Motif-Preserving Dynamic Local Graph Cut

Dawei Zhou, Jingrui He, Hasan Davulcu, Ross Maciejewski
Arizona State University
Email: {dzhou23, jingrui.he, hdavulcu, rmacieje}@asu.edu

***Abstract*—Modeling and characterizing high-order connectivity patterns are essential for understanding many complex systems, ranging from social networks to collaboration networks, from finance to neuroscience. However, existing works on high-order graph clustering assume that the input networks are static. Consequently, they fail to explore the rich high-order connectivity patterns embedded in the network evolutions, which may play fundamental roles in real applications. For example, in financial fraud detection, detecting loops formed by sequenced transactions helps identify money laundering activities; in emerging trend detection, star-shaped structures showing in a short burst may indicate novel research topics in citation networks. In this paper, we bridge this gap by proposing a local graph clustering framework that captures structure-rich subgraphs, taking into consideration the information of high-order structures in temporal networks. In particular, our motif-preserving dynamic local graph cut framework (*MOTLOC*) is able to model various user-defined temporal network structures and find clusters with minimum conductance in a polylogarithmic time complexity. Extensive empirical evaluations on synthetic and real networks demonstrate the effectiveness and efficiency of our *MOTLOC* framework.**

I. Introduction

Nowadays, large-scale network data is being generated at an unprecedented speed from various domains, ranging from social networks to collaboration networks, from finance to neuroscience. Arguably, the graph clustering algorithms provide us an important tool to study the topology of networks and to identify low conductance communities with closely related entities. Despite their algorithmic simplicity and theoretical elegance, most existing works are inherently limited to preserving the lower-order connectivity patterns, that can be captured at the level of individual nodes and edges, with the first-order Markov chain interpretation. However, in practice, many real networks are more naturally described by the higher-order connectivity patterns (e.g., triangles, loops, and stars). It has been shown that incorporating the higher-order connectivity information into the graph clustering process can significantly improve our understanding of the underlying complex system [30]. For instance, in social networks, triangles have been proven to play the fundamental role in understanding community structures [11]; in transaction networks, directed cycles might be red flags for potential money laundering activities [7]; in collaboration networks, star-shaped structures appearing in a short burst may indicate an emerging trend in research communities [27].

Moreover, in many real-world applications, the networks are evolving over time [21], i.e., the vertices and edges may appear, vanish, or even re-appear. Existing techniques generally model the dynamic networks as either (1) strictly growing graphs where once the nodes and edges appear, they stay forever [2]; or (2) time-evolving graphs where the temporal information of networks is aggregated into a sequence of snapshots [27]. However, none of these techniques can precisely capture the rich temporal network structures in the data, since they neglect the fact that the interactions between vertices are all individually time-stamped in many dynamic systems. For example, being extracted from the temporal movie review network in Fig. 1(a), Fig. 1(c) shows a 4-node temporal loop which is formed by temporal edges in chronological order, while the prior methods may neglect the order information which is crucial for social influence analysis [25] between users; Fig. 1(d) presents a 6-node star where all the edges are created on day 6, while the existing methods cannot model the cases where many related edges occur in a short burst, potentially related to a hot movie on day 6. Note that, even with the same topology, the network structures may have diverse variations in temporal networks. More details regarding temporal networks and temporal network structures will be discussed later in Section 3.

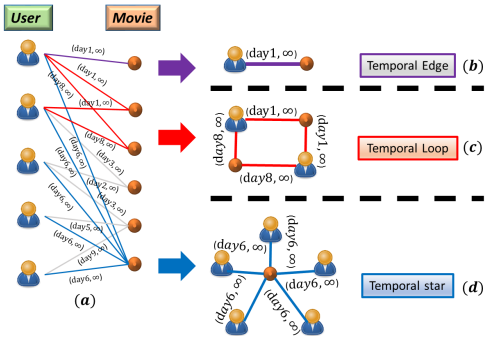


Fig. 1. Illustration of temporal network structures. (a) An example of a temporal movie review network. Each edge is presented with the starting time stamp (day) τ and existing duration (days) δ (the (τ, δ) information is located on the top of each edge). (b) An example of a temporal edge that is created on day 1. (c) An example of a 4-node temporal loop that is formed by four temporal edges in chronological order. (d) An example of a 6-node temporal star, where all the temporal edges are created on day 6.

In this paper, we address such challenges from multiple aspects. In particular, this paper tries to answer the following questions: **Q1:** How to define a good temporal network model which can represent the underlying high-order structures in dynamic systems? **Q2:** How to conduct the graph partitioning that preserves rich user-defined temporal structures in the returned clusters? **Q3:** How to ensure the proposed algorithm is scalable for massive real-world networks?

To address these problems, we start with the general notion

of the k^{th}-Order τ-Duration δ-Temporal Network Structure for modeling the high-order connectivity patterns in the temporal networks. Then, inspired by the family of local graph clustering algorithms for efficiently conducting sparse cut without exploring the entire network, we generalize the idea and propose a motif-preserving dynamic local graph cut (*MOTLOC*) framework that allows user to specify a temporal network structure and finds a sparse cut which has small conductance and largely preserves such structures in the returned cluster with a polylogarithmic time complexity. Finally, we evaluate the performance of *MOTLOC* from multiple aspects using various real-world temporal networks.

To summarize, our work makes the following contributions:

1) Definitions of temporal networks and temporal network structures for dynamic systems.
2) A general algorithm to encode user-defined high-order temporal network structure patterns into a tensor representation of the given network.
3) A local algorithm (*MOTLOC*) for structure-preserving graph cut on the temporal network.
4) Extensive experiments and case-studies on both real and synthetic data sets, showing the effectiveness and efficiency of the proposed algorithms.

The rest of our paper is organized as follows. Related works are given in Section 2. In Section 3, we introduce the notions of temporal networks and temporal network structures. Section 4 presents our proposed structure-preserving graph cut framework *MOTLOC*. Experimental results are presented in Section 5, and finally we conclude the paper in Section 6.

II. Related Work

In this section, we briefly review the existing works on dynamic network mining and graph clustering algorithms.

A. Dynamic Networks Mining

Recently, there is an increasing interest in mining dynamic networks, such as community evolution [6, 26, 34], proximity tracking [17, 27], dynamic tensor analysis [24], and dynamic graph summarization [21]. Most of the existing works model the dynamic networks in the following three ways. (1) *Static graph*: the graph totally ignores the temporal information and aggregates everything from the beginning to the end [4, 29]; (2) *Strictly growing graph*: the graph assumes each node and edge once exist, stay forever [2, 10]; (3) *Time-evolving graph*: the graph aggregates temporal information into a sequence of snapshots [26, 27]. Although these methods permit computational convenience in the sense of various fast algorithms [27] for dynamic networks, they also impose severe limitations in capturing the dynamic structures [19] in real applications. In [19], the authors introduce the novel notion of "δ-temporal motif", which helps explore the ordering of temporal edges and characterize different types of three-node structures in temporal networks. In this paper, we further generalize this idea and propose the definition of the k^{th}-Order τ-Duration δ-Temporal Network Structure, which characterizes the temporal network structures in every occasion that the edges from a particular network structure within the prescribed τ-duration time window and last for δ time. The details of modeling temporal networks will be discussed in Section 4.

B. Graph Clustering Algorithms

Graph Clustering algorithms represent an important class of tools for studying the underlying structure of networks in various applications such as fraud detection [7, 32, 33], insider detection [31], synthetic identitiy detection [35] and community detection [16]. Nonetheless, all of the above works are known to perform clustering at the level of individual nodes and edges. More recently, how to cluster networks on the basis of higher-order connectivity patterns arouses research interests in the data mining community. In [3], the authors propose a spectral clustering framework that allows for modeling high-order networks structures and conducts partition while preserving such structures on the given graph. [35] proposed a high-order graph clustering framework, which aims to find a structure-rich dense subgraph from a user specified local region. However, to the best of our knowledge, there is no previous work about high-order network structure clustering on temporal networks. To fill this gap, *MOTLOC* is proposed to identify clusters that largely preserve the user-specified structures in temporal networks.

III. Fine-Grained Temporal Network

In this section, we introduce the formal definitions of temporal network structure and temporal networks.

A. Temporal Network

Traditional techniques model dynamic networks by collecting the changes over time and aggregating temporal information into a sequence of snapshots. However, an important observation from many real applications is that the nodes and edges are time-stamped, and they may not all exist forever. For example, in social networks, a node can be added and then disappear, when one creates a user account and deletes it later; an edge can be created and then be removed, when one follows another user and then stops the "followship". Obviously, either strictly growing graphs or time-evolving graphs fail to preserve these fine-grained dynamics of temporal networks. To tackle this issue, we propose the novel notions regarding temporal network as follows.

*Definition 1 (δ-**Temporal Node**):* In the temporal network, a δ-temporal node $v^{(t,\delta)}$ represents the vertex v that appears at time-stamp t and exists for δ duration.

*Definition 2 (δ-**Temporal Edge**):* In the temporal network, a δ-temporal edge $(u,v)^{(t,\delta)}$ represents the connection between node u and node v that appears at time-stamp t and exists for δ duration.

*Definition 3 (**Temporal Network**):* A temporal network $\tilde{G} = (\tilde{V}, \tilde{E})$ is formed by a collection of $\tilde{n}$ temporal nodes $\tilde{V} = \{v_1^{(t_1,\delta_1)}, v_2^{(t_2,\delta_2)}, \ldots, v_{\tilde{n}}^{(t_{\tilde{n}},\delta_{\tilde{n}})}\}$ and a sequence of $\tilde{m}$ temporal edges $\tilde{E} = \{(u_1,v_1)^{(t_1,\delta_1)}, (u_2,v_2)^{(t_2,\delta_2)}, \ldots, (u_{\tilde{m}},v_{\tilde{m}})^{(t_{\tilde{m}},\delta_{\tilde{m}})}\}$.

Based on the above definitions, now we introduce the notion of k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ which is designed for characterizing the higher-order connectivity patterns in temporal networks.

*Definition 4 (**k^{th}-Order τ-Duration δ-Temporal Network Structure**):* In temporal networks, a k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ represents the k-node network structure composed of an ordered sequence of l temporal edges, $\{(u_1, v_1)^{(t_1,\delta_1)}, \ldots, (u_l, v_l)^{(t_l,\delta_l)}\}$, which are all created in a prescribed τ-duration time window, i.e., $t_1 \leq t_2 \leq \ldots \leq t_l$ and $t_l - t_1 \leq \tau$, and exist at least δ duration, i.e., $\delta_1, \delta_2, \ldots, \delta_l \geq \delta$.

B. Temporal Conductance

A good cut C of static graph G is a subset of vertices that are densely connected within themselves but sparsely connected with the remainder of the graph. The quality of a cut (cluster) can be measured by conductance, the fractional ratio between the internal edges and external edges of the identified cut (cluster) C. However, in many real applications, it is usually the case that the user would like to find a local cluster C on the graph G such that: (1) the cut should preserve rich prescribed high-order network structure $\tilde{\mathbb{N}}$ inside of C; (2) the preserved high-order network structure $\tilde{\mathbb{N}}$ should satisfy some temporal constraints regarding the existing period δ and creating period τ; (3) the cut should break as less such structure $\tilde{\mathbb{N}}$ as possible. Obviously, the traditional definition of the conductance Φ does not serve this purpose. In [3], the authors introduce the definition of "high-order conductance" ϕ_3 for conducting a sweep cut with respect to triangles on a given static network. Based on the definition of the k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(\tau,\delta)}$, we further generalize ϕ_3 [3] to higher-order network structures in dynamic networks as follows.

*Definition 5 (**Temporal Conductance**):* For any given cluster C in the temporal network $\tilde{G}$ and the k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$, the k^{th}-order temporal conductance $\Phi(C, \tilde{\mathbb{N}})$ is defined as

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

where $cut(C, \tilde{\mathbb{N}})$ denotes the number of temporal structures broken due to the partition of G into C and $\bar{C}$, i.e., $cut(C, \tilde{\mathbb{N}}) = \sum_{i_1,\ldots,i_k \in \tilde{V}} T(i_1, \ldots, i_k) - \sum_{i_1 i_2,\ldots,i_k \in C} T(i_1, \ldots, i_k) - \sum_{i_1,\ldots,i_k \in \bar{C}} T(i_1, \ldots, i_k)$ and $\mu(C, \tilde{\mathbb{N}})$ $(\mu(\bar{C}, \tilde{\mathbb{N}}))$ denotes the total number of temporal structures $\tilde{\mathbb{N}}^{(\tau,\delta)}$ incident to the vertices within C $(\bar{C})$, i.e., $\mu(C, \tilde{\mathbb{N}}) = \sum_{i_1 \in C; i_2,\ldots,i_k \in \tilde{V}} T(i_1, i_2, \ldots, i_k)$ $\mu(\bar{C}, \tilde{\mathbb{N}}) = \sum_{i_1 \in \bar{C}; i_2,\ldots,i_k \in \tilde{V}} T(i_1, i_2, \ldots, i_k)$.

C. Temporal Motif-Based Graph Cut Problem

The identification of structure-rich clusters can be considered as an optimization problem: Given a temporal graph $\tilde{G}$, a user-defined structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ and a parameter ϕ, find a cluster C such that $\Phi(C, \tilde{\mathbb{N}}) \leq \phi$ or determine that no such cluster exists. The above problem is challenging since: (1) even only considering the simplest case, i.e., the user-defined structure is an edge, the above problem is NP-complete [12, 22]; (2) the network is always highly skewed, i.e., it is usually the case that the clusters of interest are rare [9]. To tackle these issues, we transform the global clustering problem into a local graph clustering problem as follows.

Problem 1: Motif-preserving dynamic local graph cut

Input: (i) a temporal network $\tilde{G} = (\tilde{V}, \tilde{E})$, (ii) a user-defined Temporal Network Structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$, (iii) a conductance upper bound ϕ; and (iv) an initial vertex v.

Output: a cluster $C \in \tilde{V}$ such that $\Phi(C, \tilde{\mathbb{N}}) \leq \phi$.

IV. The Proposed *MOTLOC* Framework

In the previous two sections, we introduce the basics of vector-based graph cut and high-order temporal network structure. Now, we are ready to present our motif-preserving dynamic local graph cut framework, which generalizes the vector-based graph cut methods to produce local clusters that largely preserve user-defined high-order temporal network structures from the graph cut.

A. High-order Representation of Temporal Network

Our first goal is to extract and represent the structures of interest in the temporal network $\tilde{G}$. [3] first introduced the idea of representing triangles in a static graph by using an "order-3 tensor". Here, we generalize this idea to represent any user-defined Structures $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ in a tensor representation of $\tilde{G}$. In general, given $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$, the corresponding adjacency tensor $\boldsymbol{T}$ can be constructed by Definition 6. For instance, to represent the temporal triangle in Fig. **??** (a), we can use a three-mode adjacency tensor $\boldsymbol{T} \in \mathbb{R}_+^{\tilde{n} \times \tilde{n} \times \tilde{n}}$, where the entry $T(i, j, k) = 1$ if and only if vertices $i, j, k \in V$ form the prescribed temporal triangle. In particular, the number of non-zero entries of $\boldsymbol{T}$ indicates how many user-defined temporal triangles exist in the given data set; the number of non-zero entries in slice $\boldsymbol{T}(i, :, :)$ represents how many the user-defined temporal triangles consist the i^{th} node in the network; the number of non-zero entries in fiber $\boldsymbol{T}(i, j, :)$ shows how many the user-defined temporal triangles are formed by the i^{th} node and the j^{th} node in the network.

*Definition 6 (**Adjacency Tensor**):* Given a temporal graph $\tilde{G} = (\tilde{V}, \tilde{E})$, the k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ on $\tilde{G}$ can be represented in a k-mode adjacency tensor $\boldsymbol{T}$ as follows

$$T(i_1, i_2, \ldots, i_k) = \begin{cases} 1 & \{i_1, i_2, \ldots, i_k\} \text{ form } \tilde{\mathbb{N}}^{(k,\tau,\delta)}. \\ 0 & \text{Otherwise.} \end{cases} \tag{2}$$

In Definition 7, we induce the notion of transition tensor with respect to $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ in $\tilde{G}$. Considering each vertex in $\tilde{G}$ as a state, we can interpret the k-mode transition tensor $\boldsymbol{P}$ as the transition probabilities of the $(k-1)^{\text{th}}$-order Markov chain, i.e., $P(i_1, \ldots, i_k) = Pr(S_{t+1} = i_1 | S_t = i_2, \ldots, S_{t-k+2} = i_k)$. In other words, given the nodes (previous states) $i_2, \ldots, i_k$, the probability of forming $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ by using node (current state) i_1 is $P(i_1, i_2, \ldots, i_k)$.

*Definition 7 (**Transition Tensor**):* Given a temporal graph $\tilde{G} = (\tilde{V}, \tilde{E})$ and the adjacency tensor $\boldsymbol{T}$ for the k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$, the corresponding transition tensor $\boldsymbol{P}$ can be computed as

$$P(i_1, i_2, \ldots, i_k) = \frac{T(i_1, i_2, \ldots, i_k)}{\sum_{i_1=1}^{\tilde{n}} T(i_1, i_2, \ldots, i_k)} \tag{3}$$

In Algorithm 1, we present a general framework for computing the temporal motif-based transition tensor $\boldsymbol{P}$. The input of Algorithm 1 consists of the temporal network $\tilde{G}$ and user-defined temporal structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$. Here, we assume that $\tilde{G}$ has edges sorted by the order of creating time, which ensures that we can access all the temporal edges in $O(1)$ in Step 5.

Algorithm 1 Pre-processing: Constructing Transition Tensor

Input: $\tilde{G} = (\tilde{V}, \tilde{E})$, $\tilde{\mathbb{N}}^{(\tau,\delta)}$.
Output: $\boldsymbol{P}$.

1: Construct the static graph G induced by $\tilde{G}$.
2: Identify all the instances of the static network structure $\mathbb{N}$ induced by $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ on static graph G.
3: **for** each identified static network structure $\mathbb{N}$ **do**
4: Gather the corresponding k distinct vertices $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$.
5: Gather the corresponding m' temporal edges and form an ordered sequence $\tilde{S}$, i.e., $\{(u_1, v_1)^{(t_1,\delta_1)}, (u_2, v_2)^{(t_2,\delta_2)}, \ldots, (u_{m'}, v_{m'})^{(t_{m'},\delta_{m'})}\}$.
6: **if** $t_{m'} - t_1 \leq \tau$ and $\delta_1, \delta_2, \ldots, \delta_{m'} \geq \delta$. **then**
7: Let $T(i_1, i_2, \ldots, i_k) = 1$.
8: **end if**
9: **end for**
10: Compute the transition tensor $\boldsymbol{P}$ by Eq. 3.

B. High-order Random Walk with Restart

Next, we introduce the basics of HRWR, which will be used for exploring the high-order organizations in given networks. In particular, HRWR extends the random walk with restart (RWR) to the higher-order Markov chain [8, 35]. Entries of the transition tensor $\boldsymbol{P}$ w.r.t. the k^{th}-Order τ-δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(\tau,\delta)}$ can be interpreted as the transition probability of a $(k-1)^{\text{th}}$-order Markov chain. More generally, for the o^{th}-order RWR, the corresponding Markov chain S describes a stochastic process that satisfies $Pr(S_{t+1} = i_1 | S_t = i_2, \ldots, S_{t-o+1} = i_{o+1}, \ldots, S_1 = i_{t+1}) = Pr(S_{t+1} = i_1 | S_t = i_2, \ldots, S_{t-o+1} = i_{o+1})$,where $i_1, \ldots, i_{t+1}$ denote the set of states associated with different time stamps. In particular, the future state (node) v_{i_1} only depends on the past o states (nodes) $v_{i_2}, \ldots, v_{i_{o+1}}$. [8] shows that, when the stationary distribution $\boldsymbol{X}$ of the o^{th}-order Markov chain (RWR) exists, it satisfies $X(i_1, i_2, \ldots, i_{k-1}) = \alpha \sum_{i_k} P(i_1, i_2, \ldots, i_k) X(i_2, \ldots, i_k) + (1-\alpha)\chi_v(i)$. where $X(i_1, \ldots, i_{k-1})$ denotes the probability of being at states $i_1, \ldots, i_{k-1}$ in consecutive time steps upon convergence of the HRWR, and $\sum_{i_1,\ldots,i_{k-1}} X(i_1, \ldots, i_{k-1}) = 1$.

While in real applications the tensor product structure in the state-space and the tensor representation stationary distribution $\boldsymbol{X}$ make the HRWR hard to be scalable in large-scale problems. Specifically, storing the stationary distribution $\boldsymbol{X}$ requires $O(n^o)$ space complexity. To overcome this scalability limitation, a commonly held assumption is the "rank-one approximation" [8, 15], i.e., $X(i_2, \ldots, i_k) = q(i_2) \ldots q(i_k)$, where $\boldsymbol{q} \in \mathbb{R}_+^{n \times 1}$ with $\sum_i q(i) = 1$. Then, we have

$$q(i_1) = \alpha \sum_{i_2,\ldots,i_k} P(i_1, \ldots, i_k) q(i_2) \ldots q(i_k) + (1-\alpha)\chi_v(i_1). \tag{4}$$

By this way, storing the stationary distribution $\boldsymbol{X}$ only requires $O(n)$. For the paper presentation purpose, we rewrite Eq. 4 in the form of Kronecker product as follows

$$\boldsymbol{q} = \alpha \bar{\boldsymbol{P}} (\boldsymbol{q} \otimes \ldots \otimes \boldsymbol{q}) + (1-\alpha)\boldsymbol{\chi_v}. \tag{5}$$

where $\otimes$ denotes the Kronecker product.

C. High-order Vector-based Graph Cut

Here, we present our high-order vector-based graph cut algorithm (Algorithm 2) for conducting a structure-preserving local graph cut in the given temporal networks. The key challenge of motif-based graph cut is the prohibitive computational cost. To tackle this, we generalize the key idea of vector-based graph cut methods [1, 23] to locally explore the high-order connectivity patterns in the neighborhood of initial vertex v and return the desired graph cut, by using HRWR [8] with "rank-1" assumption [15]. For further reduce the computational cost of HRWR, we adopt an approximation of HRWR in our framework by restricting the probability mass of HRWR to a local subset of vertices, thus we can limit the computation to the neighborhood of the seed. To achieve this, we apply the following truncation operator [23] in order to round all small probability values to zero and do not differ too much from the one without truncation.

$$[q]_\epsilon(u) = \begin{cases} q(u) & \text{if } q(u) \geq d(u)\epsilon \\ 0 & \text{Otherwise} \end{cases} \tag{6}$$

where ϵ is the truncation threshold that can be computed as $\epsilon = 1/(1800 \cdot (l+2) t_{last} 2^b)$, l can be computed as $l = \lceil log_2(\mu(V)/2) \rceil$, and t_{last} can be computed as $t_{last} = (l+1)\left\lceil \frac{2}{\phi^2} ln\left(140(l+2)\sqrt{\mu(V)/2}\right)\right\rceil$.

Intuitively, the complexity of Algorithm 2 largely depends on the density of the constructed transition tensor P, i.e., the number of $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$ in the given data. However, it is hard to directly estimate the number of high-order structures in real applications. Here, we present the time complexity analysis in Lemma 1, which shows that Algorithm 2 runs in *polylogarithmic* time with respect to the size of $\tilde{G}$.

Lemma 1: The time complexity of Algorithm 2 is upper bounded by $O\left(t_{max} \frac{2^{bk}}{\phi^{2k}} log^{3k} \tilde{m}\right)$, where k is the order of prescribed temporal network structure $\tilde{\mathbb{N}}^{(k,\tau,\delta)}$, $\tilde{m}$ is the number of temporal edges in the given temporal network $\tilde{G}$, and t_{max}, b, ϕ are the input parameters of Algorithm 2.

Proof 1: Omitted for brevity.

Algorithm 2 High-order Vector-based Graph Cut

Input: $\boldsymbol{P}$, M, v, k, ϕ, $t_{\max}$, b, c_1.
Output: C.

1: Compute the unfolding matrix $\bar{\boldsymbol{P}}$ of the transition tensor $\boldsymbol{P}$ and initial ϵ using ϕ.
2: Compute the initial HRWR distributions $\boldsymbol{r^{(t)}} = [M^{\boldsymbol{t}\ \ 1}\boldsymbol{\chi_v}]_\epsilon$, $t = 1, \ldots, k-1$.
3: Let $C = \emptyset$.
4: **for** $t = k : t_{\max}$ **do**
5: Compute the t-step HRWR vector $\boldsymbol{q^{(t)}} = \alpha\bar{\boldsymbol{P}}(\boldsymbol{r^{(t\ \ 1)}} \otimes \ldots \otimes \boldsymbol{r^{(t\ \ k+1)}}) + (1-\alpha)\boldsymbol{\chi_v}$.
6: Truncate the HRWR vectors $\boldsymbol{q^{(t)}}$ by $\boldsymbol{r^{(t)}} = [\boldsymbol{q^{(t)}}]_\epsilon$.
7: Compute the permutation π regarding the HRWR vector $\boldsymbol{q^{(t)}}$ such that $\frac{q^{(t)}(\pi(1))}{d(\pi(1))} \leq \frac{q^{(t)}(\pi(2))}{d(\pi(2))} \leq \ldots \leq \frac{q^{(t)}(\pi(n))}{d(\pi(n))}$.
8: **for** $j = \lfloor 2^{\frac{b}{2}} \rfloor : n$ **do**
9: Let $C_j = \{\pi(1), \ldots, \pi(j)\}$ be the set of the first j vertices in permutation π.
10: Quit and return $C = C_j$ such that:
11: **(a)Conductance:** $\Phi(C_j, \tilde{\mathbb{N}}) <= \phi$,
12: **(b)Volume:** $2^b <= \mu(C_j)$,
13: **(c)Prob Mass:** $I_x(\boldsymbol{q^{(t)}}, 2^b) >= \frac{1}{c_1(l+2)2^b}$.
14: **end for**
15: **end for**

V. Experimental Results

In this section, we present the experimental results of our empirical studies in terms of effectiveness and efficiency.

A. Experiment Setup

Data Sets: we test our algorithm on a diverse set of real-world temporal networks. The statistics of all data sets used are summarized in Table I.

Network	Temporal Nodes	Temporal Edges	Time Span
MathAQ	21,688	107,581	2,350 days
MathCQ	16,836	203,639	2,349 days
MathCA	13,840	195,330	2,350 days
Email	986	332,334	803 days
MSG	1,899	20,296	193 days
ETrust	46,948	245,749	3,744 days

TABLE I
Statistics of the data sets.

- *Collaborative Network:* The MathAQ, MathCQ and MathCA data sets [13] are collected from MathOverflow with respect to three different types of users' interactions, i.e., answer questions, make comments on questions, and make comments on answers.
- *Communication Network:* The Email data set [19] is a collection of emails within an European research institution. The MSG data set [18] is extracted based on a social network at the University of California, Irvine.
- *Review Network:* The ETrust data set [14] is a who-trust-whom network derived from a review network named Epinion. Each node represents a user, and one edge exits if and only if when one user trusts another user at a certain time stamp.

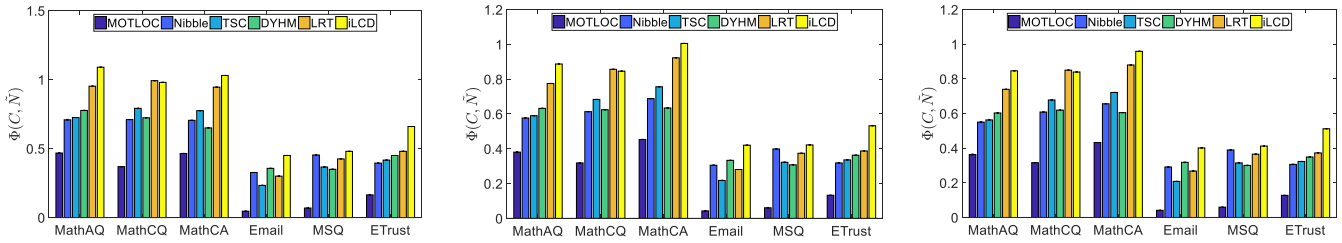
Comparison Methods: In our experiments, we select five state-of-the-art graph clustering methods, including two static methods, i.e., Nibble [23] and TSC [3], and three dynamic methods, i.e., DYHM [20], iLCD [5] and LRT [28]. Note that all the static methods are performed on the induced static graph G, that aggregates all the temporal nodes and the temporal edges into one time stamp.

B. Effectiveness Results

We perform the effectiveness comparisons with five baseline algorithms in Fig. 2, by treating temporal triangles as the user-defined structure. Note that, to evaluate the convergence of local algorithms, we randomly select 10 vertices from the same 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 (only for local algorithms) represent the standard deviation of evaluation metrics in multiple runs. We let the duration of temporal triangle be $\tau = 10$ days, 20 days and 50 days, respectively. Fig. 2(a) to Fig. 2(c) present the results regarding high-order temporal conductance $\Phi(C, \tilde{\mathbb{N}})$. In general, we have the following observations: (1) *MOTLOC* outperforms other comparison methods across all the data sets. For example, when $\tau = 10$, *MOTLOC* is 33.9% smaller than the second best, i.e., Nibble, on $\Phi(C, \tilde{\mathbb{N}})$ in Fig. 2. (2) Comparing with the existing methods without high-order structures (e.g., Nibble, DYHM, LRT, and iLCD), *MOTLOC* demonstrates significant reduction in terms of conductance and expansion with respect to different durations, showing the usefulness of high-order structures. (3) Comparing with TSC, which also uses high-order structures but did not use temporal information, *MOTLOC* shows the importance of the use of temporal information in our model.

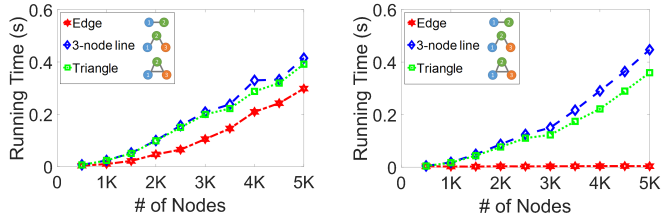
C. Efficiency Results

Here, we evaluate the scalability of *MOTLOC* with different user-defined structures on a series of increasing scale synthetic graphs. The results presented in this subsection are calculated over 500 runs. Note that the overall running time of *MOTLOC* may vary a lot with different initial vertices and data sets. Considering the most time-consuming parts in Algorithm 2 are the HRWR diffusion process (Step 7 to Step 9) and the query process (Step 10 to Step 16), we show the running time of HRWR diffusion process and query process separately as follows. In Fig. 3, we show the running time per HRWR and the running time per query vs. the number of nodes. For all the results in Fig.3, we let the edge density be fixed as 1%. In general, we observe that (1) the running time increases way slower than $O(n^k)$, i.e., a little bit above linear with respect to the number of nodes. It is because we take the advantage of sparse computation techniques in implementing our proposed algorithms; (2) the running time of both HRWR and query process is higher when the order of user-defined network structure is higher; (3) the running time of experiments regarding 3-node line is slightly higher comparing with the experiments



(a) $\tau = 10$ days (b) $\tau = 20$ days (c) $\tau = 50$ days

Fig. 2. High-order Temporal Conductance over Different Temporal Graphs. (lower is better)



(a) Running time per HRWR (b) Running time per query

Fig. 3. Scalability Analysis with Respect to Number of Nodes.

regarding triangles. One possible reason is that the transition tensor of 3-node line is denser than the transition tensor of triangle, which leads to more computational workload.

VI. Conclusion

In this paper, we study how to perform high-order structure-preserving graph cut on temporal network, which opens the door for high-order connectivity pattern analysis in dynamic systems. To the best of our knowledge, we are the first to study graph clustering in this setting. The major contributions of this paper include the following aspects. First, we develop the definitions of temporal network $\tilde{G} = (\tilde{V}, \tilde{E})$ and the k^{th}-Order τ-Duration δ-Temporal Network Structure $\tilde{\mathbb{N}}^{(\tau,\delta)}$ as the tool for modeling and characterizing the underlying high-order temporal structure of dynamical systems. Besides, we propose a general algorithm for encoding user-defined high-order temporal network structure patterns into the tensor representation of the given network. Furthermore, we develop the motif-preserving dynamic local graph cut framework *MOTLOC*, which gives the user flexibility to model any high-order network structures in temporal networks. Finally, extensive experiments on both real and synthetic data sets demonstrate the effectiveness and efficiency of the proposed algorithms.

Acknowledgment

This work is supported by the United States National Science Foundation under Grant No. IIS-1552654, Grant No. IIS-1813464, Grant No. PFI:BIC-1430144, and Grant No. CNS-1629888, the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-02-00, and an IBM Faculty Award. 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] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *IEEE FOCS*, 2006.
[2] A. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 1999.
[3] A. R. Benson, D. F. Gleich, and J. Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *SIAM SDM*, 2015.
[4] A. R. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 2016.
[5] R. Cazabet, F. Amblard, and C. Hanachi. Detection of overlapping communities in dynamical social networks. In *SocialCom*. IEEE, 2010.
[6] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *ACM SIGKDD*, 2007.
[7] K.-R. Choo. *Money laundering risks of prepaid stored value cards*. Australian Institute of Criminology, 2008.
[8] D. F. Gleich, L.-H. Lim, and Y. Yu. Multilinear pagerank. *SIMAX*, 2015.
[9] J. He. *Rare category analysis*. Carnegie Mellon University, 2010.
[10] A. Z. Jacobs, S. F. Way, J. Ugander, and A. Clauset. Assembling thefacebook: Using heterogeneity to understand online social network assembly. In *ACM Websci*, 2015.
[11] C. Klymko, D. Gleich, and T.-G. Kolda. Using triangles to improve community detection in directed networks. *arXiv preprint arXiv:1404.5874*, 2014.
[12] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 1999.
[13] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.
[14] J. Li, X. Hu, L. Jian, and H. Liu. Toward time-evolving feature selection on dynamic networks. In *IEEE ICDM*, 2016.
[15] W. Li and M. K. Ng. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra*, 2014.
[16] F.-D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 2013.
[17] N. Ohsaka, T. Maehara, and K. Kawarabayashi. Efficient pagerank tracking in evolving networks. In *ACM SIGKDD*, 2015.
[18] P. Panzarasa, T. Opsahl, and K. M. Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology*, 2009.
[19] A. Paranjape, A. R. Benson, and J. Leskovec. Motifs in temporal networks. 2017.
[20] Y. Park, C. Moore, and J. S. Bader. Dynamic networks from hierarchical bayesian graph clustering. *PloS one*, 2010.
[21] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *ACM SIGKDD*, 2015.
[22] J. Šíma and S.-E. Schaeffer. On the np-completeness of some graph cluster measures. In *SOFSEM*. Springer, 2006.
[23] D.-A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SICOMP*, 2013.
[24] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *ACM SIGKDD*, 2006.
[25] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *ACM SIGKDD*, pages 807–816, 2009.
[26] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *ACM SIGKDD*, 2007.
[27] H. Tong, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SIAM SDM*, 2008.
[28] J. Xie, M. Chen, and B. K. Szymanski. Labelrankt: Incremental community detection in dynamic networks via label propagation. In *DNMM*, 2013.
[29] Ö. N. Yaveroğlu, N. Malod-Dognin, D. Davis, Z. Levnajic, V. Janjic, R. Karapandza, A. Stojmirovic, and N. Pržulj. Revealing the hidden language of complex networks. *Scientific reports*, 2014.
[30] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. In *ACM SIGKDD*, 2017.
[31] D. Zhou, J. He, K.-S. Candan, and H. Davulcu. Muvir: Multi-view rare category detection. In *IJCAI*, 2015.
[32] D. Zhou, J. He, Y. Cao, and J. Seo. Bi-level rare temporal pattern detection. In *IEEE ICDM*, 2016.
[33] D. Zhou, J. He, H. Yang, and W. Fan. Sparc: Self-paced network representation for few-shot rare category characterization. In *ACM SIGKDD*. ACM, 2018.
[34] D. Zhou, K. Wang, N. Cao, and J. He. Rare category detection on time-evolving graphs. In *IEEE ICDM*, 2015.
[35] D. Zhou, S. Zhang, M. Y. Yildirim, S. Alcorn, H. Tong, H. D., and J. He. A local algorithm for structure-preserving graph cut. In *ACM SIGKDD*, 2017.