2016 IEEE 16th International Conference on Data Mining Workshops

Segmenting Sequences of Node-labeled Graphs

Sorour E. Amiri, Liangzhe Chen, B. Aditya Prakash
Department of Computer Science, Virginia Tech
Email: {esorour, liangzhe, badityap}@cs.vt.edu

***Abstract*—Detection of the changes in pattern of disease spread over a population network, Meme tracking and opinion spread on the Twitter network and product-rating-cascade over a social network are a few among the many embodiments of graph sequence segmentation problem with labeled nodes.**

Most of the previous approaches to network sequence segmentation are on plain graphs without considerations for the dynamics of propagation process. These approaches either fix observation scales or extract a long list of expensive features. In this paper, we propose SNAPNETS a *parameter free*, and *comprehensive* algorithm, to find segmentation of networks sequences with node labels such that adjacent segments are different in characteristics of nodes of each label. Our method leverages a principled, multi-level, flexible framework which maps the original problem to a path optimization problem over a weighted DAG.

Extensive experiments on several diverse real datasets show that our method finds cut points matching ground-truth or meaningful external signals outperforming non-trivial baselines.

I. INTRODUCTION

Let us set up the research problem in the following manner: Given a sequence of influenza infections and the underlying contact network of who-can-infect-whom, how are we able to help a public health expert when the infection patterns change possibly due to a virus mutation or effects of vaccination? The answer to such question is important in designing efficient immunization strategies based on understanding the virus propagation and consequences of treatment methods.

A trivial approach may be limited to segmentation of sequences based on the features of nodes in the sequence [1], [2]. Nevertheless, taking the dynamics of the underlying social network into account can help us gain more accurate results and deeper insight into specific patterns of propagation. An example of such intricate dynamics is disease spread in a tree-like fashion among elderly until a certain day, afterward changing into clique-like propagation among the younger individuals.

Segmentation of data sequences is an essential tool in better understanding the evolution of datasets. The study of segmentation methods in the context of graphs has many applications in epidemiology (public health issues), social media analysis (rumor/meme propagation in Twitter) and cyber security (anomaly detection and anti-malware safeguards). In this paper, we study the problem of segmenting a graph sequence with varying binary node-label distributions. For diseases, the labels are 'infected'/'active' (1) & 'healthy'/'inactive' (0), and the network can be the underlying contact-network. Our problem is:

PROBLEM 1: SEGMENTATION

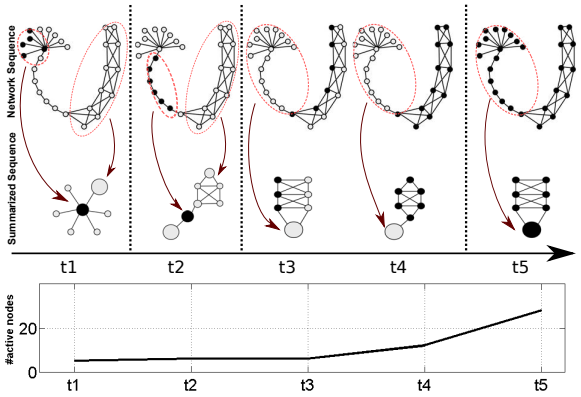


Figure 1: **TOY EXAMPLE: SNAPNETS automatically identifies four significant steps of the network sequence. The extracted time series (e.g. #active nodes) can not capture a proper segmentation. Gray nodes are inactive (i.e. label** 0**), and black nodes are active (i.e. label** 1**).**

***Given*:** a sequence $\mathcal{G}$ of networks $G_1, G_2, \ldots, G_T$ with labeled nodes,

***Find*:** the best segmentation, which captures different patterns of node labels in $\mathcal{G}$ such that adjacent segments have different characteristics of nodes with the same label.

TOY EXAMPLE. Given $\mathcal{G} = \{G_1, G_2, G_3, G_4, G_5\}$ with 0, 1 labeled nodes (Fig. 1 - top row), Find the best segmentation. *Answer:* $c^* = \{t_1, t_2, t_3, t_5\}$ that means segments are $\{[t_1, t_2), [t_2, t_3), [t_3, t_5), [t_5, t_5]\}$. This answer captures the fact that the label assignment of nodes has four significant steps in the sequence: At the beginning a central node in a star and some of its spokes have the label 1; next, low degree nodes in a chain-shaped manner got label 1 (structural change). In the third segment, the label moves to another community of the graph (community change). Finally, the whole graph gets label 1 which indicates the increase of node activation rate in the network (rate changes). Note that even though the 'active' subgraphs in t_2 and t_3 are both chains, their roles in the entire graph are different and hence we have a cut-point there. In t_2 the active chain is a bridge between two parts while the chain in t_3 is part of a near-clique community (role change).

Any algorithm should have the following desired properties:

P1. Parameter-free: Find the best number of segments and segmentation without use of any parameter such as change threshold and time window.

2375-9259/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDMW.2016.152

Property	SNAPNETS	K-MEANS [7]	DYNAMMO [8]	VOG [9], [3]
Graph based	✓	x	x	✓
Parameter-free	✓	x	x	x
Comprehensive	✓	x	x	x

Table I: **Comparison of SNAPNETS with some alternative approaches.**

P2. Comprehensive: Use the entire snapshot for segmentation, instead of merely active subgraphs.

Also, we want the algorithm to run in a reasonable amount of time. As pointed out in the related work, this problem has been barely (if at all) studied in literature, especially as we assume the number of segments and length of possible segments in the best segmentation are *not given* and we must calculate them automatically. In any case, most work silently ignores the actual dynamics and patterns of the propagation process (as observed through node labels). Moreover the limited recent work on this topic needs expensive feature extraction [3], or lock-in observation scales by just looking at one level of granularity [4] or do not scale [5]. Further they do not take the *entire* graph into account—typically they just focus on subgraphs with same label [6]. Tab. I shows a brief comparison of SNAPNETS with alternative algorithms.

We present SNAPNETS which satisfies all the properties. It is a novel multi-level approach which summarizes the given networks/labels in a very *general way* at *multiple different time-granularities*, and then converts the problem into an appropriate *optimization problem* on a data structure. A strong advantage of this framework is that it allows us to *automatically* find the right number of segments avoiding over or under segmentation in a very systematic and intuitive fashion. Further it gives naturally interpretable segments, enhancing its applicability. Finally, we also demonstrate SNAPNETS's usefulness via multiple experiments on diverse real-datasets.

The rest of the paper is organized as follows: We first give an overview of our approach. We then describe SNAPNETS in detail followed by empirical results. We then discuss related work and conclude.

II. OVERVIEW AND MAIN IDEAS

For sake of simplicity, we focus on the case when nodes have binary labels i.e. active: 1 or inactive: 0. Also, for ease of description, we assume that the network remains constant through time[1]. Finally, we assume that nodes can even *switch* between labels freely (c.f. Fig. 1). This means that the label assignment dynamics can be complex and we can handle both progressive/non-progressive scenarios: e.g., in the fundamental **S**usceptible-**I**nfected (SI) or **I**ndependent **C**ascade (IC) virus propagation models (where nodes can only go from inactive/healthy to active/infected) and the 'flu-like' **S**usceptible-**I**nfected-**S**usceptible (SIS) model where infected nodes can get healthy again. First, we give some notations (Table II) and useful definitions:

Definition 1 (Act-snapshot): $G_i(V_i, E_i, \mathbf{L}_i)$ is an *Act-snapshot* at time stamp t_i. $V_i = \{v_1, v_2, \ldots, v_n\}$ and $E_i = \{e_1, e_2, \ldots, e_m\}$ are sets of nodes and edges of the graph G_i. $\mathbf{L}_i = [l_1, l_2, \ldots, l_n]$ shows the labels of nodes. l_j is 1 if v_j is active and 0 otherwise.

Definition 2 (AS-Sequence): Set $\mathcal{G} = \{(G_1, t_1), (G_2, t_2), \ldots, (G_T, t_T)\}$ is sequence of T *Act-snapshots*, where each observation (*Act-snapshot*) G_i has an associated time stamp $t_i \in \mathbb{R}^+$. WLOG, $t_1 < t_2 < \ldots < t_T$.

Definition 3 (Segmentation): A segmentation c of size m is a partition of time interval $[t_1, t_T]$ with m time stamps i.e. $c = \{t_{a_1}, t_{a_2}, \ldots, t_{a_m}\}$ where $a_i \in \{1, 2, \ldots, T\}$. The set of all possible segmentations is $\mathcal{C}$.

Definition 4 (Act-set$_i$): Act-set$_i$ contains the active nodes in *Act-snapshot* G_i i.e. *Act-set*$_i = \{v_j | l_j = 1\}$.

Definition 5 (Segment): A segment $s_{i,j}$ is a time interval between two time stamp t_i, t_j i.e. $s_{i,j} = \{[t_i, t_j) \mid t_i < t_j\}$. Set of all possible segments is $\mathcal{S} = \{s_{1,2}, s_{1,3}, \ldots\}$.
Hence, our problem is to automatically segment a given *AS-Sequence*. We give the big picture of our framework.

Table II: Symbols

Symbol	Description
$G_\cdot$, $G_\cdot^c$	the original and the summarized graph.
$\mathcal{G}$	a sequence of T *Act-snapshots* (i.e. *AS-Sequence*)
t_T	The last time stamp in *AS-Sequence*
c	A valid segmentation of time interval $[t_1, t_T]$
c^*	The best segmentation of time interval $[t_1, t_T]$
$\mathcal{C}$	The set of all possible segmentations.
$s_{i,j}$	A segment $[t_i, t_j)$.
$\mathcal{S}$	the set of all possible segments.
$\mathbf{F}_\cdot$	The feature vector of each *C-graph*
$\widehat{\mathbf{F}}_\cdot$	The feature vector of each segment.
$d(\cdot, \cdot)$	Distance between two segments.
$\mathsf{G_s}$	segmentation graph
s , t	Source and target nodes of the segmentation graph
ρ	The remained portion of nodes remained in summarized graph
λ_G	the leading eigenvalue of graph G
$l_\cdot$	is the label of a node.
γ	Lagrangian multiplier.

A. Overview of our method SnapNETS

We propose a "global" framework which looks at the entire *AS-Sequence* $\mathcal{G}$ and computes the correct segments. Note that, due to **P1**, we want to examine all possible segmentations $\mathcal{C}$ over all granularities. How to do this efficiently? Our first main idea is to use a graph data structure (called the *segmentation graph* $\mathsf{G_s}$) to efficiently represent the exponential number of all segmentations in space polynomial with respect to the length of the sequence. The nodes mainly represent the segments in $\mathcal{S}$, while the edge weights indicate the distance ('difference') between adjacent segments. Hence any segmentation is mapped to a path between start and end time in $\mathsf{G_s}$.

How to now compute the distance between adjacent segments $w(s_{i,j}, s_{j,k})$ (each segment will contain sets of graph

[1]Extending our results to multiple labels and varying network structure is interesting future work.

snapshots G_i)? Note that we want to use the entire graph due to **P2**. This is related to the problem of graph isomorphism. It is natural to compare the segments by examining extracted features such as number of nodes, cliques, diameter and so forth. This will require complex feature construction to correctly and sufficiently represent each graph snapshot. Nevertheless, despite the size of the graphs, patterns in the real-world are usually much less complex. Hence, our second main idea is to develop a *smaller* summary G_i^c which maintains important information in an efficient manner. Subsequently, we only need a few standard features to represent these summaries.

Finally, how to find the best path in $\mathsf{G_s}$? We need to define this best path and find it with an efficient algorithm. Our third main idea is to use the average longest path optimization problem on $\mathsf{G_s}$, as it intuitively regularizes the length of the path (number of segments) with the weight (difference between segments).

In short, we pursue 3 main goals:

Goal 1. *Summarizing* G_i: summarize each G_i to capture the structural and label dependent properties (Sec. III-A).

Goal 2. *Constructing* $\mathsf{G_s}$: extract the features of summarized graphs and compute the distance between adjacent segments; and then construct $\mathsf{G_s}$.

Goal 3. *Defining and extracting the best segmentation*: define the best segmentation (path) and find it efficiently (Sec. III-C).

Fig. 2 shows an overview of SNAPNETS (**Snap**ping **NET**work **S**equences).

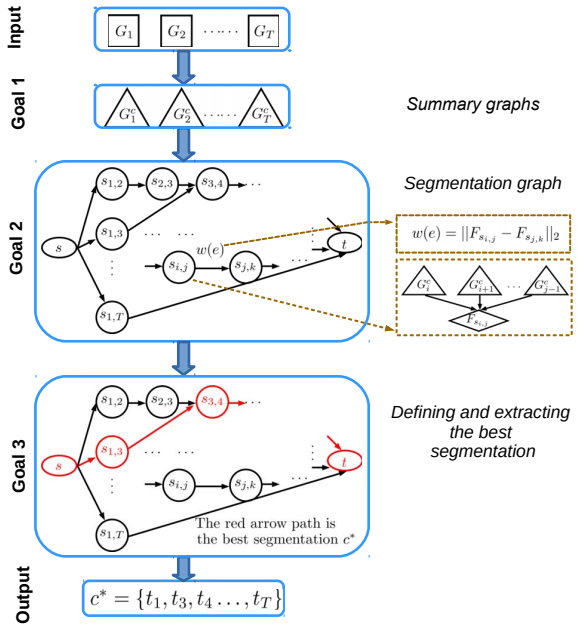


Figure 2: **SNAPNETS overview (Best viewed in color).**

III. SNAPNETS: DETAILS

In the following, we describe SNAPNETS in detail.

A. Goal 1: Summarizing Act-snapshots

As discussed before, the main purpose of summarization is to help comparing sets of *Act-snapshots* G_i. Here, we propose finding a graph G_i^c, which summarizes the structural properties as well as the nodes labels of G_i. Popular methods for graph summarization include graph sparsification [10] which try to carefully remove edges to reduce the graph's density while maintaining some properties. Nevertheless, these methods are designed for plain graphs without labeled nodes and it is not straightforward to modify them for *Act-snapshots*. Hence we adopt a different, *merging-based* approach which reduces the *number of nodes* instead while maintaining an intuitive and important property.

While summarizing, we would like to keep the properties of the *Act-snapshot* which come from the label distribution as well as the structure. In many real datasets such labels come from a diffusion/propagation process. Recent work [11] confirms that important diffusion characteristics of a graph (including the so-called 'epidemic threshold') are captured by the leading eigenvalue of the adjacency matrix, for almost all cascade models. The idea is that the pattern of activation is related to the eigenvalue as it depends on the relative placement and connections of nodes in the network. This naturally suggests that if the leading eigenvalue of the adjacency matrix of the summarized graph G_i^c and *Act-snapshot* are close, G_i and G_i^c will have similar properties.

Summarizing Act-snapshots via Coarsening: Motivated by the above, we propose to leverage the graph coarsening problem (GCP [12]) which takes a plain graph and computes a smaller summary maintaining the leading eigenvalue via successive merges of connected nodes into 'super-nodes'. This is very similar to our goal here. However, the GCP problem does not take the label of nodes into account. Thus, we modify it as follows (we use the same merge definition as in GCP):

PROBLEM 2: *Act-snapshot* SUMMARIZATION

***Given*:** an *Act-snapshot* $G_i(V_i, E_i, L_i)$, remained fraction of nodes ρ and Lagrange multiplier γ

***Find*:** a coarsened graph $G_i^c(V_i^c, E_i^c, L_i^c)$ s.t.

$$\underset{G_i^c, |V_i^c| = \rho |V_i|}{\arg\min} \left[|\lambda_{G_i} - \lambda_{G_i^c}| + \gamma \cdot \sum_{(a,b) \in E_i \text{is merged}} |l_a - l_b| \right] = \Delta\lambda + \gamma \cdot \Delta l$$

Here λ_G is the leading eigenvalue of graph G, l_a is the label of node a, and Δl is the error of merging nodes of different labels. This problem tries to merge nodes to minimize the change in the leading eigenvalue while also penalizing merges of nodes of different labels (hence maintaining the 'frontier' between active and inactive nodes). Intuitively, we want to merge nodes with similar activation characteristics, so that the final summary shows the overall activation pattern. This formulation allows us to be model-free and not assume any specific model (like IC/SIS etc.). The Δl term can help keep the same set of labels (0/1) in G_i^c as well (so helping

interpretability and making it easier to compare them). We call the coarsened *Act-snapshots*, G_i^cs, as *C-graphs*.

Jointly optimizing the two components in PROBLEM 2 ($\Delta\lambda$ and Δl) is a hard problem. Hence, we approximate the optimal solution by minimizing Δl first and then $\Delta\lambda$. Here, we leverage the *CoarseNet* algorithm [12] designed for the GCP problem which is near-linear in practice. We modify the algorithm by simply not allowing the merge of nodes with different labels. By this, we do not need to set up the multiplier γ. Apart from that, we still merge the rest of the best edge candidates as given by *CoarseNet*. Thus, we still maintain the leading eigenvalue as much as possible, while minimizing Δl (=0). This approach is intuitive, and scalable. In PROBLEM 2, ρ must be the smallest value which does not change λ_G much. Past work suggest that $\rho = 0.1$ works well [12]. Hence, we set ρ the closest value to 0.1 which gives *C-graphs* with the same number of nodes.

Figure 1 shows our summaries via PROBLEM 2 for the TOY EXAMPLE: the *C-graphs* clearly show the important non-trivial pattern changes in both the structural and label properties of the original graphs succinctly.

B. Goal 2: Constructing the segmentation graph

After summarizing the *Act-snapshots*, each segment in the *AS-Sequence* contains a set of *C-graphs*. How to find the distance between two such segments? Note that number of *C-graphs* in each segment is equal to the number of *Act-snapshots* in it. Finding the correct distance between two sets of large unlabeled graphs is a challenging problem [13]. Computing distance between two sets of *C-graphs* is even a harder task (due to the presence of labels). Fortunately, due to the small size and complexity of the *C-graphs* we can propose a cheap and effective approach: extract simple features from the *C-graphs*, and compute the distance between the average features of each set. After defining the distance between adjacent segments, we build a new graph $\mathsf{G_s}$ (the *segmentation graph*) to store the segments and distances information. As mentioned before, $\mathsf{G_s}$ can efficiently represent all the exponential number of possible segmentations in polynomial space.

1) Feature extraction of C-graphs: The feature extraction process on the *C-graph* G_i^c which is much smaller than G_i in size is much easier because of two main reasons,

(a) Efficiency: We *do not* need the complex features such as "number of particular substructures" (e.g. stars, maximal cliques, ladders, etc.) which are used in some previous methods [13]. Instead cheap features (e.g. degree, degree of neighbors, number of active nodes, etc.) are enough to represent the *C-graphs*. This is due to the fact that the *C-graph* maintains the relevant important properties of the graph effectively.

(b) Interpretation: Our *C-graphs* are of the same size (even if G_is are dynamic with various sizes we can do this by having different ρ_is)—this greatly helps in the *interpretation* of the features (compared to those on the unsummarized original graph) since changes in features such as number of active nodes are not due to different sizes of underlying original *Act-snapshots*. In other words, we can assume the feature values are standardized by the size of *Act-snapshots*.

We extracted multiple standard features used in past literature [14] and prune them by eliminating correlated features. Finally, we ended up with eight common and basic features for each G_i^c as its representative (See Tab. III). There are two types of features in the feature vector $\mathbf{F}$: *Structural* features (f_1-f_4), which show structural properties of the graph. *Label dependent* features (f_5-f_8), which represent the properties of *C-graphs* that depend on the node labels. Finally, we normalize them by range normalization to have a meaningful and fair comparison between the features [14]. Thanks to our flexible framework we can easily add any other features to detect extra patterns in a *AS-Sequence*.

2) Segmentation graph: We now describe how to construct the segmentation graph $\mathsf{G_s}$ to compactly store the segments and distances information and represent segmentations. Each *AS-Sequence* can be converted to a unique weighted DAG $\mathsf{G_s}(V_s, E_s)$ where:

Nodes (V_s): For each segment $s_{i,j} \in \mathcal{S}$, there is one node in the graph $\mathsf{G_s}$. We also add two extra nodes to the graph: a source node s and a target node t. Therefore, $V_s = \mathcal{S} \cup \{s, t\}$.

Edges (E_s): There is a directed edge from node $s_{i,j}$ to any node $s_{j,k}$. Also, the source node s links to all nodes with starting time t_1 and all nodes with ending time t_T links to the target node t. Hence, $E_s = \{e(s_{i,j}, s_{j,k})\} \cup \{e(\mathrm{s}, s_{i,j}) | t_i = t_1\} \cup \{e(s_{i,j},\mathrm{t}) | t_j = t_T\}$.

Edge Weights ($w(e)$): The weight of all edges from s or to t are zero. The weight of an edge from $s_{i,j}$ to $s_{j,k}$ is equal to the distance between sets of *C-graphs* in their corresponding segments i.e. $w(e(s_{i,j}, s_{j,k})) = d(s_{i,j}, s_{j,k})$.

How to get this distance? As explained in Sec. III-B1, we extract a feature vector $\mathbf{F}_i$ for each *C-graph* G_i^c. We compute the average feature vector over all the *C-graphs* in a segment as the segment's representative i.e. $\widehat{\mathbf{F}}_{s_{i,j}} = \frac{\sum_{a=i}^{j} \mathbf{F}_a}{(j-i+1)}$, where $\mathbf{F}_a$ is the feature vector of G_a^c in $s_{i,j}$. This representation has a natural interpretation as it captures the average 'pattern' of *C-graphs* of the segment. We define the distance $d(s_{i,j}, s_{j,k})$ between '$s_{i,j}$' and '$s_{j,k}$' as

$$d(s_{i,j}, s_{j,k}) = ||\widehat{\mathbf{F}}_{s_{i,j}} - \widehat{\mathbf{F}}_{s_{j,k}}||_2 \quad . \tag{1}$$

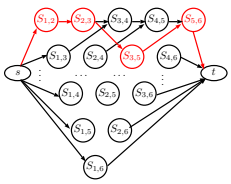


Figure 3

Figure 3 shows the $\mathsf{G_s}$ for our TOY EXAMPLE. Edge weights are not shown for clarity. Note that $\mathsf{G_s}$ is a DAG since its edges are directed and there is no cycle in it (as we cannot go back in time). Further, note that we can compute the distance for every edge in $\mathsf{G_s}$ independently and in parallel. Also, we need to compute the summary just once for each G_i, *not* for each segment in $\mathsf{G_s}$. Finally, let $\mathcal{P}$ be the set of all paths in $\mathsf{G_s}$ from s to t. We have the following observation:

OBSERVATION. Each path $p \in \mathcal{P}$ corresponds to a valid segmentation $c \in \mathcal{C}$ and for each $c \in \mathcal{C}$ there is a path $p \in \mathcal{P}$.

Type	ID	Name	Features description
Structural	f_1	Largest eigenvalue of the adjacency matrix	It indicates the structural and label dependent properties of *C-graphs*. We expect changes in eigenvalues of *C-graphs* in *AS-Sequence* due to the restriction of maintaining the frontier (Sec III-A). Hence this feature will encode how the labels are distributed with respect to the *Act-snapshot*.
	f_2	Number of edges	It measures the density of each *C-graph* which indicates how nodes got merged due to the label distribution in the *Act-snapshot*
	f_3	Entropy of the edge weight distribution	It encodes which edges in the original graph got merged. During *CoarseAct*, the edge weights change to maintain the leading eigenvalue (see Sec III-A). Therefore, the distribution of edge weights contains information about which edges are merged and how much merging happens.
	f_4	Average clustering coefficient	It measures the relative frequency of triangles in a *C-graph*. It shows when a new community get activated.
Label based	f_5	Number of active nodes	It counts the remaining active nodes after summarizing an *Act-snapshot*.
	f_6	Average PageRank of active nodes	It measures the structural importance of active nodes in *C-graphs*.
	f_7	Average degree of active nodes	It measures centrality among active nodes in *C-graphs*.
	f_8	Average degree of active neighbors of active nodes	It indicates the role and importance of active nodes.

Table III: **Features extracted to represent each summarized *Act-snapshot* (i.e. *C-graph*).**

C. Goal 3: Finding the best segmentation

We now need to define and find the best path (segmentation) in $\mathsf{G_s}$. After constructing the segmentation graph $\mathsf{G_s}$, here we define the problem of finding the best segmentation and show how to find such segmentation.

1) Average longest path: Note finding the best path is a different and independent question to that of defining edge weights. We define the best segmentation as follows:

PROBLEM 3: FINDING THE BEST SEGMENTATION

***Given*:** a segmentation graph $\mathsf{G_s}$

***Find*:** the average longest path from s to t in $\mathsf{G_s}$ i.e.

$$c^* = \arg\max_{c \in \mathcal{P}} \frac{\sum_{s_{i,j}, s_{j,k} \in} w(e(s_{i,j}, s_{j,k}))}{|c|} \quad (2)$$

Thus, PROBLEM 3 is the *Average*-Longest Path (ALP) problem on $\mathsf{G_s}$ (restricted to the path set $\mathcal{P}$). ALP defines the path quality as the average value of edge weights in the path. In other words, the segmentation quality is the average distance between its segments. Note that ALP is parameter-free: no further parameters are need to find the ALP. More importantly, the ALP also naturally *balances* the 'length' (weight) of the path (difference between segments) with the number of nodes in the path (number of segments).

An alternative 'parameter-free' optimization would have been the Longest Path (LP) problem: which will try to find the *longest* (heaviest) path in $\mathcal{P}$ (Eq. 2, without the denominator). However, the LP formulation will suffer from *over-segmentation*—it is biased by the number of segments in the path, in the sense that it tends to prefer longer paths with more nodes, irrespective of the edge weights [15]. In practice our observations confirm that LP contains unnecessary edges with low weight (see Sec IV). Our ALP objective is intuitive and overcomes the disadvantage of LP. Figure 3 shows the ALP for TOY EXAMPLE in red.

Finding the ALP in general graphs is NP-hard. This can be shown using a reduction from the LP problem which is NP-complete. Nevertheless, it can be solved in polynomial time on DAGs (recall that $\mathsf{G_s}$ is a DAG). Current state-of-the-art algorithm [15] computes the ALP from s to t in $\mathsf{G_s}$ with the negative cycle detection in running time $O(V_s^2.E_s)$. It takes the negative of all edge weights, and then iteratively adds the estimated optimal average to all edges until the derived graph has no negative cycle (see more in [15]). We use the same approach to find the ALP in our problem.

D. The complete algorithm

Algorithm 1 SNAPNETS

Require: *AS-Sequence* $\mathcal{G}$
Ensure: The optimal segmentation c^*
1: For each $G_i \in \mathcal{G}$ coarsen the *Act-snapshot* (Sec III-A)
2: Extract feature vector of segments in $\mathcal{S}$ (Sec. III-B1)
3: Generate the segmentation graph $\mathsf{G_s}$ (Sec. III-B2)
4: c^* = ALP($\mathsf{G_s}$, $\mathcal{G}$, s, t) (Sec. III-C)

In summary SNAPNETS has four main steps:
(Step 1) Summarize *Act-snapshots* in *AS-Sequence*.
(Step 2) Extract features from summarized graphs (*C-graphs*).
(Step 3) Construct the segmentation graph $\mathsf{G_s}$ and compute the distance $d(s_{i,j}, s_{j,k})$ between any adjacent segment based on their representatives ($\frac{\sum_{i=1}^{n} \mathbf{F}_i}{n}$).
(Step 4) Compute the best segmentation by finding the ALP.
Algorithm 1 shows the final pseudo code of SNAPNETS.

IV. EMPIRICAL STUDY

In this section, we design various experiments to evaluate SNAPNETS. We implemented SNAPNETS in MATLAB and Python. Our experiments were conducted on a 4 Xeon E7-4850 CPU with $512GB$ of $1066Mhz$ main memory.

A. Setup

Datasets. We collected a number of datasets from various domains such as social and news media, and co-authorship network to evaluate SNAPNETS. See Table IV.

The ground truth segmentations in the datasets are non-trivial. They are induced from complex structural changes such as activation in different parts of the *Act-snapshots* (*Higgs*), and varying role of active nodes e.g. change of active nodes centrality (*BA-degree*).

Moreover, detecting the number of correct cut points is also a difficult task itself.

Dataset	#Nodes	#Edges	Timesteps	GT
BA-degree	500	4,900	240 units	✓
Higgs	456,626	14,855,843	7 days	✓
Memetracker	960	5,001	165 days	
DBLP	369,855	1,109,452	13 years	

Table IV: **Datasets details. (GT == Ground Truth)**

(1) BA-degree. We activate highest degree and then lowest degree nodes on a random Barabasi Albert graph [16].

(2) Higgs [17] is a related tweets dataset (with the follower-followee network) when the Higgs particle was discovered. Nodes of the graph (i.e. twitter users) are active when they participate in related activities (e.g. retweeting, replying). The ground truth cut is the official release date of the discovery.

(3) Memetracker. It is the who-copies-from-whom blog and website network [18]. We consider a blog/website as active if it adopts the phrase 'hey can I call you joe'.

(4) DBLP is a co-authorship network [19] related to 'network'. Authors are nodes, and edges show the co-authorship relations. An author is active if she co-authors a related paper.

Baselines. To the best of our knowledge, there is no existing algorithm which has all the desired properties. Hence, we adapt three alternative approaches as baselines: time series, graph summarization algorithms and a variation of SNAPNETS.

(1) DYNAMMO uses the spike of reconstruction errors to find change points in time series [8]. We adapt it for our problem by using features in Tab. III for each *Act-snapshot* as a time series. The algorithm requires number of cut points as an input and we set it to the one SNAPNETS finds.

(2) VOG finds succinct descriptions of graphs in terms of common substructures [3], [9]. It does not work on labeled graphs. Hence, we extract active *sub-graph* of *Act-snapshots* and use VOG to find the 10 most important sub-structures. We output a segment if this set of sub-structure changes sufficiently (i.e is above a threshold which is set to be the one which gives the best results).

(3) SN-LP finds the **L**ongest **P**ath instead of the average longest path in $\mathsf{G_s}$ as the segmentation.

Metrics. We measure the performance by calculating the F_1 score of the set of detected cut-points with the ground-truth set (using the same methodology as [1]). Additionally, we perform case-studies as well.

B. Segmentation Results

Next, we give the results of SNAPNETS on our datasets. We calculate $F_1\ score$ to quantitatively evaluate the segmentations for datasets with ground truth. Further, for the rest, we show how our algorithm reveals interesting segmentations: we match the results with external news/events, and show that they are easily interpretable.

Dataset with ground truth	F_1 score			
	SNAPNETS	SN-LP	DYNAMMO	VOG
BA-degree	**1**	0.08	0	0.67
Higgs	**1**	0.15	0	0

Table V: **The F_1 score of the segmentation detected by SNAPNETS, variations, and baselines on datasets with ground truth. The higher value is the better.**

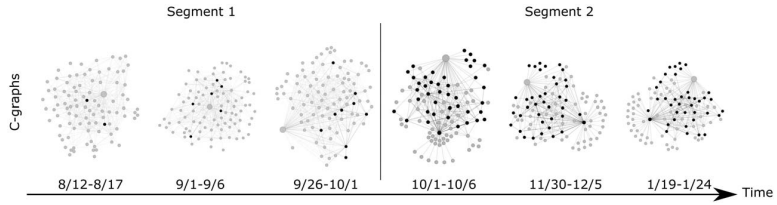
1) Quantitative analysis: In short, we notice in Table V that SNAPNETS has the best performance. Note that the baselines require some input parameters: such as the number of cut points (DYNAMMO), and difference threshold for outputting a cut point (VOG). We test a wide range of these input values and pick the ones that give the best results. Nevertheless, their performance is clearly worse than SNAPNETS. More details for each dataset follow:

***BA-degree*:** SNAPNETS detects the ground truth segmentation while three other baselines do not perform as well.

***Higgs*:** SNAPNETS finds the exact ground truth Jul. 4 ($F_1 = 1$), while other baselines perform much worse. Note that according to external news, there were rumors about the Higgs boson discovery from Jul. 2-4, these rumors make the ground truth harder to detect (for example VOG finds a cut point on Jul. 2 instead of Jul. 4). Further in the first segment, *C-graphs* have multiple near-clique structures of *active* nodes ($f_1 \simeq 0.9$, $f_5 = 0.2$) which demonstrates the existence of a rumor in the network (multiple small groups adopt the news). In comparison in the second segment, *C-graphs* become chain-like with many active nodes ($f_1 \simeq 0.02$, $f_5 = 0.9$). It suggests that many communities adopt the news (with nodes in the same community merged), and few bridge (inactive) nodes connect different communities, matching our expectation since the official announcement is evidently more influential.

2) Case studies: In short, SNAPNETS finds meaningful segments in multiple datasets from various domains with varied patterns of evolution (both structurally and in labels) while none of the baselines perform as well (for example, VOG finds no cut-point in *DBLP*, and DYNAMMO finds no cut-point in *Memetracker*).

***Memetracker*:** SNAPNETS finds a cut point on Oct 01, 2008, which matches the date of the televised debate between Joe Biden and Sarah Palin. In the first segment, *C-graphs* (Fig. 4a) are close to the case when all nodes have the same label—suggesting that few nodes got infected ($f_5 \simeq 0.1$). Furthermore, the low value of $f_8 \simeq 0.2$ indicates random nodes became active in the original graph. In the second segment, *C-graphs* are substantially sparser (i.e. f_2 dramatically dropped to 0.02) and contain large stars and leaf nodes with active centers. The size of the centers (the size of the nodes in Fig. 4a is proportional to the number of merged nodes) and average PageRank (f_6) in the *C-graphs* shows that many



(a) Visualization of *C-graphs* in the segmentation for *Memetracker*.

Feature	Seg. 1	Seg. 2
f1	0.93	0.06
f2	0.97	0.02
f3	0.09	0.97
f4	0.12	0.87
f5	0.10	0.95
f6	0.22	0.91
f7	0.45	0.76
f8	0.20	0.95

(b) The average feature values in each segment for *Memetracker*.

Figure 4: **Interpretation of the segmentation results found by SNAPNETS. (a) shows the *C-graphs* for the segmentations found for *Memetracker*. The vertical line is the detected cut point. Black nodes are active and gray ones are inactive. (b) shows the average feature values in each of the segment for *Memetracker*.**

active nodes got merged into important nodes to form hubs. For instance, we found, "CNN" and "BBC" websites became active in *Memetracker* and spread the meme to many nodes in the original graph.

***DBLP*:** We detect a cut point in the year 1997, which matches the publication time of the ground-breaking papers in network science [20], [21], [16].

V. Related Work

Time series mining, graph summarization, and dynamic graph analysis are related to our work.

Time series analysis: The research community has made great efforts in developing algorithms for different problems on time series data. These include algorithms for mining multivariate time series, summarizing time series with missing values [8], a generative analysis of time series, and many others. Among them, there is also much work about time series segmentation [22] or other specialized algorithms like motion capture sequences [1]. However, our problem is fundamentally different because we deal with sequences of *Act-snapshots*; and converting graphs (even without labels) to multivariate feature values while preserving the desired patterns is already a nontrivial task [13]. Hence time series segmentation algorithms cannot be easily applied for our problem to get meaningful segmentations.

Graph summary and sparsification: aims to find compact representations of graphs which maintain desired properties. The properties can be defined based on specific user queries [23], action logs [10], or more generally the encoding cost [4], weights of nodes and edges [6], the drop of the leading eigenvalue [12]. These algorithms help reducing the processing cost of large graphs, and maintain (sometimes amplify) the patterns in the graphs. Unlike these methods our summarization algorithm maintains the structural as well as label dependent properties of a graph.

Dynamic graph analysis is gaining much interest due to the evolutionary nature of many networks we see nowadays (see [24] for an overview). Many traditional machine learning tasks on static graphs have been extended to dynamic ones [2], [25], [26] such as the clustering and classification problems, link prediction, anomaly detection, trend mining. There has been work in finding time cut-points according to change of patterns in dynamic graphs. Ferlez et al. [5] and Sun et al. [27] use the MDL principle to detect the cut points when communities/partitions in the evolving network change abruptly; while [28] uses tensor decomposition to discover temporal communities in dynamic graphs. These work only on plain graphs (not labeled graphs) and are community-based. In contrast, in our problem, we study the patterns in a more general way not restricted to communities or clusters.

VI. Conclusions

We presented SNAPNETS, an intuitive and effective method to segment *AS-Sequences* with binary node labels. The main novelty of SNAPNETS is that it is the *first* method to satisfy the desired properties **P1**, **P2**: it finds the best segmentation automatically and taking the entire *Act-snapshot* into account. As our experiments show, it efficiently and accurately finds high-quality segmentations and gives useful insights in diverse complex datasets which indicates the ALP objective performs well. We would like to further scale up the algorithm and run it on larger datasets by designing a more efficient algorithm for ALP problem and parallelize the SNAPNETS. Our framework is flexible and it can be extended to handle varying network structure which is an interesting future work.

Acknowledgments. This paper is based on work partially supported by the National Science Foundation (IIS-1353346), the National Endowment for the Humanities (HG-229283-15), ORNL (Task Order 4000143330) and from the Maryland Procurement Office (H98230-14-C-0127), and a Facebook faculty gift. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the respective funding agencies.

References

[1] Y. Matsubara, Y. Sakurai, and C. Faloutsos, "Autoplait: Automatic mining of co-evolving time sequences," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 193–204.

[2] K. Henderson, T. Eliassi-Rad, C. Faloutsos, L. Akoglu, L. Li, K. Maruhashi, B. A. Prakash, and H. Tong, "Metric forensics: a multi-level approach for mining volatile graphs," in *KDD*, 2010.

[3] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, "Vog: summarizing and understanding large graphs." SIAM, 2014.

[4] W. Liu, A. Kan, J. Chan, J. Bailey, C. Leckie, J. Pei, and R. Kotagiri, "On compressing weighted time-evolving graphs," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 2319–2322.

[5] J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik, "Monitoring network evolution using mdl," in *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 1328–1330.

[6] Q. Qu, S. Liu, C. S. Jensen, F. Zhu, and C. Faloutsos, "Interestingness-driven diffusion process summarization in dynamic networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 597–613.

[7] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[8] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos, "Dynammo: Mining and summarization of coevolving sequences with missing values," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 507–516.

[9] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos, "Timecrunch: Interpretable dynamic graph summarization," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1055–1064.

[10] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 529–537.

[11] B. A. Prakash, D. Chakrabarti, N. C. Valler, M. Faloutsos, and C. Faloutsos, "Threshold conditions for arbitrary cascade models on arbitrary networks," *Knowledge and information systems*, vol. 33, no. 3, pp. 549–575, 2012.

[12] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian, "Fast influence-based coarsening for large networks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1296–1305.

[13] D. Koutra, J. T. Vogelstein, and C. Faloutsos, "DELTACON: A principled massive-graph similarity function," in *Proceedings of the SIAM International Conference in Data Mining. Society for Industrial and Applied Mathematics*. SIAM, 2013, pp. 162–170.

[14] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Effective graph classification based on topological and label attributes," *Statistical Analysis and Data Mining*, vol. 5, no. 4, pp. 265–283, 2012.

[15] D. Salvi, J. Zhou, J. Waggoner, and S. Wang, "Handwritten text segmentation using average longest path algorithm," in *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*. IEEE, 2013, pp. 505–512.

[16] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[17] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi, "The anatomy of a scientific rumor," *Scientific reports*, vol. 3, 2013.

[18] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 497–506.

[19] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila, "Finding effectors in social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1059–1068.

[20] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[21] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM computer communication review*, vol. 29, no. 4. ACM, 1999, pp. 251–262.

[22] X. C. Chen, K. Steinhaeuser, S. Boriah, S. Chatterjee, and V. Kumar, "Contextual time series change detection." in *SDM*. SIAM, 2013, pp. 503–511.

[23] W. Fan, J. Li, X. Wang, and Y. Wu, "Query preserving graph compression," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 157–168.

[24] C. Aggarwal and K. Subbian, "Evolutionary network analysis: A survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 10, 2014.

[25] C. C. Aggarwal and N. Li, "On node classification in dynamic content-based networks." in *SDM*. SIAM, 2011, pp. 355–366.

[26] P. Sarkar, D. Chakrabarti, and M. Jordan, "Nonparametric link prediction in dynamic networks," *arXiv preprint arXiv:1206.6394*, 2012.

[27] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: parameter-free mining of large time-evolving graphs," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 687–696.

[28] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra, "Com2: Fast automatic discovery of temporal ('comet') communities," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2014, pp. 271–283.