Hyper-network based Change Point Detection in Dynamic Networks

Tingting Zhu
School of Computer Science,
Southwest Petroleum University
Chengdu, China
tingtingzhu93@gmail.com

Ping Li
School of Computer Science,
Southwest Petroleum University
Chengdu, China
dping.li@gmail.com

Kaiqi Chen
School of Computer Science,
Southwest Petroleum University
ckqcins@gmail.com

Yan Chen
School of Computer Science,
Southwest Petroleum University
carly.chen@gmail.com

Lanlan Yu
School of Computer Science,
Southwest Petroleum University
yu.lanlan.yoki@gmail.com

ABSTRACT

Detecting event related change points on dynamic networks becomes an increasingly important task, as a change in network's structure may associate with a change in function of the networked system. However, general change point detection methods either fail to extract effective features or do not scale well. In this work, we introduce the distribution of nodes' importance to characterize static networks, the profile of a network that allows for networks' comparison and clustering on snapshots of dynamic networks. Based on this, we develop an approach to detect change points on dynamical networks by segmenting the snapshots into disjoint clusters, which can guarantee the scalability on large dynamical networks. Specifically, we construct a hyper-network whose nodes represent the snapshots. Then we do community detection on the hyper-network and serialize the community detection results in chronological order. The resultant sequence naturally indicates the potential changes. Experiments on both synthetic and real-world networks show the outperformance of our framework compared to the state of the art methods.

KEYWORDS

Dynamic networks, change point detection, hyper-network, node importance

ACM Reference Format:
Tingting Zhu, Ping Li, Kaiqi Chen, Yan Chen, and Lanlan Yu. 2018. Hyper-network based Change Point Detection in Dynamic Networks. In *MiLeTS '18, August 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With powerful ability to express complex relationships, networks are always used to model complex systems. Generally, elements of the system are represented by network nodes while interactions between the elements are represented by network ties. Examples include Senate co-sponsorship network connecting congresspersons [15], international trade network connecting countries or economies [16], and MIT proximity network connecting subjects who are physically in close proximity to each other [13], and so on. Those networks are all temporally evolving and generally called dynamic networks. Such dynamic networks are often represented by the snapshot model, where each network provides a snapshot of the interactions over a logical time-interval. Over dynamic networks, one of the most challenging task is change point detection, aiming to find time points where significant changes occur, or which according to Type 4 "Event and Change Detection" in [14] are defined as those snapshots that are significantly different from predecessors.

In recent years, with a growing interest focused on such a task, many approaches have been proposed. Studies can be categorized into two groups, i.e., unsupervised methods and supervised methods. The former is called anomaly detection in [9] while the latter mainly refers to classification methods. Among existing unsupervised approaches, there are roughly two lines (trends), i.e., generative models and similarity-based approaches. Generative models assume that there is some underlying model governing the generative process of networks. Those models such as EdgeMonitoring [15] and GHRG [13] typically construct a model of what is considered "normal" or expected, then flag deviations from this model as the anomalous. Similarity-based approaches are directly based on the observable networks. This kind of method can be formulated as in [14]: Given a network sequence $G(t)$ with a fixed length T and a scoring function $f : G_t \rightarrow \mathbb{R}$, a change is defined as a time point t, if $|f(G_t) - f(G_{t-1})| > c_0$ and $|f(G_t) - f(G_{t+1})| \leq c_0$. An essential problem for this kind of approach is how to measure the similarity of networks, which is the premise of clustering network sequence to detect change points [2, 17].

Although those methods have been successfully applied in some real-world networks, there are emerging issues to be addressed when we face the overwhelming scale (in terms of both network size and the number of snapshots) and the complex evolving process of the dynamic networks. First, real-world networks are very large in size and also evolve with time. As their size grows, the complexity of hypothesis testing based approaches such as GHRG grows as well. It has been shown [15] that GHRG takes 60h to detect

MiLeTS '18, August 2018, London, United Kingdom

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

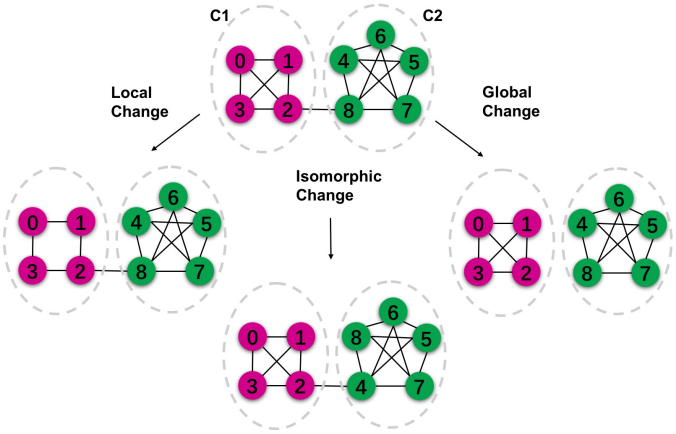

Figure 1: Toy example to show global change, local change and isomorphic change on a two-community network.

changes in the weekly Enron networks whose network size is 150. Second, selecting the most suitable summary features to help with supervised or unsupervised learning scheme is not a trivial task. In particular, for the network's structure, how to extract the features is still a considerable question. On the other hand, changes can be located at different level of networks. In fact, except for collective anticipation of or response to external events or system "shocks", there are still some events associated with minor or local changes of networks [16]. When we summarize the network and model the network from a global perspective, we may fail to detect some local changes. Moreover, there is a special type of change when two consecutive network snapshots are isomorphic, which may be related to the reorganization of the networked system (as shown in Fig. 1) but ignored by most of the existing methods.

In this paper, we propose a change-point detection framework based on clustering hyper-network to detect both global, local and even isomorphic changes. On assumption that network snapshots in between two successive changes are temporally coherent, we develop a two-step segmentation scheme to detect the change points in dynamically evolving networks. Namely, we summarize the network snapshot sequence that represents the dynamical network based on their features. Then, we serialize the summarization result and finally gain a segmentation of the network sequence. Consequently, the beginning of each segment can be flagged as a potential change.

Our main contributions are two-fold:

1. Summarize networks by leveraging nodes' importance: Network snapshots are summarized according to the relative importance of the nodes at the corresponding time point, and the dissimilarity of the snapshots is measured through comparing their summarizations using Jensen-Shannon divergence.

2. Present a new clustering scheme on change-point detection: instead of employing hierarchical clustering methodology, we construct a hyper-network by connecting the snapshots and detect the communities on the hyper-network, then we utilize the temporal information in serializing the clustering results to segment the network sequence. Accordingly, the cut points are identified to be change points.

The remainder of the paper is organized as follows. In Section 2, we give a brief review of related works, some of which will be compared with our framework. Then we formulate the change-point detection task and describe the proposed framework in detail in Section 3. After that, we experimentally evaluate the proposed scheme on both synthetic and real-world networks in Section 4. Finally, we give concluding remarks and discussion of future works in Section 5.

2 RELATED WORK

2.1 Similarity-based methods

As we aforementioned in Section 1, what features we extract from each network and how to compare the extracted features are both of significance. Many related works contribute a lot to feature extraction. Akoglu et al. [1] extract "behavior" features via decomposition eigenvector of the constructed correlation matrix of node pairs over specific time window (EigenBehavior).

On the basis of sociological theories, Koutra et al. [8] utilize fast belief propagation to derive pairwise vertex affinity scores as features of networks. Here, the feature is about how much each vertex influences another vertex. Essentially, this feature represents the connectivity of networks, which contains much more information than the adjacency matrix for it captures 1-step, 2-step, 3-step etc. neighborhoods in a weighted way.

On assumption that real-world networks follow global evolutionary trends, and the change points correspond to some moderate to high changes of this trends, features named Jaccard coefficient either based on node sets or edge sets for each network are constructed by [2].

From the literature, we can see that different features are extracted or constructed from a variety of perspectives. Besides, as to detect change points, there are also some other choices, e.g., quality control with individual moving range used in [8]. Berlingerio et.al [2] devise a methodology hierarchically clustering the networks based on the Jaccard coefficient features. Networks follow a constant trend will be in the same cluster and the beginning of each cluster indicates a change in counter-trend with the previous cluster.

Different from clustering methods, a method named DynSnap proposed in [4] automatically slices the time evolution process of a complex system to intervals in the event landscape. Based on an assumption that the adjacent intervals have different but not too dissimilar sets of events, they find each slice by maximizing the similarity between the sets of event on consecutive time intervals. The approach is not directly designed for change-point detection. However, it does find change points which coincide with boundaries of its' intervals. Moreover, by modeling the events with the emergence or absence of the edges of time-evolving networks, DynSnap can be applied to change point detection in evolving networks. In this work, we use it as a baseline method.

2.2 Generative models

The basic idea of generative models is to generate a probabilistic model from multiple snapshots and then detect change points by

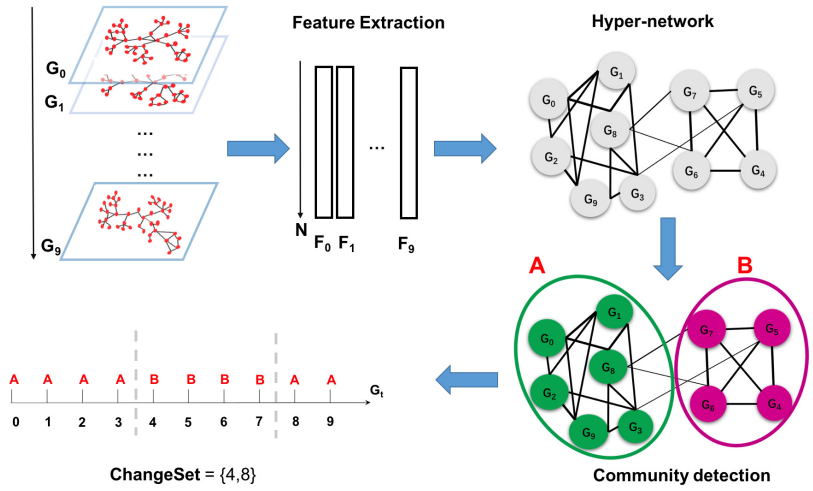

Figure 2: Overview of the Framework of Change-Point Detection

testing a new snapshot with the estimated model. The difference between those generative models lies in the model construction and the method for determining outliers. Focusing on capturing hierarchical community structures, Peel et.al [13] put forward a method named GHRG, in which they assume that the observable networks are generated from a particular model and there is a change only when the generative model shifts from a structural "normal" at some point in time. The advantage of this method is that p-value accompanied with the results quantifies the confidence of the conclusion and the change of the network structure can be visualized to help one get an intuitive understanding. However, in order to find a proper threshold, it bootstraps networks from the generative model, causing it to be a time-consuming and highly complex method.

Taking the temporal dependency into consideration, Wang et.al [15] hypothesize that the observed network snapshots are samples depending on the latent model and the previous snapshot. Different from GHRG model, they make no assumption on the concrete form of the generative model. They stress the change of the edges, proposing a conditionally independent two-state Markov chain to trace the presence or absence of the edges. Under such model, they directly extract joint edge probability as "feature" vector to represent each snapshot, and then, they compute the dissimilarity score of each consecutive snapshots, and flag out a change point when the dissimilarity score is above a threshold estimated by permutation test.

Wang et.al [16] study change-point detection problem on networks with community structures. Since many existing algorithms always fail to detect minor and local changes, Wang et. al develop a hierarchical framework which models both the intra-community evolution and the inter-community evolution so as to capture and even distinguish the local and global changes. As the generic design of their framework, many state-of-the-art change point detection algorithms can fit into the framework.

3 PROPOSED FRAMEWORK

In this section, we put forward our framework to detect change points.

3.1 Problem definition

Real world networks are always evolving with time either in their structure or attributes. With time goes by, there may be some changes such as insertion or deletion of nodes, even modification of some attributes [14]. Generally, each observed network can be viewed as one system state under some evolution patterns. when some events or "shocks" happen, the underlying pattern will change, causing change in the networks. Here, we refer to such change as "EventChange". A special "EventChange" may occur when the snapshot is structurally isomorphic to its predecessors. We also include the periodic change into "EventChange" though there's no real event behind it. Besides, there is another type of change called "GeneralChange" which is smoother comparing to the sharp "EventChange". It happens even when the pattern holds, because of the change nature of the real world.

Since there are always fluctuations in the original network sequence, we may wrongly treat some "GeneralChange" as "EventChange" when we directly do the change point detection on the original scale of networks with an improper threshold. But when our view changes to pattern level, we can find smoother subsequences and naturally flag each shift of patterns as change point. That is, if we know what type of pattern each network snapshot belongs to, we can precisely detect change points.

Problem Definition Given a network sequence $\{G_t\}_{t=0}^{T-1}$, where T is the number of snapshots, G_t is the snapshot of a dynamical network at time point t. E_t is the corresponding edges of G_t. Note that each G_t has the same node set V and $|V| = N$. Otherwise, if G_t has different but overlapping V_t, we assume that $V = V_0 \cup V_1 \cup V_2 ... \cup V_{T-1}$. Our goal is to find a set $S \subset \{1, 2, ...T-1\}$ such that $t \in S \iff P_t \neq P_{t-1}$, here P_t is the label of the pattern to which G_t belongs. That is, detection of change points means

to find all the time points whose pattern label is different from its predecessor's.

3.2 Soft Summary on Temporal Compact Subsequence

Even the pattern that govern the evolution of networks is invisible, we can still design an algorithm to find change points. Firstly, it is known that though there exists "GeneralChange", owing to temporal dependency, network snapshots under the same pattern are still highly coherent so as to form temporally compact subsequence. Thus, it is expecting to find ways to segment the snapshot sequence by summarizing subsequences. Intuitively, pair-wise clustering technique can be used to segment the network sequence to find the change points. To be more specific, the target is to encode the subsequences of snapshots with the same pattern label so long as there's a certain similarity to their features. Hereby, we define the **Soft Summary Scheme** as follows:

Definition 3.1. Given a network sequence $\{G_t\}_{t=0}^{T-1}$, we want to find an summary scheme $f : G_t \rightarrow f(G_t)$ to map each network a nominal, such that the following objective is minim

$$\min_{f} \sum_{\substack{0 \le i,j \le T-1 \\ d_{G_i,G_j} \le \epsilon}} \delta(f(G_i), f(G_j)), \tag{1}$$

$$\delta(x,y) = \begin{cases} 1 & x \neq y \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where ϵ is a threshold parameter which represents to what extent the dissimilarity would be tolerated. And d is the distance metric, which will be discussed in details later. We note that, networks that are not temporally adjacent may gather together in one cluster, which is undesirable for our task. Next, the temporal information will be added to refine the summary result. An overview of the proposed framework is provided in Fig. 2.

3.3 Feature Extraction and Similarity Measure

It is critical for the proposed soft summary scheme to determine what features to be extracted and how to compare the extracted features. As for feature extraction, there are several choices.

Scalar values such as mean degree, mean geodesic distance, mean local clustering coefficient, are too rough to capture much information about a dynamical network. Thus, based on those features, change-point detection approaches may fail to find some changes. In [13], it is manifested that the methods that just simply utilizes those scalar features performs poorly with high false negative rates even for significant structural changes. In fact, a network is not a set of elements but an interactive system on which information diffuses and influence flows, simply viewing edges as feature vectors ignores the interactions of the units in networks and the holistic network structure. Besides, adjacency matrix or its variants can also be used as the representation for networks [8]. However, owing to the high dimensionality and sparsity of real-world networks, it is clearly unsuitable for further network data analysis. We need to find a feature which not only contains abundant information about the network but also be low-dimensional.

To the best of our knowledge, the impacts of different events vary in range. That is, the changes can be located at different level of networks, some changes are related just with a small spatial extent on network. It is of particular interest not only to find the change points corresponding to an event, but also the top-k nodes, edges, or parts of the graphs that contribute most to the change. To this end, we extract features at local level to detect both local and global changes. Precisely, we choose the relative importance of the nodes in the snapshots as local feature. There are many ways to measure the node importance, such as PageRank [12] which computes a global ranking of all web pages regardless of their content, based solely on their location in the Web's graph structure, LeaderRank [11], an adaptive and parameter-free algorithm to rank users or quantify user influence in social networks, ClusterRank [3], etc. The motivation behind our choice is that, from the viewpoint of network science, node importance indicates the role of a node in the network and is intimately related to the node's neighborhoods at different scales. The change in structure can be reflected in the change of node importance. Hence we measure the similarities of network snapshots by considering the change of the roles individuals play in the snapshots. By using a specific measure of node importance, we get the feature sequence $\{F_t\}_{t=0}^{T-1}$ corresponding to the dynamical network $\{G_t\}_{t=0}^{T-1}$.

As most of random walk based ranking methods suggest the probability explanation of node importance (also known as node centrality [6]), then the feature vector consisting of the scores of node importance can be treated as the probability distribution where each score is the probability of a node being visited by a random walker (Note that, the node importance scores are normalized to abide by the probability distribution rule). Consequently, the distance between two snapshots is measured by comparing their corresponding probability distributions. Here we use Jensen-Shannon divergence to compute the distance as follows:

$$d_{G_i,G_j} = d_{F_i,F_j} = JS(F_i||F_j) = \frac{1}{2}\{D_{KL}(F_i||\frac{F_i+F_j}{2}) + D_{KL}(F_j||\frac{F_i+F_j}{2})\} \tag{3}$$

where

$$D_{KL}(F_k||F_l) = -\sum_{i=1}^{N} F_k(i) \ln \frac{F_l(i)}{F_k(i)} \tag{4}$$

Eq. 4 can be viewed as the weighted summation of the feature difference for node i between two snapshots, in which the influential nodes are weighted more. Thus, the more important the feature $F_k(i)$, the more contribution the i^{th} term to the distance formulated by Eq. 4. For the convenience of computing, we convert the distance measure in Eq. 3 to similarity using the (Gaussian) radial basis function kernel as follows:

$$K_{P,Q} = \exp(-\frac{||P-Q||_2^2}{2*\alpha^2}). \tag{5}$$

Then the similarity measure is formulated as:

$$sim_{P,Q} = \exp(-\frac{d_{P,Q}^2}{2*\alpha^2}), \tag{6}$$

where α is a free parameter representing the width of the Gaussian kernel.

Algorithm 1: Hyper-network Construction

Input: Network Feature sequence $\{F_0, F_1, F_2 \ldots F_{T-1}\}$,free parameter α
Output: hyper-network G_{hyper}
Construct G_{hyper} with N=T, Initialize weights $W = \mathbb{R}^{T \times T}$
for $i = 0$ *To* $T-1$ **do**
  for $j = 0$ *To* $T-1$ **do**
    update $W_{ij} = sim_{F_i,F_j} = \exp(-\frac{d^2_{F_i,F_j}}{2*\alpha^2})$
  end
end
return G_{hyper} whose weighted adjacency matrix is W

3.4 Hyper-network Construction

As one can see, Eq. 1 is an integer programming problem shown to be NP-hard. Inspired by [10], we transform this problem into community detection on the hyper-network represented by G_{hyper} :

$$\min_{\mathbf{y} \in \mathbb{R}^T} \mathbf{y}^\mathsf{T}(D - W)\mathbf{y} \quad s.t. \quad \begin{cases} \mathbf{1}^\mathsf{T} D\mathbf{y} = 0 \\ \mathbf{y}^\mathsf{T} D\mathbf{y} = 1 \end{cases} \tag{7}$$

Here, each node of G_{hyper} represents a snapshot of the original network sequence, and the edges of G_{hyper} is weighted by the similarities between pairs of snapshots. W is the corresponding weighted adjacency matrix of G_{hyper}. D is a diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$. $\mathbf{y} \in \mathbb{R}^T$, and $\mathbf{y}_i = f(G_i)$. And $\mathbf{1}$ is a vector of all 1's. Following [10], these constrains added here are to avoid trivial solutions. In practice, for Eq. 7, one usually compute several (e.g., K) eigenvectors as embedding of G_{hyper} , and then utilize some clustering algorithms on the embedding space to find the clusters (communities in G_{hyper}). In general, the embedding space are low-dimensional, for K is much smaller than N. In this way, each snapshot is summarized (labeled) by its community identity in G_{hyper} .

There are several simple clustering algorithms available to identify the clusters of hyper-nodes corresponding to the communities in G_{hyper}, e.g., Discretize, KMeans, Mean-shift and DB-Scan as well. Among them, we recommend Discretize for this method is less sensitive to random initialization than KMeans, and it do not introduce more parameters than DB-Scan and Mean-shift.

3.5 Serialization and Detection

In Soft Summary Scheme, we cluster the networks ignoring the temporal information, which is not enough to identify possible change points. Because the resultant clusters may include some snapshots that are similar to each other but separated by other clusters on timeline. Next, we serialize the clustering results by encoding the network sequence with corresponding community identity (some nominal values like 'A', 'B', etc). Then the encoding result naturally gives a segmentation of the sequence, as Fig. 4 shows. The alteration of the community identity in time reflects the boundary of the segments. Namely, the beginning of one segment indicates a change point. More formally:

Algorithm 2: Change-point Detection Framework

Input: parameter k, α and network sequence $\{G_0, G_1, G_2 \ldots G_{T-1}\}$
Output: change points **ChangeSet**
for $t = 0$ *To* $T-1$ **do**
  Extract Summary Feature F_t of G_t ▷ see section 3.3
end
Construct G_{hyper} with adjacency matrix ▷ see section 3.4
Solve optimization problem Eq. 7 by detecting communities on G_{hyper} ▷ see section 3.4
Serialize the community detection results
{ Below: Detect all the change points }
$p = C_0$
ChangeSet $= \emptyset$
for $t = 1$ *To T-1* **do**
  if $C_t = p$ **then**
    continue
  end
  else
    $p = C_t$
    ChangeSet $\cup \{t\}$
  end
end
return ChangeSet

Definition (Detection based on Community detection on Hyper-network) Given the community detection result of the hyper-network, we serialize the hyper-network node into a summary sequence $\{C_t\}_{t=0}^{T-1}$ (where C_t is the cluster label of G_t) by combining the temporal information. Thus the time point $t \in \{1, \ldots T-1\}$ will be flagged out if $C_t \neq C_{t-1}$.

Algorithm 2 gives the details of the whole procedure and Fig. 2 give a graphical depiction of our scheme.

4 EXPERIMENTS AND EVALUATION

This section evaluates the proposed approach on one synthetic network and two real-world networks with ground truth events and change points.

4.1 Synthetic Data and Real-world data

Synthetic Data We manually construct a series of small synthetic networks as the snapshots of a dynamical network with 9 nodes and length $T = 14$. Moreover, each snapshot is unweighted and undirected. As shown in Fig. 3, we incorporate 8 either global or local changes in the network sequence.

MIT Proximity networks MIT proximity network is extracted from The Reality Mining dataset[5] collected by The Reality Mining project, which depicts the proximity between 94 subjects including faculty and graduate students via Bluetooth device discovery scans. More precisely, the edges of the network denote physical proximity of pairs of subjects and are weighted by the numbers of scans of that particular week. According to the findings on the daily Bluetooth data in [4], it is suggested that the basic evolutionary timescale is about one week. Therefore, in this work, we use one week as the time scale to extract a sequence of weekly networks

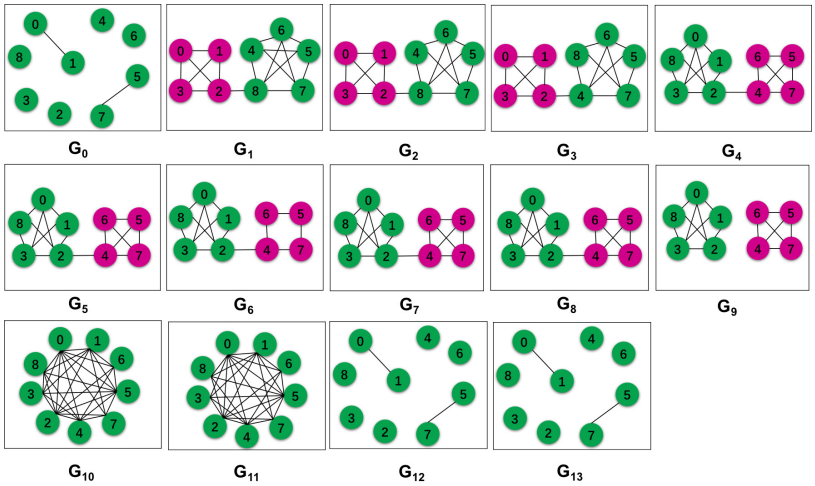

Figure 3: Synthetic network sequence. $T = 14$ and $N = 9$. Noticeable Community Structure(Global change) at time point 1, Position of two nodes in one community changes(Local change, isomorphic) at time point 3, One node shift from one community to another(Global change, isomorphic) at time point 4, link within one community reduces(Local change) at time point 6, Link within one community increases(Local change) at time point 7, Link between the two community breaks(Global change) at time point 9, The two community combine(Global change) at time point 10, Almost all the links broken(Global change) at time point 12.

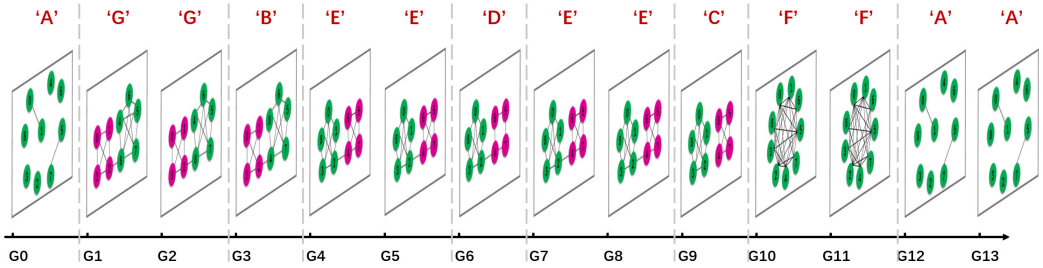

Figure 4: Serialized community detection results in detecting change points on synthetic networks based on our method. The corresponding community label of each G_t is given by the red letter over it. The segments of the sequence is given by grey dotted vertical lines. There are in total 8 changes$\{1, 3, 4, 6, 7, 9, 10, 12\}$. Each change point is a beginning of one segment.

Table 1: Comparison of the four methods on three dynamic networks. On synthetic networks (Toy), we choose the width of the Gaussian kernel $\alpha = 0.001$ and number of eigenvectors $K = 7$; on MIT proximity networks (MIT), $\alpha = 0.001, K = 7$; on Enron email networks (Enron), $\alpha \approx 0.01284, K = 7$. Note that in the three network sequence, we all choose the inverse importance rank as features. As for the parameter (i.e., window size w for short) in GHRG, On Toy, we choose the threshold $w = 3$; on MIT, $w = 4$; on Enron, $w = 4$. And for EigenBehavior, we do experiments on three different features (EgoNet, Degree and ClusterCoeff) and we chose the best result for $F1$ on each dynamic network.

Method	Toy			MIT			Enron		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
DynSnap	50	12.5	20	**66.67**	10	17.39	46.43	72.22	56.52
GHRG	100	37.5	54.55	41.94	65	50.98	42.86	16.67	24
EigenBehavior	75	37.5	50	50	10	16.67	**72.73**	44.44	55.17
OUR	**100**	**100**	**100**	55.56	**100**	**71.43**	56.67	**94.44**	**70.83**

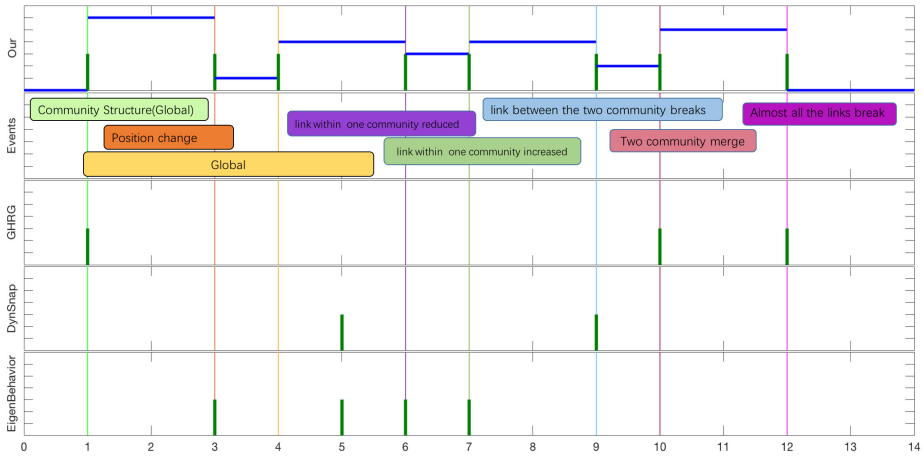

Figure 5: Change Point Detection on synthetic networks. The change points detected using the Four methods (Our, GHRG, DynSnap, EigenBehavior) are marked with green vertical bars. The community clusters obtained by our method are also shown in the figure with blue horizontal lines. We also indicate the known events by the colored vertical bars spanning all methods.

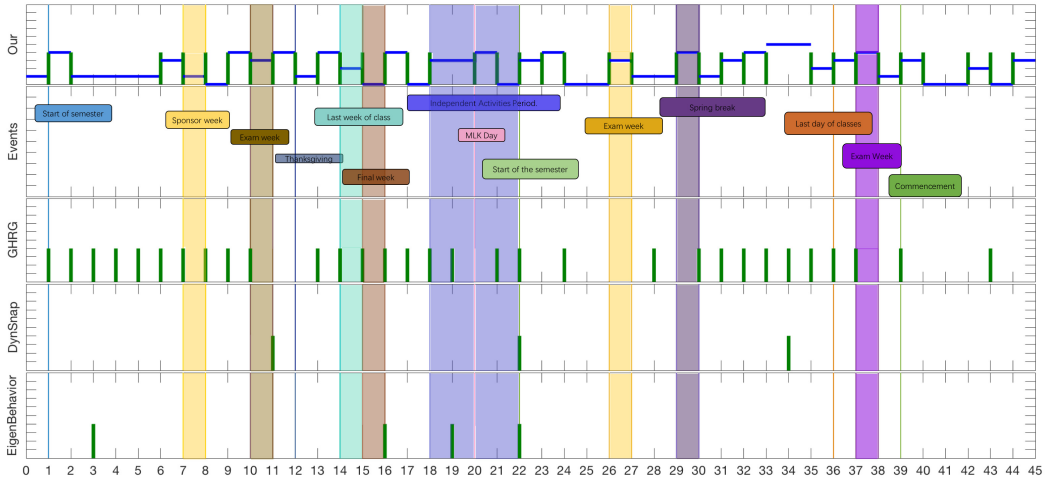

Figure 6: Change-point Detection on MIT proximity network. The labels are given as in Fig. 5.

from 30th August 2004 to 10th July 2005. As a result, we obtain $T = 45$ snapshots from the dataset. During the time, there are some known external events such as exam period and spring vacation from the MIT academic calendar or internal Media Lab events as shown in Fig. 6. Then the time corresponding to each event is labeled as the ground truth change point.

Enron Email networks The Enron Email Dataset is comprised of emails among many senior managers of the Enron energy company and was made public during the legal investigation. Different from MIT proximity, the time span of Enron Email Dataset is very long. Here we extract a sequence of monthly networks with $T = 44$ and $N = 147$ from November 1998 to June 2002, where each network is a snapshot of the communication between the managers within the corresponding month. Fig. 7 shows the major events (labeled as the ground truth change points) in this dataset.

We note that there may be some unreported events in the dataset, then the ground truth will be incomplete. For example, the MIT ACADEMIC CALENDAR 2004-2005 just use "SUMMER SESSION (incl. Exam Period)" to record the period JUNE 6 (Mon) - AUG 16 (Tues) so that we don't have information about the events that happened in that period.[1] That is, the "false positives" are not necessarily incorrect as [7] says. Thus, for a fair comparison, we compare our model and the baselines by using the comprehensive evaluation measure $F1$.

[1] https://web.archive.org/web/20041009182037/http://web.mit.edu/registrar/www/calendar.html

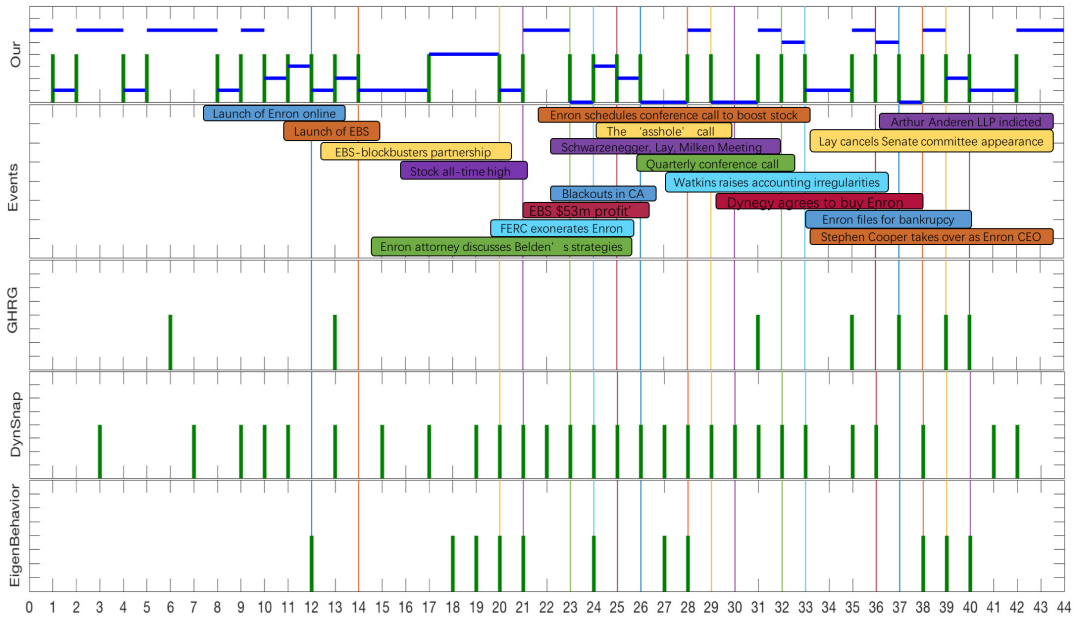

Figure 7: Change-point Detection on Enron email network. The labels are given as in Fig. 5.

4.2 Results

To evaluate the proposed method, we systematically compare it with three aforementioned detection algorithms (GHRG, DynSnap and EigenBehavior) on the synthetic and real-world datasets. The main results are reported in Table 1 where the methods are required to identify the exact change positions. We see that our approach outperforms the baseline methods. By examining the recall rate specifically, the efficiency of our method is remarkably better than the others'. From Fig. 5, closer examination of the change points detected with each of the baseline methods reveals that for the synthetic network sequence GHRG only picks out 3 change points corresponding to global changes, i.e., the emergence and split of community structures, while DynSnap distinguishes one global-change point. Although EigenBehavior identifies both global (one) and local changes (two), it still misses a large proportion of change points. In contrast, our method detects all types of change points with no false detection.

For MIT proximity network sequence, it is shown in Fig. 6 that the baseline methods either miss the majority of the events or identify much additional change points. In particular, DynSnap and EigenBehavior only detect a few change points include Exam week end and Start of the semester and Final week end, which explains their relatively high precision. And GHRG fails to detect the end of Exam Week (indicted by week 11) and the start of Spring vacation (indicted by week 29). In comparison to GHRG, our method achieves better recall with higher precision. By comprehensive evaluation of precision and recall, we can see that our method outperforms the three state-of-the art methods.

The result for the four methods on Enron are shown in Fig. 7. Different from the MIT proximity network, we find GHRG and EigenBehavior behaves poorly in recall. They miss many change points. Both our method and DynSnap detect most of the change points but wrongly treat some points as change points. But the number of regular events being identified as change points in our method is less than DynSnap. Moreover, our method finds more changes in the period before the launch of Enron online (Nov 1999, month index is 12). By looking at the email communication records during those days, we find that the network structure changes dramatically, which may explain why both our approach and DynSnap detect a lot of changes during that period. However, our detection highly coincides with the scheduled events after the launch of Enron online.

5 CONCLUSIONS

In this paper, we developed an unsupervised similarity based change point detection method for dynamically evolving networks. A hyper-network clustering framework, along with its construction based on structural feature extraction and similarity measurement, was proposed to summarize the network sequence and then detect change points. Different from existing topological feature assembling methods, we use the role of a node to characterize the local information on networks and the role distribution of the nodes to capture the global information. This way, by quantifying the change in this feature for network snapshots, our method can effectively identify the global and local changes in the sequence. The experimental results showed the advantages of our method over existing methods.

ACKNOWLEDGMENTS

This research was partially supported by National Natural Science Foundation of China No. 61573107, SWPU Youth Innovation Project No.2015CXTD06 and Applied Basic Research Foundation of Sichuan

Provincial Science and Technology Department No.18YYJC1147.
Ping Li is the corresponding author.

REFERENCES

[1] Leman Akoglu and Christos Faloutsos. 2010. Event detection in time series of mobile communication graphs. In *Army science conference*. 77–79.
[2] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2013. Evolving networks: Eras and turning points. *Intelligent Data Analysis* 17, 1 (2013), 27–48.
[3] Duan-Bing Chen, Hui Gao, Linyuan Lü, and Tao Zhou. 2013. Identifying influential nodes in large-scale directed networks: the role of clustering. *PloS one* 8, 10 (2013), e77455.
[4] Richard K Darst, Clara Granell, Alex Arenas, Sergio Gómez, Jari Saramäki, and Santo Fortunato. 2016. Detection of timescales in evolving complex systems. *Scientific reports* 6 (2016), 39713.
[5] N Eagle, A Pentland, and D Lazer. 2006. Infering social network structure using mobile phone data. *Proc. of National Academy of Sciences* (2006).
[6] Linton C Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41.
[7] David Hallac, Jure Leskovec, and Stephen Boyd. 2015. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 387–396.
[8] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. 2013. Deltacon: A principled massive-graph similarity function. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 162–170.
[9] Ze Li, Duoyong Sun, Renqi Zhu, and Zihan Lin. 2017. Detecting event-related changes in organizational networks using optimized neural network models. *PloS one* 12, 11 (2017), e0188733.
[10] Chuanren Liu, Kai Zhang, Hui Xiong, Guofei Jiang, and Qiang Yang. 2016. Temporal skeletonization on sequential data: patterns, categorization, and visualization. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 211–223.
[11] Linyuan Lü, Yi-Cheng Zhang, Chi Ho Yeung, and Tao Zhou. 2011. Leaders in social networks, the delicious case. *PloS one* 6, 6 (2011), e21202.
[12] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.
[13] Leto Peel and Aaron Clauset. 2015. Detecting Change Points in the Large-Scale Structure of Evolving Networks.. In *AAAI*. 2914–2920.
[14] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. 2015. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics* 7, 3 (2015), 223–247.
[15] Yu Wang, Aniket Chakrabarti, David Sivakoff, and Srinivasan Parthasarathy. 2017. Fast Change Point Detection on Dynamic Social Networks. *arXiv preprint arXiv:1705.07325* (2017).
[16] Yu Wang, Aniket Chakrabarti, David Sivakoff, and Srinivasan Parthasarathy. 2017. Hierarchical Change Point Detection on Dynamic Networks. In *Proceedings of the 2017 ACM on Web Science Conference*. ACM, 171–179.
[17] Yunli Wang and Cyril Goutte. 2017. Detecting Changes in Twitter Streams using Temporal Clusters of Hashtags. In *Proceedings of the Events and Stories in the News Workshop*. 10–14.