[24] Arnau Prat-Pérez, David Dominguez-Sal, Josep M Brunat, and Josep-Lluis Larriba-Pey. 2012. Shaping Communities out of Triangles. In *ACM CIKM 2012.*
[25] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI 2015.*
[26] Martin Rosvall, Alcides Viamontes Esquivel, Andrea Lancichinetti, Jevin West, and Renaud Lambiotte. 2014. Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications* (2014).
[27] Jirí Síma and Satu Elisa Schaeffer. 2006. On the NP-Completeness of Some Graph Cluster Measures. In *SOFSEM 2006.*
[28] 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.* (2013).
[29] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. 2007. GraphScope: Parameter-free Mining of Large Time-evolving Graphs. In *ACM SIGKDD 2007.*
[30] Hanghang Tong, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. 2008. Proximity Tracking on Time-Evolving Bipartite Graphs. In *SIAM SDM 2008.*
[31] Konstantin Voevodski, Shang-Hua Teng, and Yu Xia. 2009. Spectral affinity in protein networks. *BMC Systems Biology* (2009).
[32] Tao Wu, Austin R. Benson, and David F. Gleich. 2016. General Tensor Spectral Co-clustering for Higher-Order Data. In *NeurIPS 2016.*
[33] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *ACM SIGKDD 2017.*
[34] Hongyang Zhang, Peter Lofgren, and Ashish Goel. 2016. Approximate Personalized PageRank on Dynamic Graphs. In *ACM SIGKDD 2016.*
[35] Dawei Zhou, Jingrui He, Hasan Davulcu, and Ross Maciejewski. 2018. Motif-Preserving Dynamic Local Graph Cut. In *IEEE Big Data 2018.*
[36] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A Local Algorithm for Structure-Preserving Graph Cut. In *ACM SIGKDD 2017.*
[37] Yao Zhou and Jingrui He. 2017. A Randomized Approach for Crowdsourcing in the Presence of Multiple Views. In *IEEE ICDM 2017.*
[38] Yao Zhou, Lei Ying, and Jingrui He. 2017. MultiC^2: an Optimization Framework for Learning from Task and Worker Dual Heterogeneity. In *SIAM SDM 2017.*

A NOTATION

Table 3: Table of Notation

Symbol	Definition and Description
$\mathbb{N}$	user-defined motif
$G^{(t)}$	time-evolving graph at time stamp t
$\Delta E^{(t)}$	updated edge set at time stamp t
$\boldsymbol{d}^{(t)}$	degree vector of graph $G^{(t)}$
$\boldsymbol{x}^{(t)}$	multilinear PageRank vector at time stamp t
$\boldsymbol{r}^{(t)}$	residual vector of multilinear PageRank vector $\boldsymbol{x}^{(t)}$
$\bar{\boldsymbol{I}}^{(t)}$	dynamic indicator tensor at time stamp t
$\bar{\boldsymbol{P}}^{(t)}$	dynamic transition tensor at time stamp t
$\boldsymbol{P}^{(t)}$	unfolded matrix of $\bar{\boldsymbol{P}}^{(t)}$

B ALGORITHM ANALYSIS

B.1 Proof of Lemma 3.4

If an updated edge $e = (v_1, v_2)$ is a "far-away" updated edge, then according to Eq. 9, we have

$$\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)} < \frac{x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))}, \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)} < \frac{x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))}$$

hence, we know that the "far-away" update $e = (v_1, v_2)$ can only occur on the complement of local cluster (i.e., $\bar{C}^{(t)}$), furthermore, only on the nodes who rank after $j+1$ in the permutation π at time t. Also, from Eq. 11, v_1 and v_2 are at least $k-1$ hops away from the local cluster $C^{(t)}$ which means v_1 and v_2 are not directly connected with any nodes in $C^{(t)}$. Because permutation π is nonincreasing, when a "far-away" update occurs on the complement $\bar{C}^{(t)}$, node $\pi(j+1)$ has the largest potential to break through and rank higher than node $\pi(j)$ at time $t+1$ to change the structure of the local cluster $C^{(t)}$. If node $\pi(j+1)$ fails to rank higher than node $\pi(j)$ at time $t+1$, then any nodes behind $\pi(j+1)$ will also fail.

Next, we need to prove that adding the probability mass increment of the "far-away" updated edge $e = (v_1, v_2)$ on node $\pi(j+1)$ is not enough to make node $\pi(j+1)$ rank higher than node $\pi(j)$ at time $t+1$.

In Eq. 10, γ is formed as the largest probability mass increment that v_1 and v_2 can contribute to node $\pi(j+1)$. Based on Eq. 10, we have

$$\frac{\gamma + x^{(t)}(\pi(j+1))}{d^{(t)}(\pi(j+1))} < \frac{x^{(t)}(\pi(j))}{d^{(t)}(\pi(j))}$$

which states that v_1 and v_2 are incapable to help the node $\pi(j+1)$ to rank higher than $\pi(j)$ at time $t+1$. Because the permutation π is nonincreasing which implies that nodes rank lower than $\pi(j+1)$ at time t have also no chance to rank higher than $\pi(j)$ at time $t+1$. So far, the first part of Lemma 3.4 is proved that the first j entries of permutation π remains between two consecutive time stamps, $S_j(\boldsymbol{x}^{(t)}) = S_j(\boldsymbol{x}^{(t+1)})$.

Moreover, the reason why the constructed γ represents the largest contributed probability mass increment from nodes v_1 and v_2 to node $\pi(j+1)$ states as below.

At time stamp t, assume node $\pi(j+1)$ is reachable from the "far-away" updates v_1, then the most efficient channel to transmit the probability mass from node v_1 to node $\pi(j+1)$ is that v_1 is directly connected with node $\pi(j+1)$. Then, in the k^{th}-order setting, the $(k-1)^{th}$-order Markov Chain is modeled as follows.

$$Prob(S_{t+1} = \pi(j+1) | S_t = v_1, S_{t-1} = v_i, \ldots, S_{t-k+2} = v_i) \quad (19)$$

where S_t denotes a discrete time stochastic process on the state space $1, \ldots, n$, v_i is the node (i.e., state of the stochastic process) such that $i \in \{1, \ldots, n\}$.

We reduce the $(k-1)^{th}$-order Markov Chain to a first-order Markov Chain by omitting the tail from S_{t-1} to S_{t-k+2}, which is modeled as follows.

$$\begin{aligned} &Prob(S_{t+1} = \pi(j+1) | S_t = v_1) \geq \\ &Prob(S_{t+1} = \pi(j+1) | S_t = v_1, S_{t-1} = v_i, \ldots, S_{t-k+2} = v_i) \end{aligned} \quad (20)$$

Then, we can take the $Prob(S_{t+1} = \pi(j+1) | S_t = v_1) = \frac{1}{d^{(t)}(v_1)}$ which means the next random walk step starting from node v_1 will reach node $\pi(j+1)$ with probability $\frac{1}{d^{(t)}(v_1)}$. Therefore, node v_1 can transmit probability mass $\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)}$ to node $\pi(j+1)$. Analogously, node v_2 can transmit probability mass $\frac{x^{(t)}(v_2)}{d^{(t)}(v_2)}$ to node $\pi(j+1)$ respectively.

Case 1. "Far-away" updated edge $e = (v_1, v_2)$ is an inserted edge.

$$\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)} \leq \frac{x^{(t)}(v_1) + x^{(t)}(v_2)}{\min(d^{(t)}(v_1), d^{(t)}(v_2))} \quad (21)$$

where the left-hand-side is the summation of the transmitted probability mass from node v_1 and v_2 before the insertion; the right-hand side is the maximum transmitted probability mass after the insertion by taking v_1 and v_2 as one node and selecting the less transition loss channel (i.e., minimum of degree). Therefore, in the insertion scenario, we construct γ as $\frac{x^{(t)}(v_1)+x^{(t)}(v_2)}{\min(d^{(t)}(v_1), d^{(t)}(v_2))}$.

Case 2. "Far-away" updated edge $e = (v_1, v_2)$ is a deleted edge.

$$\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)} < \frac{x^{(t)}(v_1)}{d^{(t)}(v_1) - 1} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2) - 1} \quad (22)$$

where the left-hand-side is the summation of the transmitted probability mass from node v_1 to $\pi(j+1)$ and the transmitted probability mass from node v_2 to $\pi(j+1)$ before the deletion; the right-hand-side is the summation of the transmitted probability mass from v_1 to $\pi(j+1)$ and from v_2 to $\pi(j+1)$ after the deletion. Due to the deletion, the probability mass increment of node $\pi(j+1)$ after one random walk is actually $(\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)-1} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)-1}) - (\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)})$, we enlarge it by forming γ as $\frac{x^{(t)}(v_1)}{d^{(t)}(v_1)-1} + \frac{x^{(t)}(v_2)}{d^{(t)}(v_2)-1}$.

For Lemma 3.4, we have proved that one "far-away" update has no impact on sweep set $S_j(\boldsymbol{x}^{(t)}) = S_j(\boldsymbol{x}^{(t+1)})$, i.e., the ranking of first j nodes of permutation π between two consecutive time stamps does not change. Now, we only need to prove that node $\pi(j)$ is still the first node to satisfy conditions (C.1) to (C.3) regrading the same parameter set $B = \{\phi, b\}$.

Proof of condition (C.1). At time t, we have the local cluster $C^{(t)} = S_j(\boldsymbol{x}^{(t)})$ such that $\Phi(S_j(\boldsymbol{x}^{(t)}), \mathbb{N}) \leq \phi$. According to Eq. 1, we have

$$\Phi(S_j(\boldsymbol{x}^{(t)}), \mathbb{N}) = \frac{cut(S_j(\boldsymbol{x}^{(t)}))}{\min(\mu(S_j(\boldsymbol{x}^{(t)}), \mathbb{N}), \mu(\bar{S_j}(\boldsymbol{x}^{(t)}), \mathbb{N}))} \leq \phi$$

According to Eq. 11, the "far-away" updated edge $e = (v_1, v_2)$ is not directly connected with the local cluster $C^{(t)}$ and at least $k-1$ hops away from $C^{(t)}$, in other words, Eq. 11 denies any new k^{th}-order motif $\mathbb{N}$ involved into the cut identified at time t. Then, we can derive that $cut(S_j(\boldsymbol{x}^{(t)}) = cut(S_j(\boldsymbol{x}^{(t+1)})$. Because Eq. 9 limits the "far-away" update can only occur on the complement of the local cluster, i.e., $\bar{S}_j(\boldsymbol{x}^{(t)})$, such that if $\mu(S_j(\boldsymbol{x}^{(t)}), \mathbb{N}) < \mu(\bar{S}_j(\boldsymbol{x}^{(t)}), \mathbb{N})$ at time t, then an "far-away" inserted edge at time $t+1$ will have

$$\min(\mu(S_j(\boldsymbol{x}^{(t+1)}), \mathbb{N}), \mu(\bar{S}_j(\boldsymbol{x}^{(t+1)}), \mathbb{N})) = \mu(S_j(\boldsymbol{x}^{(t)}), \mathbb{N}) \tag{23}$$

and if $\mu(S_j(\boldsymbol{x}^{(t)}), \mathbb{N}) > \mu(\bar{S}_j(\boldsymbol{x}^{(t)}), \mathbb{N})$ at time t, then an "far-away" deleted edge at time $t+1$ will have

$$\min(\mu(S_j(\boldsymbol{x}^{(t+1)}), \mathbb{N}), \mu(\bar{S}_j(\boldsymbol{x}^{(t+1)}), \mathbb{N})) = \mu(\bar{S}_j(\boldsymbol{x}^{(t)}), \mathbb{N}) \tag{24}$$

Thus, according to the motif conductance score of the sweep cut procedure, node $\pi(j)$ is still the first node satisfying condition (C.1) at time $t+1$.

Proof of conditions (C.2) and (C.3). Conditions (C.2) and (C.3) limit the volume of the local cluster $C^{(t)}$ should not be too small or too big, which is bounded by $2^b \leq \lambda_j(\boldsymbol{x}^{(t)}) \leq (2/3)\mu(V^{(t)})$.

Since Eq. 9 allows "far-away" updates can only occur on the complement $\bar{C}^{(t)}$, the volume of the local cluster $C^{(t)}$ is not changed, which means $\lambda_j(\boldsymbol{x}^{(t)}) = \lambda_j(\boldsymbol{x}^{(t+1)})$. Because we assume the size of updated edge set $\Delta E^{(t)}$ is much smaller than the whole graph $G^{(t)}$, not to mention the size of "far-away" updated edges, therefore, $\mu(V^{(t)}) \approx \mu(V^{(t+1)})$. Thus, node $\pi(j)$ is still the first node satisfying conditions (C.2) and (C.3) at time $t+1$ is proved.

To summarize, we have proved that node $\pi(j)$ is still the first node satisfying conditions (C.1) to (C.3) in the sweep cut procedure regarding the same parameter set $B = \{\phi, b\}$.

B.2 Proof of Theorem 3.5

A single "far-away" updated edge has no impact on the evolution of the local cluster has been proved by the Lemma 3.4. Then, Lemma 3.4 and Eq. 13 limit that the summation of probability mass increments of sampled "far-away" inserted (or deleted) edges set $\{e_1, \ldots, e_m\}$ could also not improve node $\pi(j+1)$ rank higher than node $\pi(j)$ at time $t+1$; and $\mu(C^{(t)}, \mathbb{N}) < \mu(\bar{C}^{(t)}, \mathbb{N})$ (or $\mu(C^{(t)}, \mathbb{N}) > \mu(\bar{C}^{(t)}, \mathbb{N})$) indicates that the node $\pi(j)$ is still the first node to satisfy conditions (C.1) to (C.3) of sweep cut procedure at time $t+1$.

B.3 Proof of Theorem 3.6

We define the potential δ_v of $\boldsymbol{r}_v^{(t+1)}$ at the v-th iteration of the motif push operation as follows.

$$\delta_v = \|\boldsymbol{r}_v^{(t+1)}\|_1 \tag{25}$$

Based on Cauchy-Schwarz inequality, in Eq. 16, we have the third term $\|\alpha r_v^{(t+1)}(i)\boldsymbol{P}^{(t+1)}(\boldsymbol{e} \otimes \boldsymbol{x}_{v-1}^{(t+1)} \ldots \otimes \boldsymbol{x}_{v-k+2}^{(t+1)})\|_1 \leq \alpha r_v^{(t+1)}(i)$. Thus, the v-th iteration reduces δ_0 by at least $(1-\alpha)r_v^{(t+1)}(i)$. Hence, after an adequate number of iterations, motif push operation makes the final residual vector $\boldsymbol{r}^{(t+1)} \to \boldsymbol{0}$. Furthermore, for each motif push operation iteration, we have

$$\delta_{v+1} \leq \delta_v - (1-\alpha)r_v^{(t+1)}(i) \leq \delta_v - (1-\alpha)\epsilon \tag{26}$$

since $\delta_v \geq 0$ and maximum of δ_v is $\delta_0 \leq 1$, hence, motif push operation will terminate in at most $\frac{1}{(1-\alpha)\epsilon}$ number of iterations. The compelling complexity of each iteration is the lazy updating rule (i.e., Eq. 17) which runs in polylogarithmic time $O(\frac{1}{\xi^k})$ with respect to the number of edges [36], where $\xi \propto \frac{1}{\log_2(\mu(V^{(t+1)}))}$, $\mu(V^{(t+1)})$ is the volume of graph $G^{(t+1)}$, k is the order of motif $\mathbb{N}$.

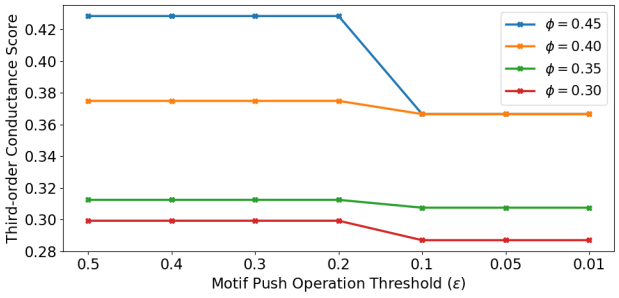


Figure 5: Parameter sensitivity.

C PARAMETER SENSITIVITY

There are two important parameters in L-MEGA framework, they are motif push operation threshold ϵ and motif conductance upper bound ϕ. The first parameter is responsible for limiting the error of the approximated stationary distribution of the multilinear PageRank vector, and the second parameter is responsible for controlling the quality of the returned motif-aware local cluster. Here, we generate a synthetic graph having $2,000$ vertices with 0.5% edge density. Setting the undirected triangle as the motif $\mathbb{N}$, we measure the third-order conductance of the returned local cluster identified by different values of the third-order conductance upper bound ϕ and the motif push operation threshold ϵ. In Fig. 5, we have the following observations: (1) the quality (i.e., the first local minimum third-order conductance) of the returned local partition is positively correlated to the third-order conductance upper bound ϕ and motif push operation threshold ϵ; (2) the performance of L-MEGA algorithm is sensitive to different values of the third-order conductance upper bound ϕ but not very sensitive to the changes of the motif push operation threshold ϵ, because during the process of approximating stationary distribution of the multilinear PageRank vector, much more precision may not change the ranking of vertices in the $S_j(\boldsymbol{x}^{(t+1)})$ to a corresponding large extent, which means less precise motif push operation threshold ϵ can achieve the comparable accuracy.

D REPRODUCIBILITY

The real-world data sets (i.e., Alpha[1], OTC[2], Call[3], Contact[4]) are publicly available. The synthetic data and the code of the L-MEGA framework are released on the author's website[5]. The experiments are programmed based on Python 3.7 on a Windows machine with four 3.6GHz Intel Cores and 64GB RAM.

[1] http://snap.stanford.edu/data/soc-sign-bitcoin-alpha.html
[2] http://snap.stanford.edu/data/soc-sign-bitcoin-otc.html
[3] http://networkrepository.com/ia-reality-call.php
[4] http://networkrepository.com/ia-contacts-dublin.php
[5] https://github.com/DongqiFu/L-MEGA