A SUPPLEMENTARY MATERIAL

A.1 PARAFAC Decomposition

Our work is inspired by the tensor decomposition. A PARAFAC decomposition is one way to decompose a tensor into multiple components via a high-order singular value decomposition (HOSVD). It has been applied in many fields and a good reference is the survey from [5]. Given a three-mode tensor $\mathcal{X}$, a rank-R PARAFAC decomposes $\mathcal{X}$ into R components, each of which consists of a scalar λ_r and a rank-1 tensor produced by the outer product of 3 column vectors $A_r \otimes B_r \otimes \mathcal{T}_r$, with the objective of minimizing the Frobenius norm of the error:

$$f(\{\lambda_r, A_r, B_r, \mathcal{T}_r\}_{r=1}^R) = \|\mathcal{X} - \hat{\mathcal{X}}\|_{\mathbf{F}} = \sqrt{\sum_{i,j,k}(\mathcal{X}_{ijk} - \hat{\mathcal{X}}_{ijk})^2} \tag{9}$$

where $\hat{\mathcal{X}} = \sum_{r=1}^R \lambda_r A_r \otimes B_r \otimes \mathcal{T}_r$, $A = [A_1, \cdots, A_R], B = [B_1, \cdots, B_R]$ and $\mathcal{T} = [\mathcal{T}_1, \cdots, \mathcal{T}_R]$ are call loading matrices. The PARAFAC decomposition model can be written as:

$$\mathcal{X}_{ijk} \approx \sum_{r=1}^R \lambda_r a_{ir} b_{jr} t_{kr} \tag{10}$$

where a_{ir} is the i-th element of A_r, b_{jr} is the j-th element of B_r and t_{kr} is the k-th element of $\mathcal{T}_r$.

Previous work [12] applied PARAFAC decomposition to detect communities in a small network. In our experiment, however, two unique communities may be mistakenly grouped as one even nodes between communities have no edges when using their method. It remains unclear how the rank R or the time granularity affect the accuracy of communities detection.

A.2 Processing Edge Generation Rate As Time Series

we assume $\vec{\lambda_r}$ includes noise, δ, that is Gaussian, $\delta \sim N(\bar{\delta}, \sigma)$ and causes error E_d. We use $\bar{\delta} + 3\sigma$ as threshold for $E_{\max}$ because 99.7% of δ's values lie within three standard deviations of $\bar{\delta}$. The next step is to estimate the mean and variance of δ. We apply a Sliding Window Filter to $\vec{\lambda_r}$ to compute $\hat{\lambda}_r(t)$, then obtain noise samples $\delta_j = \hat{\lambda}_r(j) - \lambda_{jr}$ for $j = 1, \cdots, T$, and calculate the mean $\bar{\delta} = \frac{1}{T}\sum_{j=1}^T \delta_j$ and standard error $\hat{\sigma} = \sqrt{\frac{1}{T-1}\sum_{j=1}^T(\delta_j - \bar{\delta})^2}$.

peicewise linear regression for edge generating rate We propose an approach based on time series segmentation with linear regression to construct $\lambda^{(r)}(t)$: given an integer D and a time series $\vec{\lambda}_r = [\lambda_{1r}, \cdots, \lambda_{Tr}]$, let π_D be a segmentation consisting of D time segments Q_d on time interval $[1, T]$. We define a piecewise linear model:

$$f_D(t) = \begin{cases} b_1 t + c_1, & t \in Q_1 \\ b_2 t + c_2, & t \in Q_2 \\ \vdots & \vdots \\ b_D t + c_D, & t \in Q_D \end{cases} \tag{11}$$

The objective is to find optimal $M_D^* = \{\pi_D, f_D^*\}$ that minimizes the squared error

$$e(\vec{\lambda}, M_D) = \sum_{j=1}^T (\lambda_{jr} - f_D(j))^2 \tag{12}$$

Then $\lambda^{(r)}(t)$ is represented by $f_D^*(t)$ in the solution M_D^*.

Note that $C(M_D^*)$ is sensitive to D: in the extreme case, the cost is 0 if $D = T-1$ where each time step is a segment; at the other extreme when $D = 1$, it corresponds to the error of a linear regression on the entire time series. The most common methods use bottom up algorithms with a stopping criterion to set D. A typical algorithm begins with $T-1$ segments and obtains a solution M_{T-1}. At each iteration, it merges two neighboring segments so as to minimize the increase in cost (denoted as $\Delta C(D) = e(\vec{\lambda}, M_{D-1}^*) - e(\vec{\lambda}, M_D^*)$).

Let E_d be the squared error of using $f_D(t)$ to approximate $\vec{\lambda}$ in time segment Q_d, for $d = 1, \ldots, D$. One stopping criterion [15] defines a maximum error $E_{\max}$ such that $E_d \le E_{\max}$, for $d = 1, \cdots, D$. However, it is difficult to choose the of threshold $E_{\max}$ for a given time series.

Finally, the *time series segmentation problem* can be formalized as:

$$\min D \quad \text{s.t.} \quad E_d \le (\bar{\delta} + 3\hat{\sigma})^2 \quad \text{for } d = 1, \ldots, D \tag{13}$$

Algorithm 1: Time Mode Segmentation Algorithm

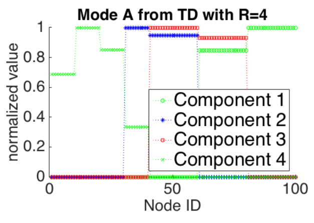
```
Data: Time Series $Y = [y_1, \cdots, y_T]$
Result: Segmentation $Q = [Q(1), \cdots, Q(D)]$, $f_D(x)$
$Q = [[1], [2], \cdots, [T]]$; $\hat{Y}$ = SlideWindowFilter($Y$);
$\Delta = Y - \hat{Y}$; $\bar{\delta}$ = mean($\Delta$); $\hat{\sigma}$ = standardError($\Delta$);
for $i = 1 : T - 1$ do
    merge_cost(i) = LinearRegressionError(Y[i:i+1]);
while min(merge_cost)$\le (\bar{\delta} + 3\hat{\sigma})^2$ do
    i = indexOf(min(merger_cost));
    Q(i) = merge(Q(i),Q(i+1));
    delete(Q(i+1);
    merge_cost(i) = LinearRegressionError(merge(Q(i),
      Q(i+1));
    merge_cost(i-1) =
      LinearRegressionError(merge(Q(i-1), Q(i));
```

In Algorithm 1, the function LinearRegressionError($Y[i:j]$) calculates the linear regression $f_D(x)$ for a segment $Q_d = Y[i:j]$ and returns the error. The edge-generating rate $\lambda^{(r)}(t)$ can be constructed as $f_D(x)$ from the Algorithm 1 for $d = 1, \ldots, D$.

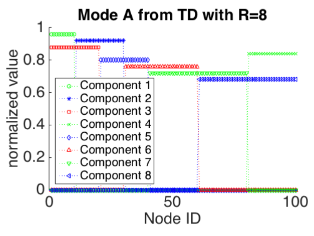
A.3 Silhouette Clustering Criterion

Assume that we have K clusters $\{C_k\}, (k = 1, \cdots, K)$ and a similarity function $s(i, j)$ for data points i and j (in our model, a data point is a node). The Silhouette score graphically measures how well a point lies within its cluster [26]. The similarity of a node i to a cluster C_k is defined as

$$d(i, C_k) = \sum_{j \in C_k} s(i, j)/|C_k| \ .$$



(a) Mode A w/ R=4



(b) Mode A w/ R=8

Figure 7: Plot of Mode A: (a) small R combines clusters with common nodes into one component; (b) proper R put each cluster in a separate component.

Suppose that node i is in cluster C_k, then the *Silhouette score* of i is

$$S(i) = \frac{d(i, C_k) - \max_{l\neq k} d(i, C_l)}{\max_m d(i, C_m)} \in [-1, 1]$$

Here, $S(i)$ measures how well i belongs to its own cluster, with a positive value indicating a good fit.

Finally, the average Silhouette score over all data points, $\sum_{i=1}^{|V|} S(i)/|V|$, is a measure of the quality of the K clusters. The Silhouette criterion is to choose K with the highest average Silhouette score.

Suppose $K^{(r)}$ clusters are detected in the generative model $X^{(r)}$, with the m-th cluster denoted as $C_m^{(r)}$. K-means clustering in our framework can produce a cluster $C_m^{(r)}$ where $\forall i, j \in C_m^{(r)}, s(i, j) \approx 0$,. This means that $C_m^{(r)}$ generates almost no edges. Such clusters are not meaningful and will be removed by a cluster ranking step.

A.4 Baseline Methods

Evolutionary Clustering

Let $\mathcal{X}_t = \mathcal{X}_{(:,:,t)}$ be a similarity matrix for graph G_t of N nodes. EC performs clustering on the similarity matrix $\mathcal{R}_t = (1-\beta)\mathcal{X}_t + \beta\mathcal{X}_{t-1}$ at time t to obtain a clustering $\mathcal{C}_t = [\mathcal{C}_{1,t}, \dots, \mathcal{C}_{N,t}]$, where $\beta \in [0, 1]$ and $\mathcal{C}_{i,t}$ is the index of the cluster that node i belongs to at time t. Since $\mathcal{R}_t$ differs from the current similarity matrix $\mathcal{X}_t$, $\mathcal{C}_t$ is not necessarily the best clustering for $\mathcal{X}_t$, a snapshot quality function $sq(\mathcal{C}_t, \mathcal{X}_t)$ is used to evaluate the clustering result. EC includes a historical cost function $hc(\mathcal{C}_t, \mathcal{X}_{t-1})$ to calculate the difference between current and previous clustering results. The objective is to minimize

$$\sum_{t=1}^{T} sq(\mathcal{C}_t, \mathcal{X}_t) - c\sum_{t=2}^{T} hc(\mathcal{C}_t, \mathcal{X}_{t-1})$$

where c is a scalar.

PARAFAC with BC considers each component $X^{(r)}$ as a community. It chooses a threshold for binary classification: For a node i, if a_{ir} from the loading vector A_r is larger than the threshold, then i belongs to that community.

A.5 Effect of R on clustering

The number of latent structures, R, affects greatly the error of objective function in a model. In our model, R is the number of generative models and in PARAFAC decomposition, the number of components. To our knowledge, the effect of R have not yet been discussed in community detection with tensor factorization.

We begin with a toy problem to show how R affects clustering in our model. We create a dynaimc graph with 100 nodes. For the first 50 time steps, nodes 1-10, 11-30, 31-60 and 61-100 form four disjoint cliques (cl1; cl2; cl3 and cl4). Then for the next 50 steps, nodes 1-20, 21-40, 41-80 and 81-100 form another set of disjoint cliques (cl5; cl6; cl7 and cl8). $\mathcal{X}_{ijk} = 1$ if node i, j belongs to the same clique at time step k, otherwiese $\mathcal{X}_{ijk} = 0$.

We set $R = 2 - 16$. When R larger than the number of ground truth components, denoted as K, a network is over-factorized. Previous work [3] has investigated components in this situation and proposed a regularization method to improve on ALS algorithms to recover ground truth components with small error. Extra components from the result usually have very small norm. As a result, communities from these components have low SO scores and are filter out by our model. Thus, we only focuses on $R \leq K$. Some results are shown in Figure 7 (More results are provided in supplementary material). When R=4 (Fig 7(a)), clusters with common nodes are combined into one component. For example, component 4 plotted with "x" represents a mixture of cl1, cl2, cl5 and cl6. These cliques are hard to distinguish because small R causes information loss, although subgroups can be detected. When R=8, each component contains only one clique. Fig 7(b) shows the plots of A_r, where $r = 1, \cdots, 8$. The value of A_r is normalized to $[0, 1 - 0.04r]$ for clear illustration.

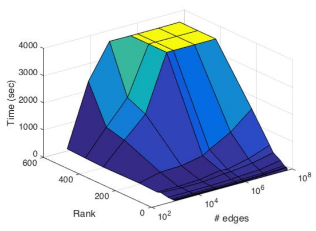
In summary, the effect of R on clustering is:
(1)If R is smaller than the number of clusters, independent clusters with similar norm value, or clusters with common nodes tend to combine into one component. Cluster detection may become difficult in this case. Clusters with small norm are likely to be eliminated.
(2)If R is large, a component tend to contain fewer clusters. Ground truth clusters are more easily detected.

A.6 Running Time

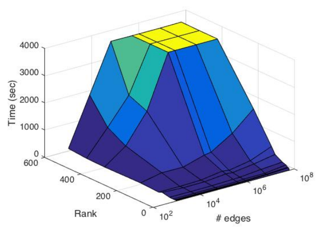
The time and space complexities to run our model and PARAFAC with BC are both affected by the iteration of the ALS algorithm, number of generative models/components and the size of the network (for sparse network, the number of edges). We choose R from 2 to 600 and compute on temporal networks with number of edge ranging from 10^3 to 10^8. We set the maximum number of iteration of ALS to 10^4 and terminate the algorithm if the change in the error is smaller than 10^{-8}.

It turns out that the average running times are similar for both methods (Fig 8). However, when network size is large, running time increase significantly. When the number of edges is large than 10^6 and $R > 400$, ALS terminates because the maximum number of iterations is reached.

The state of the art algorithm [14, 24, 31] try to deal with networks with large size with the trade-off to reduce R. For



(a) PARAFAC with BC



(b) TC

Figure 8: Average running times as a function of R and # number for both algorithm are similar. Running time increases significantly for large network. ALS terminates because the maximum number of iterations is reached when $R > 400$ and # edges are larger than 10^6.

example, Jeon et. al [14] design a method decompose billion scale network with $R = 10$. Since we observe that R greatly affects clustering performances in Section A.5, we focuses on community detection when R is small.