Detecting Cluster with Temporal Information in Sparse Dynamic Graph

Kun Tu Bruno Ribeiro Ananthram Swami Don Towsley

Abstract

Detecting and keeping track of clusters in a dynamic graph is challenging, especially when all snapshots are too sparse to obtain any meaningful clusters. Instead of clustering at each snapshot, we propose a temporal clustering framework based on PARAFAC Decomposition to detect clusters and temporal information on how they change. We provide a deeper understanding of PARAFAC from clustering point of view. We apply our method to both synthetic and real-world data, compare the result with state of the art and show that our methods can detect clusters as well as their meaningful moment.

1 Introduction

Clustering in dynamic graphs \( \{G_t(V, E_t)\}_{t=1, \ldots, T} \) has been a popular task for knowledge discovery. With correctly defined similarity/distance, a cluster usually suggest “meaningful” communities where nodes in the graph are more densely connected. For real-world social networks or communication networks, these communities usually represent groups of close friends or frequent contacts. The task of detecting clusters and keeping track how they change is helpful for mining the real-world data.

Commonly asked research questions for this task are: what clusters can we get from a dynamic network? How do we know the cluster is worth investigating? How do we track those clusters overtime? And most importantly, what hidden knowledge can we get from the clusters given background information of the data?

This task is challenging because of several issues: first, snapshots \( G_t \) of real-world dynamic graphs with fine temporal granularity are sparse. The information in the sparse graph snapshots is incomplete for researchers to obtain clusters with useful knowledge. Second, even if a cluster is obtained from the dynamic graph, it is hard to keep track of its temporal information in sparse snapshots. Third, the number of snapshots of dynamic graph is huge, potentially resulting in too many clusters. It will be difficult to decide which clusters are worthy of investigation.

These issues can be formalized as following problems for temporal clustering: Given a dynamic graph \( \{G_t(V, E_t)\}_{t=1, \ldots, T} \), (1) design a similarity function \( s(i, j, G_t) \) between nodes \( i, j \in V \) at snapshot \( G_t \) and generate a set of clusters \( \{C_s \subset V\} \) (2) Generate a rank list of clusters with a scoring function
S(C_s, T_s). (4) For each cluster C_s, return a set of time periods \( \{P_i = [t_i-1, t_i]\}\) when C_s is forming, dissolving or remaining stable.

To solve the problems, we studied rank-\( R \) tensor decomposition (TD)\cite{2, 10, 9} and designed a framework to discover temporal clusters in a sparse dynamic graph. Our contribution includes: (1) Propose a temporal clustering framework to obtain clusters in a sparse dynamic network. (2) Automate detection of best number of clusters, with Silhouettes criteria, for K-means clustering on TD results. (3) Automate segmentation of time series that measure similarity within clusters and return time periods when the similarity within a cluster changes or remain stable.

The organization of our paper is as follow: Section 2 gives an overview of previous work that try to solve this problem. Section 3 introduces PARAFAC decomposition and Silhouette that are applied in our work. Section 4 introduce our framework of temporal clustering for dynamic graph. Section 5 are experiments with both synthetic data and real data to test our method.

2 Related Work

Previous work such as \cite{5} solve the issues by merging snapshots into a non-sparse one. In practice, however, it is difficult to decide the merging window length since we have no priority knowledge of the cluster membership. Evolutionary Clustering\cite{4, 12} clustering on a similarity matrix \( M_t \) created from the snapshot at each time step \( t \), with historical clustering information so that current clusters should be similar to the ones in previous time steps. But again, it is hard to know how much historical information should be used or how to do clustering on similarity matrix \( M_1 \) at the 1st time step.

Another related work \cite{8} applied TD directly on a dynamic graph for a small social network in a primary school (detail knowledge of TD is explained at section 3). However, for each component, they use “binary classification” (BC) to simply classify nodes into 2 classes, and admit that it is only for their data, but not suitable for community detection of other data when a TD with small rank combines multiple clusters into one component.

Our work is an improvement on \cite{8} in that: (1) we apply K-means clustering with Silhouettes criterion to detect clusters in a component; (2) define a ranking score for each cluster to avoid large false positive rate when recovering the ground truth clusters.(3) We study time series for clusters and give information on how a cluster changes.

3 Preliminaries

3.1 PARAFAC Decomposition for Dynamic Graph

Tensor decomposition (TD) decomposes an \( N \) mode tensor into multiple components via high-order singular value decomposition. It has been applied in many fields and a good reference is the survey from \cite{1}.

In our problem, a dynamic graph is converted to multiple snapshots of similarity matrices, which can be represented as a 3 mode tensor \( X \in \mathbb{R}^{I \times J \times T} \), where \( X_{i,j,t} \) represents the similarity between node \( i \) and node \( j \) at time \( t \). The first 2 modes in the tensor represent nodes and the third mode represents time.
We use rank-$R$ PARAFAC decomposition \cite{2} to analyze a dynamic graph. Given a 3 mode tensor $X$ and rank $R$, PARAFAC decomposes $X$ into $R$ components, each of which consists of a scalar $\lambda_r$ and a rank-1 tensor produced by the outer product of 3 column vectors $A_r \otimes B_r \otimes T_r$ (Fig 1), with the objective function of minimizing the Frobenius Norm of the error:

$$f(\{\lambda_r, A_r, B_r, T_r\}_{r=1}^R) = \min_{\{\lambda_r, A_r, B_r, T_r\}_{r=1}^R} ||X - \hat{X}||_F$$  \hspace{1cm} (1)$$

The decomposition by PARAFAC can be written as:

$$X_{i,j,t} \approx \sum_{r=1}^R \lambda_r a_{ir} b_{jr} t_{tr} \hspace{1cm} (2)$$

where $a_{ir}$ is the $ith$ element of $A_r$. Depending on the physical interpretation of the data, $a_{ir}$ relates to some property of the $ith$ node in the first mode. For example, if $a_{ir}$ is constrained to be non-negative and normalized between $[0,1]$, it serves as a measurement of the probability of node $i$ belonging to component $r$. If negative values are allowed, $a_{ir}$ can be interpreted as the attitude of node $i$ towards component $r$, where a positive value means “supportive”, 0 means “neutral” and a negative value means “against”. Similarly, $b_{jr}$ provides information for the $jth$ node in the second mode and $t_{tr}$ is related to the overall connectivity between nodes at the $tth$ time step.

Usually, $a_{ir}, b_{jr}$ and $t_{tr}$ are normalized such that $||A_r \otimes B_r \otimes T_r|| = 1$. $\lambda_r$ is the Frobenius norm of component $r$. We use matrices $A = [A_1, \cdots, A_R], B = [B_1, \cdots, B_R]$ to represent the modes of nodes and $T = [T_1, \cdots, T_R]$ for mode of time.

In our framework, tensor decomposition requires non-negative constraint $\lambda_r, a_{ir}, b_{jr}, t_{tr} \geq 0$ and constraint $A = B$.

### 3.2 Silhouettes Clustering Criteria

Since clustering is unsupervised learning and often applied to data without ground truth, criteria such as Silhouettes criterion \cite{13, 6} and Bayesian Information criterion (BIC) \cite{14} have been propose to evaluate how well data are grouped. We use the Silhouettes criteria because information criteria like BIC require probability models.

The Silhouette score measures how well a point lies within its cluster. Given $K$ clusters $\{C_k\}, (k = 1, \cdots, K)$, a distance function $d(i,j)$ for data points $i, j$, the average distance of $i$ to other data points in its own cluster $C_k$ is

$$a(i) = \frac{\sum_{j \in C_k \setminus \{i\}} d(i,j)}{|C_k|}.$$  \hspace{1cm} (3)$$

The shortest average distance of $i$ to other cluster $C_l$ is

$$b(i) = \frac{\sum_{j \in C_l \setminus \{i\}} \min_{\{k \neq l\}} d(i,j)}{|C_l|}.$$  \hspace{1cm} (4)$$

The Silhouette score is

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$  \hspace{1cm} (5)$$

A high Silhouette score indicates that $i$ is well clustered.
$$b(i) = \min_l \left\{ \frac{\sum_{j \in C_l} d(i,j)}{|C_l|} \right\}, l \neq k.$$ The Silhouettes score of data point $i$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max \{a(i), b(i)\}} \in [-1, 1]$$

$s(i)$ measures how well $i$ belongs to its own cluster: when $i$ is reasonably clustered, we have $b(i) > a(i)$, thus $s(i) > 0$. An extreme case is that every node in $C_k$ is exactly the same ($d(i,j) = 0$), then $a(i) = 0$ and $s(i) = 1$.

Finally, to describe the similarities of nodes within a cluster, we have the Silhouettes score of a cluster $C_k$:

$$s_{C_k} = \frac{\sum_{i \in C_k} s(i)}{|C_k|} \tag{3}$$

## 4 Temporal Clustering for Dynamic Graphs

This section introduced steps in our temporal clustering framework to obtain potential meaningful clusters: 1) Construct tensor with. 2) After applying PARAFAC and obtain components, we will show how to use k-means to generate clusters from $A_r$ in component $r$. 3) We discuss the effect of $R$ on clustering result and provide a way to select components that represents a cluster. With the assumption that a meaningful cluster has longer duration or members with larger similarity, we define a score to order the clusters. 4) Since clusters are expected to change over time, we detect phases in the time series representing a forming/disappearing or stable period of a cluster by solving a segmentation problem.

### 4.1 Generating Tensor for Dynamic Graph

**Similarity Measure**

There are multiple measures for similarity such as cosine and Gaussian kernel. The choice of similarity metric strongly affects the resulting clusters. We choose commonly used similarity functions in graphs: (1) Gaussian similarity: $X_{ijt} = \exp \left(-\frac{||x_{i,t} - x_{j,t}||^2}{\sigma^2} \right)$, where $x_{i,t}$ is the feature vector of node $i$ at time $t$ and $|| \cdot ||$ is L2 norm. This similarity is suitable for mobility data where a node’s location is recorded (2) Binary similarity given threshold: $X_{ijt} = 1$ if $||x_{i,t} - x_{j,t}|| < \alpha$ or if an undirected edge $e(i,j)$ exists at time $t$ in a graph with $X_{iit} = 1$. Parameters $\sigma$ and $\alpha$ affect the clustering results and must be carefully chosen.

We will show later that Gaussian similarity enable time series $T_r$ from PARAFAC to provide more detail on the change of a cluster, while binary similarity measures the number of members in a cluster at time $t$.

**Granularity of Time** The choice of time-interval (how the original K snapshots are merged into T snapshots) is crucial. If $T$ is large, snapshots are likely to be very sparse; on the other hand, if $T$ is very small, dynamics of cluster changes will be blurred. As a specific example, consider the problem of identifying groups based on email or message exchanges. If the time interval is too short, only pair-wise behaviors can be observed; if it is too long, all the users appear as one group.
Our framework bypass the time granularity issue. Instead of local similarity at a snapshot, PARAFAC takes the global similarities overtime into account. There is a trade-off though: finer granularity incurs larger computational time cost for PARAFAC, causes noises in time mode but maintains detailed information about changes in cluster members; coarse granularity costs less time but loses information. We will explore this trade-off in section 5.

4.2 Clustering in a PARAFAC Component

After applying a rank-$R$ PARAFAC to tensor $X$, we obtain $R$ components, including scalar $\lambda_r$, column vector $A_r$, $B_r$ and $T_r$, where $r = 1, \cdots, R$. Note that the tensor represents similarities between nodes through out the time. $A_r \otimes B_r$ can be interpreted as a similarity matrix, so we can obtain clusters by running clustering algorithm on it. $T_r$ then measure the similarity within a cluster through out the time. Since snapshots of the graph are symmetric, $A = B$ and clustering can be performed on each vector $A_r$ in component $r$.

We use K-means algorithm in this paper because it is scalable and easy to show our idea of temporal clustering. Moreover, with properly defined distance function, we can obtain results close to our assumption on the data.

The clustering problem by K-means is defined as: Given $N$ data points $(x_1, \cdots, x_N)$, a number $K$ and distance function $d(i, j)$, partition the data points into $K$ clusters $\{C_k\}$ so as to minimize objective function $\sum_{k=1}^{K} \sum_{x \in C_k} d(x, \mu_k)$ where $\mu_k$ is the mean of data points in $C_k$.

We apply K-means, with $K = 1, \cdots, K_{\text{max}}$, on each $A_r$, evaluate the results with average Silhouettes (Eq3) and choose $K$ with best score. $K_{\text{max}}$ can be set to small value. Ideally, we should have 2 clusters from each $A_r$, one should be meaningful and the other meaningless (with $a_{ir} \approx 0$). The time series $T_r$ describes the change of similarity of the meaningful one. However, TD sometimes would combine multiple clusters into one. This is usually because the rank $R$ for decomposition is small. In this case, it could be hard to distinguish the combined clusters, the time series cannot provide accurate temporal information for any of the clusters. One way to solve this is to increase $R$ to see if the clusters will be split to multiple components.

4.3 Ordering Meaningful Clusters

After clustering on all the components from TD, we obtain a set of clusters $\{C_k^{(r)} | r = 1, \cdots, R; k = 1, \cdots, K_r\}$, together with $R$ time mode vectors $T_r$, where $K_r$ is the number of clusters generated from component $r$ and $C_k^{(r)}$ is the $k$th cluster from that component. As previous section pointed out, however, not all $C_k^{(r)}$’s contain knowledge on the graph. And we only need clusters that are meaningful. A reasonable way is to order the clusters with some criterion. Previous work use $\lambda_r$ value or core consistency to rank components because the top component usually contain a significant community structure in a graph. The cluster inside a component then can be order. However, the ranking might be incorrect if a component contains multiple clusters in it.

The similarities between nodes is the main property of a cluster, and thus a good candidate as a criterion to order clusters.
Note that in Equation 2 from PARAFAC decomposition, the similarity $X_{i,j,t}$ of nodes $i, j$ at time $t$ is decomposed into $R$ elements $\{\lambda_r a_{i,r} b_{j,r}, t_{kr}\}, r = 1, \cdots, R$. Large $\lambda_r a_{i,r} b_{j,r}, t_{kr}$ means node $i, j$ are similar in component $r$. With this intuition, the $j$th cluster’s average similarity overtime score $SO_{j}^{(r)}$ can be defined as:

$$SO_{j}^{(r)} = \lambda_r \frac{1(t_{kr} > 0)(\sum_k \sum_{n \in C_{j}^{(r)}} t_{kr}) P_r(m, n)}{|C_{j}^{(r)}|^2}$$

(4)

where $P_r = A_r \otimes B_r$ and $C_{j}^{(r)}$ is the $j$th cluster in component $r$. $m, n \in C_{j}^{(r)}$ and $t_{kr}$ be the $k$th positive value in time mode $T_r$.

We call the ranking with the score $SO_{j}^{r}$ as Similarity Overtime Ranking (SO ranking). It is in favor of clusters in components with large $\lambda$ values or with either high average similarity or long duration.

Under PARAFAC decomposition, if node $i$ is irrelevant to component $r$, $a_{i,r}$ or $b_{i,r}$ is usually very small, resulting a small value in $P_r(i, n)$, where $n = 1, \cdots, I$. If a cluster has low score, we consider it to be “meaningless”.

### 4.4 Automating Phase Detection

In our temporal clustering framework, we want to track changes of a cluster such as how and when it forms, how it maintains stability or disappears. We assume that the changes of a cluster’s state can be measure by the similarity of that cluster (defined as the average similarities between nodes within it). The motivation comes from a real world observation: people in communication network who frequently exchange messages with one other are likely to belong to the same community. An increase in message exchange could indicate the community come into existence or there is a heated discussion. If the similarity between people is defined as the number of their exchanged messages, then the changed of similarity can reflect the change of the network structure.

We discovered that if a component represents a cluster, the change of value in the time mode has the same trend as the cluster’s similarity. Detecting a forming state of a cluster is equivalent to detect a time phase when the value in the time mode is increasing. Similarly, a time phase with decreasing value indicates a disappearing state and a phase with relatively stability and highest values is a stable state.

For a time period in the time mode, we label a cluster being in one of the states 1) formation (with value of average similarity increasing in this time period), 2) dissolution (with value decreasing), 3) sustenance (with stable value) and 4) non-exist (value close to zero).

Thus, we propose an approach based on time series segmentation with linear regression to detect those states: given a constant $s$ and a time series $T_r = [t_1, t_2, \cdots, t_T]$, let $\pi_s$ be a segmentation with $s$ segments $P_i = [t_{a_i}, t_{a_i+1}-1]$ and $f_i(x) = b_i x + c_i$ is a linear model on $P_i$, where $\{a_i\} \subset T$, $a_1 = 1, a_{i+1} = t_T + 1$ and $i = 1, \cdots, s$. Given $s$, our objective function is to find $M = \{\pi_s, \{f_i\}\}$ to get a minimum cost:

$$C(s) = \arg \min_M (\sum_i E_i)$$

(5)

where $E_i = \sqrt{\sum_{j=a_i}^{a_{i+1}-1} (t_j - f_i(j))^2}$ is the least square error of $f_i(x)$ on $P_i$. 


Note that the objective function deeply depends on the number of segments $s$ (actually a non-decreasing function): in extreme cases, the cost would be 0 if $s = T - 1$ where each unit time interval is a segment; if $s = 1$, it would be the error of a linear regression on the whole time series. It is hard to decide the value of $s$, the most common methods avoid setting $s$ by using a bottom up algorithm with stopping criterion. It begins with each time step $t$ as a segment. For each iteration of the algorithm, it merges two neighboring segments such that the increase of cost of reducing segment number by 1, $\Delta C(s) = C(s) - C(s + 1)$ is minimum.

One criterion [11] defines a maximum error $E_{\text{max}}$ and add a constraint to Equation 5 that $E_i \leq E_{\text{max}}$, where $i = 1, \cdots, s$. However, it is still hard to manually choose a “reasonable value” of threshold given different time series.

Since our clustering model aims to automate the detection of time of cluster change for different dynamic networks, the two popular methods mentioned above are not suitable. We assume that the whole time series can be approximately modeled as $T_r(t) + \delta$, where $\delta$ is a noise of normal distribution $N(\bar{\delta}, \sigma)$. We can use $\delta$ to decide the max error of a linear model $f_i(x)$ on $P_i$. Considering the extra cost of approximating $T_r$ with $f_i(x)$, we use $\bar{\delta} + m\hat{\sigma}$ as threshold for the maximum standard error of $f_i(x)$ on its segment in a bottom up algorithm, where $m = 1, \cdots, 4$. The purpose of scaler $m$ is to improve the accuracy of segmentation that is affected by different magnitudes of noise that we encounter in our experiments.

Now the next step is to obtain the distribution of $\delta$. The Sliding Window Filtering in signal processing can be applied to obtain $T_r(t)$ and then get $\delta$ samples: $\delta_j = T_r(j) - t_j$, and calculate the mean $\bar{\delta}$ and standard error $\hat{\sigma}$, where $j = 1, \cdots, T$.

Finally, the automation of time series segmentation problem can be formalized with objective function:

$$
C = \arg \min_{s, M} \left( \sum_{i} E_{r_i} \right)
$$

subject to

$$
E_{r_i} \leq \bar{\delta} + m\hat{\sigma}
$$

This problem can be solved with bottom up algorithm.

4.5 Threshold of a “meaningful” time

After ordering clusters by their score and detecting their time phases, the next step is to set a reasonable threshold in the time mode to distinguish a meaningful and a meaningless cluster and determine the time steps when a cluster truly exist.

This is a tough problem for analysis related to the spectrum of a matrix/tensor generated from SVD if the eigenvalues or values in a eigenvector/mode cannot be clearly split into different groups. A good example is spectral clustering: if values in a eigenvector are sorted and a gap appears between top values and the rest, a cluster can be easily detected; if, however, there is no gap exist, clustering would be hard.

Analysis with tensor decomposition faces a similar problem. Further more, it is case specific, where the time mode is smooth w/o noise. However, if low values have meanings, it could lose information as we will show in experiment
Algorithm 1: Time Mode Segmentation Algorithm

**Data:** Time Series $TS$

**Result:** Segmentation $S$

$S = [[1], [2], \cdots, [T]]$

$\hat{TS} = \text{SlideWindowFilter}(TS)$

$\Delta = TS - \hat{TS}$

$\delta = \text{mean}(\Delta)$

$\hat{\sigma} = \text{standardError}(\Delta)$

$max\text{error} = \delta + \hat{\sigma}$

for $i = 1 : T - 1$

$merge\text{cost}(i) = \text{LinearRegressionError}(TS[i:i+1])$

while $\text{min}(merge\text{cost}) = max\text{error}$

$index = \text{indexOf}(\text{min}(merge\text{cost}))$

$S(index) = \text{merge}(S(index), S(index+1))$

$\text{delete}(S(index+1))$

$merge\text{cost}(index) = \text{LinearRegressionError}(TS[\text{merge}(S(index), S(index+1))])$

$merge\text{cost}(index-1) = \text{LinearRegressionError}(TS[\text{merge}(S(index-1), S(index))])$

for Lakehurst military action dataset. Other work would apply it to anomaly detection where data point has significant different value.

The real challenge from sparse dynamic graph is: (1) How to correctly detect a cluster’s existence in time mode $T_r(t)$? Experiments have shown that the value in $T_r$ is affected by the total weight of edge $\sum_{i,j} x_{ijt}$ between all node-pair $i, j$ in a cluster $C_k$ at time $t$. However, if $C_k$ has only one edge $e(i, j)$ with large weight making $T_r(t)$ highest at time $t$ while other nodes in the cluster has no edge, It is hard to decide the existence of $C_k$ at time $t$. (2) There is large noise in $T_r(t)$, undermining the idea that large value means meaningful moment.

We first address the existence of a cluster. Note that a cluster exists if all it member are correctly clustered, meaning the Silhouette score of any node $S_i(t) > 0$. A direct way to determine the existence of a cluster is to check $S_i(t) \geq 0$ for all nodes $i$.

A better way is to use a binary snapshot for dynamic graphs instead of weighted edges. This method filters large edge weight and make sure that the largest value in $T_r(t)$ represents the largest number of edges in $C_k$ exists at $t$. The trade off is that we lose detail information of the edges and will fail to determine the meaningful events (as will be explained in experiments for Lakehurst data).

For a fine scale of temporal granularity, snapshots are sparse and the edges are too few to to perform clustering for ground truth groups. The time mode $T_r$ has large noise. A solution to this is smoothing with techniques such as moving average filtering and then detecting a cluster’s stable phase.
5 Experiments

We use both synthetic data and real data to test our model. The Section 5.1 describes the data sets for our experiments. Section 5.2 illustrates how the rank of $R$ affect the clustering results in our framework. Section 5.3 shows the evaluation of our framework with baseline method of Evolutionary Clustering and works in [8]. Our program implements tensor toolbox from [1]. For some experiments, We modified the toolbox to satisfy some constraints such as non-negative value or keeping two modes equal.

5.1 Data Sets

5.1.1 Synthetic Data

The first data set is a toy problem. We created 2 disjoint static cliques $G_1$ and $G_2$, each with 50 nodes. In the first 50 time steps, all the edge weights are 1 and change to 0.1 in the next 50 steps. A $100 \times 100 \times 100$ tensor was created from the weighted adjacency matrices. Non-negative TD with the symmetry constraint $A_r = B_r$ was applied. We expect to see two distinct clusters corresponding to the two groups.

The second data set is created from a random direction model with reflection to simulate the movement of users, the setting is as follow:

- A $60 \times 60$ square is divided into four $30 \times 30$ small areas on upper left (UL), upper right (UR), lower left (LL) and lower right (LR).
- Each area contains 100 users simulated by the same random direction model with reflection on bounds.
- In the random direction model, each user is initialized randomly at a location within his area, with moving direction $d$ drawn from uniform distribution $U(-\pi, \pi)$, speed $v \sim U(0.1, 5)$, moving time $tm \sim U(2.0, 36)$ second. A user has his direction reflected when he touch the boundary of his area. After the moving time ends, the user pause for time $tp \sim U(0, 10)$ second. Then he continues the movement with direction $d$, speed $v$ and moving time $tm$ re-sampled from the previous distribution.
- The location of a user is recorded every second for 100,000 seconds (about 28hr).

To study how TD capture the change of clusters, we further divide nodes in each area into 4 subgroups and shift the subgroups to the four areas (totally 16 subgroups from $C_1$-$C_{16}$). Table 2 shows the time of migration and Table 1 shows the ground truth clusters we expect from our framework: the dissolution of $G_1$ - $G_4$, the formation of $G_5$ - $G_8$ and some intermediate clusters $G_9$ - $G_{12}$ after $G_4$’s migration and before $G_3$ dissolution.

At each time step $t$, an edge $e_t(i, j)$ with weight 1 is created if nodes $i$ and $j$ is within distance 1.

5.1.2 LakeHurst Data

The LakeHurst data from US Army Research Lab contains 3 hours’ (10800 seconds) trace of 70 vehicles (ground and airborne) in LakeHurst, NJ [5]. 64 of
Table 1: Ground Truth Clusters

| Group ID | Description          | subgroup IDs        | # nodes |
|----------|----------------------|---------------------|---------|
| G1       | begin in UL          | C1-C4               | 100     |
| G2       | begin in UR          | C5-C8               | 100     |
| G3       | begin in LL          | C9-C12              | 100     |
| G4       | begin in LR          | C13-C16             | 100     |
| G5       | ending in UL         | C1, C5, C9 and C13 | 140     |
| G6       | ending in UR         | C2, C6, C10 and C14| 40      |
| G7       | ending in LL         | C3, C7, C11 and C15| 100     |
| G8       | ending in LR         | C4, C8, C12 and C16| 120     |
| G9       | temporary in UL      | C1, C5 and C13      | 105     |
| G10      | temporary in UR area | G2, C6 and C14      | 30      |
| G11      | temporary in LL area | G3, C3, C7 and C15 | 175     |
| G12      | temporary in LR area | C4, C8 and C16      | 90      |

Table 2: Migration Time for Slow Group Migration with respect to Different Temporal Granularity: Group 1 begins migration at the 40,000th second (or 667th minute)

|   | G1 | G2 | G3 | G4 |
|---|----|----|----|----|
| Begin Time (s) | 40000 | 30000 | 70000 | 47000 |
| Begin Time (min) | 667 | 500 | 1167 | 783 |
| End Time (s) | 57230 | 48590 | 76910 | 65710 |
| End Time (min) | 953 | 810 | 1282 | 1095 |

them are split into 9 platoons, moving from one checkpoint to another. There are two paths to follow from the star location to the destination and each path has 5 checkpoints. A platoon choose one path and stop for a while at each checkpoint until it reaches the destination. Some platoons could meet with one another during the movement. Other 6 vehicle move separately but would meet one of those platoons.

The data we obtain for our work contains the distances between vehicles every second. Fig 2 is a density estimate of distances of all pair of vehicles during the 3 hours. Most distances are within 150 meters. Since vehicles in a platoon moves together, we assume 150 meter as a threshold distance for two vehicles within a platoon. A $70 \times 70 \times 10800$ tensors for similarities between vehicles in the 3 hours can be constructed in two ways: the first one uses the Gaussian similarity with $\sigma = 50$. The second one uses a binary similarity with a threshold distance $\alpha = 150$.

Table 3 shows the ground truth of 9 platoons provided to us, including the number of vehicles. Some platoons meet with one another and form a cluster during a period of time. There are 10 such temporally formed clusters. In total, we expect an method to detect 19 ground truth clusters (containing 233 vehicles) in the dynamic network.

Table 3: Info of Platoons

|   | PT1 | PT2 | PT3 | PT4 | PT5 | PT6 | PT7 | PT8 | PT9 |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| size | 5   | 10  | 5   | 6   | 10  | 6   | 6   | 10  | 6   |
5.1.3 UMass Mobility
The UMass mobility data set contains 8 days’ logs of around 4500 WiFi access points (APs) connected by more than 4000 mobil devices of approximately 3500 students and staff from University of Massachusetts Amherst [16]. The data we obtain records the time when a device connects (or loses connection) to an AP. Interesting knowledge could be the devices belong to the same person or group of people in the same department.

The dynamic graph can be constructed based on whether two devices connects to APs in the same building. That is, edge $e_t(i,j)$ in a dynamic graph represents that device $i$ and $j$ connects to APs in the same building a time $t$.

5.2 Effect of rank $R$ on clustering
The rank of decomposition, $R$, affects the approximation ratio to the original tensor, the uniqueness of the result [9] and the number of components. However, it is still unclear how it affect the meaning of components we can interpret for clusters. In the following experiments, we begin with simple toy problem to show how $R$ affects clusters in static graph, especially when TD combines two factor when $R$ is small. Then we continue to a more complex problem from a random direction model to further test our hypothesis.

5.2.1 Effect of Small Rank
Reports suggested in a real-world data that TD with small rank maintains strongest community structures of a dynamic network in components but ignore those weak structures, 2) combines multiple community structures into one component [8]. However, it is unclear under what circumstance TD will eliminate a weak structure instead of combining it with others into one component. Meanwhile, the report didn’t provide a measure for how strong a community structure is if a component contains multiple communities.

From cluster’s point of view, we define a Norm of a Cluster $c_k^{(r)}$ to measure the strength of community structure:

$$\text{norm}(C_k^{(r)}) = \sqrt{\sum_{\{i,j,m|i,j\in c_k^{(r)}, t_{im}>0\}} R_{ijm}^2}$$ (8)
where \( t_{mr} \) is the value at \( m \)th time step in time mode \( T_r \) in component \( r \). With this definition, \( \text{norm}(C^{(r)}_k) \) is affected by the similarity \( X_{ijt} \) as well as the size of \( C^{(r)}_k \). We discover this norm of a cluster is highly related to conditions when TD eliminates weak clusters or combines them into a component.

We applied non-negative PARAFAC decomposition on the first toy problem with \( R = 1 \) and constraint \( A_r = B_r \). To study the condition when low-rank TD eliminates a cluster or combines clusters into one component, we specifically normalize elements \( a_{i1} \in A_1 \) to the range \([0, 1]\) because we can interpret how likely node \( i \) belongs to the component. (These normalized values are only for illustration purpose and they should not be used in Equation 4, 2) Fig 3(a) shows the plot of column vector \( A_1 \). Nodes in the first clique \( C^{(1)}_1 \) have value \( g_1 = 1 \) and nodes in the second clique \( C^{(1)}_2 \) have value \( g_2 = 0.98 \). Since the difference \( d = g_1 - g_2 = 0.02 \) is not large, this suggest that TD has combined the two cliques into \( A_1 \).

Figure 3: Effect of small \( R \) on Mode A: (a) two clusters combined into \( A_1 \); (b) \( d \) decreases exponentially with \( W \); (c) change of edge weight or cluster size has same effect on \( d \)

The value \( d = g_1 - g_2 \in [-1, 1] \) shows the difference between the two cliques in the component: if \( d \approx 0 \), the two clusters are equally important in the component; if \( d \approx 1 \) or \( d \approx -1 \), we can interpret that one of the clique is eliminated. We discover that the ratio of two cliques' norms \( W = \text{norm}(C^{(1)}_2)/\text{norm}(C^{(1)}_1) \) is highly related to \( d \). As we pointed out in Equation 8 that \( \text{norm}(C^{(r)}_k) \) is affected by \( X_{ijt} \) and the cluster size, we first investigate how TD behave if cliques \( C^{(1)}_1 \) and \( C^{(1)}_2 \) are different only in \( X_{ijt} \). We add a scalar \( a = 1, 1.01, \cdots, 1.15 \) to \( X_{ijt} \), where \( i, j \in C^{(1)}_2 \) to generate new tensors and thus make \( W \) change accordingly. Note that the result of a TD is not unique \([15]\) in the toy problem,
Figure 4: Plot of Mode A: (a) PARAFAC with small $R$ combines clusters with common nodes into one component; (b) PARAFAC with proper $R$ puts each cluster in a separate component.

for each $W$ value, we ran TD for 1000 times. Fig 3(b) shows the boxplot of $d$ given different $W$ value. The left most boxplot for $d$ given $W = 1$ shows that $d$ falls within $[-0.5, 5]$ most of the time, suggesting that TD combines the two cliques given that they have the same cluster norm. However, as the cluster norm differs a little, the average $\bar{d}_w$ decreases exponentially. Since $\bar{d}_w > -1$, we set the model to be $\bar{d}_w = a \exp(bW) + 1$ and perform a regression, obtaining $a = 1.69e + 35$, $b = -81.11$. When $W > 1.06$, $\bar{d}_w$ is almost $-1$, meaning that $C^{(1)}_2$ dominates the component but $C^{(1)}_1$ is eliminated.

We also investigate how the difference of cluster size affects the TD behavior, and compare it to that effect caused by weight. We increase the size of clique $C^{(1)}_2$ to 50, 51, $\cdots$, 100, fixed the weight the same as $C^{(1)}_1$ at each time step and obtain 50 tensors, each has different norm ratio $W$. For each $W$, we run TD for 1000 times and define the average $d$ as $\bar{d}_s$. Figure 3(c) compare the log-plot of $\bar{d}_s$, $\bar{d}_w$ with $W \in [1, 1.08]$. There is little difference between the two line, suggesting that as long as the norm ratio $W$ is the same, TD will has similar behavior to generate a component regardless the difference in $X_{ijt}$ or cluster size.

Previous experiments shows the condition when TD eliminates weak community structures. We next investigate the condition when multiple clusters are combined into one component. We change the toy problem as follows: For the first 50 time steps, nodes 1-10, 11-30, 31-60 and 61-100 from 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). $X_{ijt} = 1$ if node $i,j$ belongs to the same clique at time step $t$, otherwise $X_{ijt} = 0$.

We apply PARAFAC decomposition with $R = 4, 8$; results are shown in Figure 4. When $R=4$ (Fig 4(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 4(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, effect of $R$ on clustering on TD components would be:

- If $R$ is smaller than the number of clusters in the dynamic graph, indepen-
dent clusters with similar norm value, or clusters with common nodes are
tend to combine in one component. Cluster detection can be difficult in
this case. Meanwhile, clusters with small norm are likely to be eliminated
because it is more beneficial for TD algorithm to maintain clusters with
large norm to get minimum error.

- If \( R \) is large, TD component tend to contain less cluster, especially for
  NTF. Ground truth clusters will be more easy to detect.

5.3 Evaluation with Data

We will evaluate our framework by comparing the retrieved clustering result to
that from Evolutionary Clustering. We also compare our clustering method to
the binary classification from [8].

For our framework, a Non-negative PARAFAC decomposition is applied with
constraint that \( A_r = B_r \). Rank \( R \) and time granularity are set differently based
on the data sets. The PARAFAC components are sorted by descending order of
its norm. \( A_r, B_r \) and \( T_r \) are normalized such that \( \| A_r \otimes B_r \otimes T_r \|_F = 1 \) and
\( \lambda_r \) is scaled to the norm of that component. K-means with \( k = 2, \ldots, 12 \) is
applied on vector \( A_r \). From the 11 clustering results, the one with the highest
Silhouettes score is chosen as the final clustering \( C_t \).

For evolutionary clustering framework (EC), let \( X_t = X_{(:,t)} \) be a similarity
matrix for graph \( G_t \). EC framework perform clustering on a mixture of similarity
matrix \( R_t = (1 - \beta)X_t' + \beta X_{t-1} \) at time \( t \) to obtain a clustering result \( C_t \), where
\( \beta = 0.1 \). Since the mixture \( R_t \) is different from current similarity matrix \( X_t \), \( C_t \)
is not necessarily the best clustering for \( X_t \), it has a snapshot quality function
\( sq(C_t, X_t) \) to evaluate the clustering result. It also has a historical cost function
\( hc(C_t, X_{t-1}) \) to calculate the cost when the clustering result change from previous
step. The objective is to minimize

\[
\sum_{t=1}^{T} sq(C_t, X_t) - cp \sum_{t=2}^{T} hc(C_t, X_{t-1})
\]

We use the same setting as [4] for the framework with \( cp = 0.1 \). For each
mixture \( R_t \), we also apply K-means to obtain the clustering \( C_t \).

5.3.1 Metric for Evaluation of clustering

Suppose the method retrieves a set of clusters \( \{C_k\}_{k=1}^{K} \), the ground truth clusters
are \( \{G_n\}_{n=1}^{N} \). The recall and false positive rate (FPR) of nodes in ground truth
clusters are good metrics to evaluate the performance of our methods.

We denote \( C_{jn} \) as a cluster mapped to a ground truth cluster \( G_n \) in the same
time period. The mapping to \( G_n \) is done by selecting \( C_k \) with arg max_k \( \frac{|G_n \cap C_k|}{|G_n|} \).
Note that we map an empty cluster \( \Phi \) to \( G_n \) if there is no \( C_k \) detected in the
same time period as \( G_n \). We denote these missing ground truth clusters as
\( \{G_n^{(M)}\}_{n=1}^{N} \). Also, there could be multiple clusters, say, \( C_{jn} \) and \( C_{j'n} \) from the
same component representing \( G_n \), mapped to \( G_n \) if they are subclusters of \( G_n \),
where \( C_{jn} \cap C_{j'n} = \Phi \) and \( C_{jn} \cup C_{j'n} \subset G_n \). Let \( \{C_{jn}\}_{j=1}^{M} \) be a set of clusters
mapped to the ground truth clusters. The recall (true positive rate) of nodes in
ground truth is then \( \frac{\sum_{j} |C_{jn} \cap G_n|}{\sum_{n} |G_n|} \), the false positive rate is \( \frac{\sum_{j} |C_{jn} \cap G_n|}{\sum_{j} |C_{jn}| - \sum_{n} |G_n|} \).
Since both our method and the method in [8] can return a ranking list of clusters \( \{C_k\}_{k=1}^K \), we want to evaluate how well a cluster \( C_k \) is mapped to a ground truth cluster \( G_n \). We calculate the recalls (TPR\(_k\)) and false positive rate (FPR\(_k\)) of nodes at each ranking \( k \) in the list to generate a curve in ROC space [7]: let \( m_{ik} \) be the \( i \)th node in \( C_k \), \( l(m_{ik}) = 1 \) (labeled as ground truth node) if \( C_k \) is mapped to a ground truth Cluster \( G_n \) and \( m_{ik} \in G_n \); otherwise \( l(m_{ik}) = 0 \) (labeled as non-ground truth node).

\[
\text{TPR}_k = \frac{\sum_{j=1}^k \sum_{i=1}^{\left|C_j\right|} l(m_{ij})}{N^+}
\]

\[
\text{FPR}_k = \frac{\sum_{j=1}^k (\left|C_j\right| - \sum_{i=1}^{\left|C_j\right|} l(m_{ij}))}{N^-}
\]

where \( N^+ = \sum_{n=1}^N |G_n| \) is the total number of all ground truth nodes and \( N^- = \sum_{j=1}^K \sum_{i=1}^{\left|C_j\right|} 1(l(m_{ij} = 0)) \) are the non-ground truth nodes retrieved by the algorithm. If there are ground truth clusters \( \{G^{(M)}_l\}_{l=1}^L \) missing from the list, the maximum TPR will be less than 1.0. We will add the missing clusters to that list so that the curve goes to \((1.0, 1.0)\) in ROC space and illustrates the overall performance of the methods. Those missing clusters are assigned with a negative score \(-1\) to ensure that they are at the bottom of the list.

### 5.3.2 Data from Random Direction Model

For the dynamic graph generated from nodes performing a random direction model movement, we merge snapshots every minute to avoid 0 matrix snapshots. Further more, other temporal granularity of 5 minutes, half an hour and 1 hour are also applied to merge snapshots to generate a tensor. Table 4 shows the top 10 scored clusters from all components that are generated by TD with constraint \( A_r = B_r, R = 10 \) and granularity of 1 minute on a dynamic graph with binary snapshots. Each row of the table describes a cluster: the score column represents the ranking-score \( SO_j^r \) in Equation 4. Component column shows the component ID where the cluster is generated; description column shows the groups of subgroups that belongs to the cluster; our time series analysis detects the time intervals when the cluster exists, is forming or disappearing and show them in “stable ph”, “forming” and “disappear” column (some more detail about time series will be explain in next subsection). Finally, “\# node” shows the number of nodes in the cluster.

Note that the \( R \) value is small compared to the number of possible clusters in the data: there are 4 groups at the beginning of the process and another 4 at the end, meanwhile, there are more than 3 intermediate clusters during the migration of nodes. As a result, the top 10 clusters are not all the ground truth groups we set at the beginning/end of the process. Meanwhile, S2, containing only 40 nodes at the end of the process, is missing in the whole cluster list. The top 10 clusters shows what we expect from the temporal clustering framework. First, some groups/subgroups are combined in one component. For example, G1 (row 1) and G4 (row 10) are combined in Component 3; G2 (row 3, 5) and G1 (row 9) are also combined into Component 1. Second, by Silhouette score, K-means split some group into smaller subgroups (e.g., G2 is split to subgroup C36 in row 3 and the rest subgroups in row 5). We also tried other clustering methods.
Table 4: Top 10 Clusters from TD with R=10, Granularity =1 min and Constraint \( A_r \times B_r \)

| Order | Score | Component | tinyClusterID in Component | Description | Stable Ph | Forming | Disappear |
|-------|-------|-----------|----------------------------|-------------|-----------|---------|-----------|
| 1     | 118.15| Comp3     | 1 | G4 | [1,673] | N/A | [673,1084] |
| 2     | 95.32 | Comp5     | 1 | S1 | [1273,1667] | [937,1085] [1172, 1278] | N/A |
| 3     | 94.45 | Comp1     | 1 | C6 | [1,507] | N/A | [507,971] |
| 4     | 68.12 | Comp2     | 1 | S4 | [1282,1667] | [1167,1282] | N/A |
| 5     | 59.64 | Comp1     | 2 | C5, C7, C8 | [1,507] | N/A | [507,971] |
| 6     | 49.24 | Comp4     | 1 | S3 | [1,673] | N/A | [672,1084] |
| 7     | 47.28 | Comp6     | 1 | C3, C7, G3 | [804,920] | [611,803] | [921,1269] |
| 8     | 47.21 | Comp10    | 1 | C1, C5 | [732,813] | [583,731] | [814,1086] |
| 9     | 45.96 | Comp4     | 3 | G4 | [1,507] | N/A | [507,971] |
| 10    | 43.75 | Comp3     | 2 | G1 | [1,673] | N/A | [673,1084] |

that automatically decide the number of clusters, such as Newman’s modularity method on \( A_r \times B_r \) (code could be downloaded in http://strategic.mit.edu/). However, the result show that it split even more subclusters than we desire.

Third, our framework also detects some intermediate clusters, such as the one in row 7. This cluster exists at the end of G2’s migration, before G3’s disappearing. It lasted about 100 minutes, making TD to consider it as an important component with large Frobenius norm. Fourth, the time phases detected for those combined clusters (in Component 1, 3) fail to match the time we set for the groups to migrate (Table 2). An explanation is that the time mode is for both clusters. In that case, the time phase is incorrect for any cluster. Fifth, the time intervals of other components are close to, but do not perfectly match the time we set for the groups to migrate (Table 2). There are several reasons: (1) the location of a node slowly shifted from one area to another, there will be delay where the connection between nodes in groups becomes different. (2) the snapshots are very sparse, resulting large noise in the time series and unavoidably affecting the accuracy of the phase detection algorithm (see later subsection).

Table 5 illustrate the ground truth clusters and nodes retrieved by PARAFAC of rank \( R = 10, 20, \) and 40. There are totally 12 ground truth clusters and 1200 nodes. As our previous investigation points out, PARAFAC with smaller rank tends to eliminate small independent clusters and combines clusters with common nodes into one component. When \( R = 10 \), only 8 ground truth clusters with large norm (Eq 8) are retrieved (namely, G1-G5, G7, G8 and G11) in separate components. Interestingly, the first 2 components, with largest norm, contains clusters that are a mixture of the retrieved ground truth clusters. We cannot map them to any ground truth clusters that exist in the same period of time. When \( R = 20 \), PARAFAC retrieves 11 ground truth clusters in 11 components. Other 9 components contains clusters with small number of nodes. These small clusters can be viewed as noises because the trajectories of those nodes in clusters show that they happen to move close to one another and stop moving due to the setting of random direction model. G6, consisting of only 30 nodes in a short period of time in the upper right area, is missing because the norm of G6 is smaller than those noise clusters. When \( R = 40 \), all ground truth clusters are retrieved in separate components and all nodes are correctly
Table 5: Recall of Nodes and Clusters of PARAFAC: PARAFAC of larger rank retrieves more ground truth clusters

| Rank | 10  | 20  | 40  |
|------|-----|-----|-----|
| # Cluster Retrieved | 8   | 11  | 12  |
| Recall of Cluster     | 0.6667 | 0.9167 | 1.0 |
| # Nodes Retrieved     | 935  | 1170 | 1200 |
| Recall of Nodes       | 0.7792 | 0.975 | 1.0 |

clustered.

5.3.3 Comparison

To compare our method to Binary Classification (BC) in [8], use ROC curve introduce in previous subsection. Since components in [8] are ordered by core consistence [3], a list of clusters with ranking score can also be generated: core consistence score of a component is assigned to the cluster classified as belonging to the component, while the other cluster is assigned with 0 score. Fig 5 illustrate the ROC curves of the two methods under PARAFAC of rank \( R = 10 \) and 20. We do not show the result for PARAFAC with \( R = 40 \) because both methods successfully retrieves all ground truth clusters.

Fig 5(a) shows an ROC space that illustrating the performances of two methods under PARAFAC with \( R = 10 \): The order list from BC contains 8 ground truth clusters, 12 meaningless ones and another 4 missing ground truth clusters. The top 2 clusters, coming from the first 2 components, are mixtures of clusters. They do not match any ground truth clusters, resulting in TPR of 0 and FPR of 0.1315 (black circle). The next 8 clusters are ground truth ones from 8 separate components. This make TPR sharply increase to 0.7792 (black triangle). Note that FPR also increases a little from 0.1315 to 0.1524 because BC mistakenly includes some non-ground-truth nodes to these clusters. The next 10 clusters in the list have score 0, resulting in the TPR of 0.7792 and FPR of 1.0. Finally, the rest 4 clusters are missing ground truth clusters. The ROC curve goes from (1, 0.7792) to (1, 1). On the other hand, SO ranking also has 20 retrieved clusters with their SO scores and 4 missing ground truth clusters with score -1. All the 8 ground truth clusters are on the top, making the ROC curve go straight from (0,0) to (0,7792). The next 12 clusters have positive SO scores and have higher ranks than the missing clusters. As a result, the ROC curve goes to the point (1, 0.7794) and continues to (1.0, 1.0). From point (0,0) to point (0.1524, 0.7792), the ROC curve of SO ranking stays on the upper left of that of BC, indicating that SO ranking is better than BC in avoiding false positive on the top ranking clusters. The sharp increase of TPR of both ROC curves at the beginning shows that the FPR is low if the retrieved cluster can be mapped to a ground truth cluster. This suggests that if PARAFAC decomposition only put one ground truth cluster in a component, that cluster can be accurately obtained.

Fig 5(b) shows the performances of BC and SO ranking under PARAFAC with \( R = 20 \). With PARAFAC decomposition capturing 11 out of 12 ground truth clusters, both methods improve on their maximum TPR to 0.9774. SO ranking give high score to ground truth clusters and has better performance
Figure 5: Curves of Binary Classification (BC) and Similarity Overtime Ranking (SO) in ROC space: (a) SO ranking has lower false positive rate (FPR) and higher true positive rate (TPR) for top clusters; (b) Both methods have higher maximum TPR with $R$ of PARAFAC increases.

than BC for the top 11 clusters (from (0,0) to (0.047,0.9774) marked with black circle). With larger rank $R$, no ground truth clusters are combined into one component and thus BC improves its FPR on the top clusters. However, it still adds non-ground-truth nodes to its top ranking clusters, resulting in larger FPR than SO ranking. The missing cluster is G6, a cluster with only 30 nodes in a short period of time, has a very small cluster norm (Eq 8). Although the rank is 20, 9 components contains unknown clusters that have larger cluster norm than G6 during the whole migration process. As a result, PARAFAC prefers those unknown clusters and ignores G6 to get a better approximation ratio of the tensor.

We also compare the performance of our method to Evolutionary Clustering. For EC to capture the ground truth clusters, we merge every 4500, 6000, 8000 and 12000 snapshots and generate 4 tensors of binary similarity: D1, D2, D3 and D4. Table 6 shows the average recall of clusters and nodes from 30 repeated experiments of EC. Smaller time granularities lead to sparse dynamic graphs. Applying EC on D1 and D2 results in retrieving more ground truth clusters (G1-G8, G11) but lower recall of nodes in a cluster. For larger granularities, EC retrieves ground truth cluster G1-G8 from D3 and D4, with high recall of nodes. But G9-G12 are lost because the merge of snapshots. By comparing Table 5 and 6, temporal clustering under PARAFAC decomposition shows better recall performance than EC.

| data set | D1 | D2  | D3  | D4  |
|----------|----|-----|-----|-----|
| # Clusters Retrieved | 9  | 8.87| 8   | 8   |
| Cluster Recall | 0.75 | 0.74 | 0.67 | 0.67 |
| # nodes Retrieved | 638.08 | 693.38 | 682.38 | 788.31 |
| node Recall | 0.5317 | 0.5778 | 0.5686 | 0.6569 |

5.4 Lakehurst Data

We applied PARAFAC decomposition with $R = 10, 15, 20$ and $30$ on tensors of both Gaussian and binary similarities. The recall of ground truth clusters and
nodes are illustrated in Table 7. As $R$ increases, the recall of both clusters and nodes improves. The number of retrieved ground truth clusters is smaller than $R$, indicating that some components might contain a combination of multiple clusters or some unknown community structures. We can retrieve all 19 ground truth clusters from 19 components of PARAFAC decomposition with $R = 30$. The other 11 components contain some clusters that are formed during a process when multiple platoons are merging into a big cluster. Note that when $R = 20$, we retrieve more clusters but less node in tensor of binary similarity. This is because those retrieved clusters are mostly single platoons. On the other hand, ground truth clusters retrieved from tensor of Gaussian similarity are formed by multiple platoons.

We compare the recall performance to EC on both tensors of Gaussian and binary similarities (in last two columns EC (G) and EC(B) of Table 7). EC can be applied to both tensor without merging snapshots. however, it can only retrieve the 9 ground truth platoons.

Table 7: Recall of Clusters and Nodes

| Rank | Gaussian Similarity | Binary Similarity | EC(G) | EC(B) |
|------|---------------------|-------------------|-------|-------|
|      | 10                   | 15                | 20    | 30    |
| # Cluster | 7                   | 8                 | 11    | 19    |
| Cluster Recall | 0.3684              | 0.4211            | 0.5789| 1.0   |
|      | 10                   | 15                | 20    | 30    |
| # Node | 100                  | 126               | 196   | 233   |
| Node Recall | 0.4292              | 0.5408            | 0.8412| 1.0   |

Figure 6 shows curves to evaluate performances of Binary Classification (BC) and SO ranking in ROC space. Fig 6(a) and (c) show the performances under PARAFAC with $R = 15$ on tensors with binary and Gaussian similarities. In both cases, the top ranking cluster by BC is a mixture of clusters that does not match any ground truth. Thus, BC has low TPR and higher FPR in retrieving the top clusters. Both methods have TPR at about 0.55 when FPR reaches 1.0 because of the missing ground truth clusters. As the rank of PARAFAC increases to 30, each ground truth cluster is represented by one component. The top clusters from BC’s list matches ground truth cluster, resulting in an improvement of TPR for top ranking clusters (Figure 6(b) and (d)). Since all ground truth clusters are captured, the recall of nodes improves. SO ranking performs better than BC because it has a higher TPR and lower FPR most of the time.

Time Series Figure 7 shows an experiment of PARAFAC decomposition $R = 20$ on a tensor with binary snapshot matrices. Fig 7(a) shows the plot of time series for a ground truth cluster consisting of platoon7 and platoon8, with time phases of sustenance state marked by our phase detection algorithm. Our algorithm detects 5 periods when the time series reaches a stable peak with the same normalized value “1” every 1800 seconds (0.5hr). This is what we expects for a tensor of binary similarity: the time mode value in $T_r$ represents how many members have join the cluster at $t$. When it reaches 1, a max number of members join the cluster and the time of sustenance state can be obtained. Note that there are values around 0.3 before each peak. This means one platoon stop at a check point and the other platoon is joining. For those 0.3 values after each peak, it can be interpreted as that a platoon is moving to the next check
Figure 6: ROC Curves for Retrieving Vehicles in Ground Truth Clusters: ROC curves improve with increase of $R$. SO ranking has lower FPR than Binary Classification.

Figure 7: Detection of Time Phases in Time Series for Union of PT7 & PT8 in tensors of both Binary Similarity and Gaussian Similarity. Fig 7(b) is generated with Gaussian similarity. Note that the height of peaks are no more equal, this is because the average similarity of the cluster is different at each peak time. The rise of each peak is less steep, giving more detail information how the average similarity
increases. However, the trade-off is that we cannot track the change of number of members in the union as in tensor of binary similarity because.

5.4.1 UMass Mobility Data
Many devices in the data are inactive most of the time, we select the top 500 active devices to construct a tensor. We group APs by their buildings and we have 163 AP groups for the 500 devices. \( X_{i,j,t} = 1 \) if device \( i,j \) connects to the same AP group, otherwise, it is 0. The time granularity is 1 minute to avoid 0 matrix snapshot. The rank \( R \) for decomposition is set to 40.

Figure 8(a) shows heat map for mode A with device IDs re-ordered according to their clusters. Each component can be easily split into 2 clusters from k-means algorithm: one cluster has large values and the other has small values close to 0. There are totally 422 devices assigned to the 40 clusters. Some of them share some common devices: cl

6 Conclusion and Discussion
In this paper, we propose a temporal clustering framework with tensor decomposition to detect clusters and obtain their temporal information in a sparse dynamic network, where snapshot-clustering based method such as evolutionary clustering fails to work. From clustering point of view, we study the limitation and effect of TD’s parameters and constraints on clustering results, design a automating segmentation algorithm for time series and compare the trade off between similarity functions when detecting time phases of clusters. Finally, we apply our method on both synthetic and real-world data and obtain clusters and their temporal information.

References
[1] Brett W. Bader and Tamara G. Kolda. Matlab tensor toolbox version 2.6. Available online, February 2015.
[2] Rasmus Bro. Parafac. tutorial and applications. Chemometrics and intelligent laboratory systems, 38(2):149–171, 1997.
[3] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of chemometrics*, 17(5):274–286, 2003.

[4] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560. ACM.

[5] Yung-Chih Chen, Elisha Rosensweig, Jim Kurose, and Don Towsley. Group detection in mobility traces. In *Proceedings of the 6th international wireless communications and mobile computing conference*, pages 875–879. ACM, 2010.

[6] Renato Cordeiro de Amorim and Christian Hennig. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324:126–145, 2015.

[7] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38, 2004.

[8] Laetitia Gauvin, Andr Panisson, and Ciro Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS one*, 9(1):e86028, 2014.

[9] Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an” explanatory” multi-modal factor analysis. 1970.

[10] Richard A Harshman. Determination and proof of minimum uniqueness conditions for parafac. *UCLA Working Papers in phonetics*, 22(111-117):3, 1972.

[11] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.

[12] Min-Soo Kim and Jiawei Han. A particle-and-density based evolutionary clustering method for dynamic networks. *Proceedings of the VLDB Endowment*, 2(1):622–633, 2009.

[13] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[14] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[15] Alwin Stegeman and Nicholas D Sidiropoulos. On kruskals uniqueness condition for the candecomp/parafac decomposition. *Linear Algebra and its applications*, 420(2):540–552, 2007.

[16] Jennie Steshenko, Vasanta G Chaganti, and James Kurose. Mobility in a large-scale wifi network: from syslog events to mobile user sessions. In *Proceedings of the 17th ACM IWCMC*, pages 331–334. ACM, 2014.