Mining Dynamic Discriminative Patterns in Evolving Global-State Networks

Minh X. Hoang, Nazli Dereli
University of California, Santa Barbara
{mhoang,ndereli}@cs.ucsb.edu

Abstract—The evolution of local states such as node and/or edge labels impacts the global state of a network. Thus monitoring only a discriminative/significant subgraph can save cost and still enable us to predict the global network state accurately. Moreover it is more meaningful to look at dynamic interactions since many real-world systems occur for a certain period of time only. Learning occurs at certain times in human brain, proteins interact temporally, people communicate at certain point of time, traffic congestions emerge in certain periods of day. In this study, we want to discover the impact of the subgraphs that show an identical dynamic behavior over time, on the global network state. We propose the concept of neighbors of a node in a subgraph in space and time, which is used to construct a dynamic subgraph search space. We then perform Metropolis-Hasting sampling algorithm in this search space to find discriminative time-evolving connected subgraphs that best predict the global state of a network. Experimental results show that our method effectively and efficiently extracts the subgraphs of interest.

I. MOTIVATION

Dynamic subgraphs with evolving local entities can efficiently give insights about the properties of the whole network, especially when the whole network is large, dynamic and costly to analyze. A dynamic network can be classified using only a single time-evolving subgraph that is statistically significant and powerful to predict the global state of the network. Such a method enables us to discover the dynamics of human brain as learning occurs by identifying cohesive brain regions that are activated and to classify a subject as a good or bad learner by using these cohesive brain regions. Similarly, we may find out about a traffic congestion in a city just by looking at couple of streets.

II. PROBLEM DEFINITION

Given a time-evolving global state network which has local labels on nodes and/or edges that can change with time and a global label for the whole network, we are trying to find the dynamic discriminative subgraphs that best predict the global label based on the changing local labels.

The goals of the project are:

2) Perform experiments on real-world datasets and verify the proposed approach.

III. RELATED WORK

Frequent subgraph mining has been studied extensively in the last decade and many methods have been proposed [1], [2], [3]. However, the output space of frequent subgraphs is often too huge for human to explore. Thus, scientists are becoming more interested in significant and discriminative subgraphs. Tan et al. [4] summarize 21 interestingness measures for subgraphs. Some early works mine significant subgraphs by filtering the space of frequent subgraphs produced by popular frequent subgraph mining methods. The best frequent substructures are chosen as features for classifying chemical compounds in [5]. ORIGAMI, an algorithm for mining the set of representative orthogonal patterns is proposed in [6]. Another approach to feature selection on frequent subgraphs is to take advantage of the submodularity quality criterion [7]. Mining significant subgraphs from the set of frequent subgraphs has a big bottleneck since the number of frequent subgraphs could be exponential. In addition, significant subgraphs with low frequency might be ignored. To cope with these problems, several different approaches are pursued. First, graphs can be converted into feature vectors, which can then be used to evaluate their significance [8]. Second, graphs can be grouped into smaller sets such that the frequency of graph patterns in each of these sets is higher [9]. However, it would be much better if the significant subgraphs can be mined directly from the graph dataset. Yan et al. propose a generalized framework which extract significant subgraphs without generating all frequent subgraphs using a smart heuristic to prune the search space in a branch-and-bound search [10]. Discriminative and essential frequent patterns are mined via a model-based search tree in [11]. There are two most popular threads of work to find subgraphs of interest. One thread of work focuses on finding discriminative graph patterns that help distinguish between two graph datasets of different classes, e.g. positive and negative [7], [5], [10]. Another thread is to find subgraphs that are significant by examining their p-value against a randomized graph dataset [9], [8].

More related to our work is the mining of evolving subgraphs [12], [13]. Bogdanov et al. introduce the problem of finding the highest-scoring connected temporal subgraph in an edge-weighted network whose weights evolve in time [11]. A subgraph mining method is proposed that is defined by some constraints in [14]. The dynamic evolution of subgraphs is captured by associating a temporal event type to each of them which are formation, dissolution, growth, diminution and stability of subgraphs from one time stamp to the next.

Global-State networks with local states on nodes/edges and a global state have been studied before [15]. The key difference
between this work and our problem is the dynamic evolution of the networks. In [15], a Global-State network consists of a set of snapshots, each of which has the same structure but different local and global states. They then try to find a set of discriminative subgraphs, such that each of these subgraphs can be used to predict the global state of a snapshot. In other words, both the snapshots and the discriminative subgraphs are static. On the other hand, in our problem, each of these snapshots is itself a dynamic network that could change over time. The changes include both structural and local labels on nodes/edges but the global state of the snapshot does not change. Further, we look for dynamic subgraphs that can change over time as well, but can predict well the fixed global state. To achieve this goal, we extend the definition of a neighbor of a vertex to both space and time. Therefore, the subgraph search space in our problem is significantly larger: the space of all subgraphs that can change over time. While these two problems are largely different, we adopt the idea of network-constrained decision trees (NCDT) from [15] to quantify the discriminative potential of a dynamic subgraph to capture the effect of the network structure on the evolution of the networks. In addition, we use Metropolis-Hasting sampling algorithm to combat the huge dynamic subgraph search space and find only a subset of dynamic subgraphs that have high potential.

IV. Preliminaries

Before discussing the proposed method, let us provide some guideline definitions.

Definition 1: Dynamic Global-State Network (DGS-network) is a series of network snapshots changing over time \( G = \{G_1, ..., G_n\}, S|G_t = (V,E,L_t)\), where \( G_t \) is the snapshot of the network at time \( t \); \( V \) and \( E \) are respectively the set of vertices and edges of \( G_t \); \( L_t \) is a labeling function that assigns each node and/or edge of \( G_t \) with a local state; and \( S \) is a global state of the whole evolving network \( G \).

Intuitively, a dynamic global state network is a graph with local labels on nodes or edges that evolve over time, and a fixed global label indicating some property of the whole network. For simplicity, we assume that the edges are undirected, only nodes are labeled, \( L : V \rightarrow \{1,-1\} \) and \( S \in \{1,-1\} \). Further, the structure of the network does not change with time, only the node/edge labels do. Our solution in the next section, nevertheless, can be easily extended to the case of real-value local and global states and changing structure. In case edges are labelled instead of nodes, such problem can be converted into our problem easily as well.

Definition 2: Dynamic Subgraph: Given a DGS-network \( G = \{G_1, ..., G_n\}, S|G_t = (V,E,L_t)\), a dynamic subgraph of \( G \) is a series of subgraphs \( P = \{P_1, ..., P_k\}, 1 \leq t_1 \leq t_k \leq n, P_t = (V_{P_t}, E_{P_t}), V_{P_t} \subseteq V, E_{P_t} \subseteq E, \forall u, v \in V_{P_t}, if (u, v) \in E then (u, v) \in E_{P_t}, and V_{P_t} \cap V_{P_t+1} \neq \emptyset \)

Note that a dynamic subgraph could have different set of nodes and edges at different timestamps, but must have at least one common node between any two consecutive timestamps.

We are interested in finding a dynamic subgraph that helps us predict the global state of the DGS-networks.

Definition 3: Dynamic Discriminative Subgraph (DDS): Given a dataset of DGS-networks with the same structure \( D = \{G^1 | V_{G_t} = V, E_{G_t} = E\} \), let \( f(P, L) \) be a structure-sensitive prediction function that predicts the global state of a DGS-network by using the labels on nodes and/or edges of \( P \). If \( C = \{G^k | f(G^k) = S^k\} \) is the set of correctly predicted networks, then the discriminative potential of a dynamic subgraph \( P \) is \( \phi(P) = \frac{|C|}{|D|} \). \( P \) is discriminative if \( \phi(P) \geq \theta \) for a user-provided threshold \( \theta \).

One drawback of the definition above is that one can use a whole DGS-network as a pattern, which could be huge. As a result, our goal is to extract a set of minimally discriminative dynamic subgraphs.

Definition 4: Minimally Discriminative Dynamic Subgraph (MDDS): A DDS \( P \) is minimally discriminative if \( \phi(P) \geq \theta \) and the set \( \{P | P \subseteq P, \phi(P') \geq \phi(P)\} = \emptyset \).

With these definitions, we will focus on finding the MDDS of a set of DGS-networks in the next section.

V. Proposed Method

The basic idea of our proposed method is to use Metropolis-Hasting sampling algorithm in the space of all possible dynamic subgraphs of the DGS-networks to find the MDDS. More exactly, the algorithm starts with a random connected subgraph in a single timestamp. It then either expands or shrinks the subgraph in time and space by adding or deleting nodes from the subgraphs. Each time a new dynamic subgraph is obtained, its potential is evaluated. Finally, the obtained dynamic subgraphs are checked for their minimality.

There are two questions in the above scheme. First, how can we evaluate the potential of a dynamic subgraphs? Second, which nodes can be added or removed when the dynamic subgraph is expanded or shrunk in space and time?

For the first question, we apply a modified version of Network-Constrained Decision Tree (NCDT) in [15]. In essence, NCDT is similar to a regular decision tree. The difference lies in the way a feature, i.e. a vertex, is chosen as a splitting node in the tree. More specifically, only the vertices that are neighbors of the vertices that have already been in the same path in the tree can be chosen as candidate feature for splitting. We apply the same technique, with an extension of the definition of neighbors in both space and time.

Definition 5: Neighbors of a vertex in time and space: Given a vertex \( v \) at timestamp \( t \), the neighbor of \( v \) within the same timestamp are the vertices that share an edge with it in \( G_t \). On the other hand, the neighbors of \( v \) at timestamp \( (t-1) \) and \( (t+1) \), are the vertex \( v \) itself in \( G_{t-1} \) and \( G_{t+1} \) respectively.

For the second question, given the current dynamic subgraph \( P = \{P_1, ..., P_k\} \), we need to find all the addable and removable vertices in space and time. Denote the set of neighbors of vertices in \( P_t \) at timestamp \( t \) as \( N_t \). Also, denote \( A_t \) as the set of addable nodes for \( P \) at timestamp \( t \). For the
timestamp \( t \) that has already been contained in \( P \), the set of addable nodes at timestamp \( t \) is as follows:

\[ A_t = N_t \setminus V_{P_t, t_1 \leq t \leq t_k} \]  
(1)

The dynamic subgraph \( P \) could also be expanded in time to timestamps \( t_1 - 1 \) and \( t_k + 1 \) if these timestamps exist. For these two timestamps, the addable nodes are the nodes in \( V_{P_{t_1}} \) and \( V_{P_{t_k}} \) respectively. In other words:

\[ A_{t_1-1} = V_{P_{t_1}} \] 
(2)

\[ A_{t_k+1} = V_{P_{t_k}} \]  
(3)

Deleting vertices from the dynamic subgraph \( P \) is more difficult since \( P \) must remain connected in both space and time. Assume that we are finding removable vertices at timestamp \( t \) of \( P \). We must assure that a removed vertex does not (a) disconnect the subgraph at timestamp \( t \) and (b) disconnect the subgraph at timestamp \( t \) with the subgraphs at timestamp \( t+1 \) and \( t-1 \) if exist. The former can easily be guaranteed by only allowing removal of vertices that are not cut-vertices in \( P_t \). The latter requires that the removed vertex is not the only vertex in the intersection of \( P_t \) with \( P_{t-1} \) or \( P_t \) with \( P_{t+1} \). Denote \( I_{t, t_1} = V_{P_{t_1}} \cap V_{P_{t_2}} \) as the intersection, i.e., the set of common vertices, between \( P_{t_1} \) and \( P_{t_2} \). Also, denote \( Y(t) \) as the set of vertices in \( P_t \) that are not cut-vertices. Then, the set \( D_t \) of removable vertices at timestamp \( t \) is:

\[ D_t = \{ v | v \in Y(t), I_{t-1,t} \setminus \{ v \} \neq \emptyset, I_{t,t+1} \setminus \{ v \} \neq \emptyset \} \]  
(4)

In case timestamp \( t-1 \) or \( t+1 \) does not exist, the corresponding conditions on \( I_{t-1,t} \) or \( I_{t,t+1} \) can be ignored.

Now, with the set of removable and addable vertices, we need to compute the potential of the corresponding obtained dynamic subgraphs, to perform sampling using Metropolis-Hasting algorithm toward the dynamic subgraphs with high potential. However, since the number of candidate subgraphs could be huge, this task will be computationally expensive. Thus, we apply the same heuristics in [15] to estimate the increase or decrease in potential of a subgraph when a vertex is added or removed. With this last note, our proposed method is complete. We will perform experiments to verify the validity and performance of our proposed method in the next step.

VI. EXPERIMENTAL RESULTS

A. Datasets

We evaluate DynaMinds on two real-world dynamic networks: (i) a highway transportation network from Los Angeles, California evolving during the month of April 2011, (ii) a brain network while learning a motor skill (short sequences of finger movements). Figure 1 summarizes the two datasets we worked with and Table I lists the sizes of them.

The original networks are transformed so that each edge becomes a node and each node becomes an edge since our current algorithm works with node labels. They are both thresholded as well to have binary node labels.

In original traffic network, nodes are the locations that contain a speed sensor and the edges are the roads connecting two of these locations. Given an edge and the average speed on this edge, the edge weights are calculated as statistical p-values. The significance of observing this average speed is measured according to the empirical distribution of average speeds on the edge over all time slices. The p-value is computed as the fraction of time slices in which an equal or higher speed is observed on the same edge. The lower the p-value of an observed speed, the more anomalous the speed is. Therefore, 0.1 is chosen as a threshold for this network. The edges that have p-values smaller than 0.1 are considered anomalous and may have congestion, they have value 1. The other edges have value 0. For the global state, the norm of the dynamic network is calculated as the average of p-values over all time slices for a period of month. For each sample (day), the number of p-values that are above and below this norm is counted. The samples that have high amount of values that are above the norm are chosen and their global state is set to 0 whereas the global state of the samples that have equal amount of values of that are above and below the norm is set to 1.

For our brain network, the dataset is fMRI images of twenty subjects acquired over the course of three different sessions/days as they are learning a motor skill (short sequences of finger movements). Brain networks are constructed by first parcellating the brain into 112 anatomically distinct regions which are represented as network nodes. Mean signal from each of these regions in a given session is partitioned into 25 time slices of approximately 3 minutes long each. Thus, a session consists of 25 time slices per subject. The network is fully connected and the interactions between nodes are represented with edge weights. These weights are determined by a measure of statistical similarity between activity of the region pair and it is defined as the coherence between the wavelet coefficient time series in each region which is between 0 and 1. For this study, we only used the networks of 18 subjects for the first session. 0.6 is chosen as a threshold which makes 2% of the edges active (having value 1) that is similar in the traffic network as well. The average of learning scores is used as the threshold for the global state (good/bad learner), the learning score is the time that it takes a subject to complete the learning task.

Distribution of Active Nodes: When we look at the distribution of active nodes over all time slices in each network, we observe certain time slices where the difference between the number of active nodes in positive samples and negative samples is relatively larger. These time slices are of interest to us since they might be more powerful to predict the global states. To prove this point, the relationship between difference in number of active nodes and visit percentage of

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATASET SIZES</td>
</tr>
<tr>
<td><strong>Dataset</strong></td>
</tr>
<tr>
<td>Traffic</td>
</tr>
<tr>
<td>Brain</td>
</tr>
</tbody>
</table>
corresponding time slices is plotted in Figure 2(f). It clearly shows that as the difference increases, the number of visits to those time slices increases as well in both of the datasets.

B. Experimental Setup

For all experiments, we use the same setting of parameters for the Metropolis-Hasting algorithm as those in [15]. The number of accepted iterations is set to 10,000. In addition, to examine the effectiveness of the sampler, we set the discriminative threshold to 0 so that we can investigate all of the visited subgraphs, instead of only the best ones. The take-away is that, even with a threshold of 0, the subgraphs visited by the sampler should still be small and have high potential.

C. Accuracy

To verify the accuracy of our method, we perform experiments on a synthetic dataset due to the huge computational cost for finding the ground truth in a real world dynamic graph. More specifically, we select the structure of TRAFFIC as the underlying network. We generate a random dynamic pattern in time and space, and implant it into the network structure to create a synthetic balanced dataset such that the implanted pattern has an accuracy of 1. The rest of the node labels are set randomly. The dataset contains 400 samples of 20 time steps. To test the robustness of our method, we also introduced noise into the implanted patterns so that their accuracy is less than 1. Now, we check if DynaMinds can find the implanted dynamic pattern in both of these two cases: with and without noise.

Table II presents the results averaged over 100 runs with different sizes of the implanted patterns. In all of the experiments, DynaMinds succeeds in identifying a pattern with a discriminative potential of 1 that overlaps but may have different size from the implanted pattern. Interestingly, in the case when the implanted patterns have an accuracy of 1, the found patterns are always smaller than the implanted ones due to the human-mediated construction of the implanted patterns. On the other hand, when the implanted patterns do not have an accuracy of 1, the found patterns might be bigger than the implanted versions. Nevertheless, as the size of the implanted patterns increases, the size of the found patterns gets smaller compared to the implanted patterns. This phenomenon suggests that our method can find better discriminative dynamic patterns than the implanted ones both in terms of sizes and accuracy.

D. Effectiveness

As we use Metropolis-Hasting sampling algorithm to find patterns in the space of all possible dynamic subgraphs, it is expected that the sampler should visit mostly the subgraphs of interest, i.e., subgraphs of small size and high accuracy. This is clearly demonstrated in Figure 2(a)-2(d). As the size and length in time of the dynamic subgraphs increase, the average discriminative potential, i.e., the accuracy, also increases. Further, while the space of subgraphs is gigantic, the sampler focuses mostly on subgraphs of size around 5 for the Traffic network and 7 for the Brain network. The subgraphs found for Brain network are in general bigger than those for the Traffic network could be due to the more complex structure and the larger size of the Brain network.

An interesting question is whether the Metropolis-Hasting sampler is really better than a random sampler over the subgraph search space or not. To answer this question, we compare our method with a random sampler which visits subgraphs of random size and time length, but within the same ranges as those visited by DynaMinds. The results for both datasets are shown in Figure 2(e). Clearly, while DynaMinds mostly visits subgraphs with high discriminative potential, a random sampler does not care about the potential of subgraphs. Indeed, the random sampler visits more subgraphs of low quality, which is the opposite of what we want. The reason could be due to the huge space of possible subgraphs, within which only a small proportion of subgraphs has high discriminative potential.

| \( |V_I| \) | \( |V_G| \) | \( |V_N| \) | \( |V_G \cap V_I| \) | \( |V_N \cap V_I| \) | \( \phi(G) \) | \( \phi(N) \) |
|---|---|---|---|---|---|---|
| 6 | 2 | 10.4 | 0.33 | 0.16 | 1 | 1 |
| 9 | 3.8 | 10.9 | 0.31 | 0.22 | 1 | 1 |
| 12 | 3 | 11.1 | 0.25 | 0.22 | 1 | 1 |
| 15 | 2.95 | 9.4 | 0.20 | 0.25 | 1 | 1 |
| 18 | 2.95 | 9.55 | 0.16 | 0.25 | 1 | 1 |
| 21 | 3.05 | 10.1 | 0.15 | 0.24 | 1 | 1 |
Fig. 2. (a-b) Growth rate of discriminative potential with subgraph size and time length (c-d) Distribution of sampled subgraph size and time length. (e) Growth rate of visit count with discriminative potential.

E. Efficiency

The reason we resort to sampling instead of enumerating all dynamic subgraphs is to find the best ones is the huge space of subgraphs, especially in both space and time. Our algorithm can be run very fast. For 10,000 accepted iterations, it only takes 26 seconds for the Traffic network since the network is quite small with few edges. In addition, there is a clear difference in the distribution of active nodes between the positive and negative datasets, making it easier to find a subgraph of high discriminative potential. On the other hand, the Brain network is a full graph in each time stamp with similar distribution of edges in the positive and negative sets. Thus, it takes 35 minutes for 10,000 active iterations. Nevertheless, DynaMinds is still able to find subgraphs with very high discriminative potential, many of which have potential of 1.

VII. CONCLUSION AND FUTURE WORK

In this report, we propose a technique called DynaMinds to mine discriminative dynamic patterns which captures the evolutional rules of a time-evolving network. DynaMinds uses the memoryless principles of Metropolis-Hasting to efficiently and effectively move in the search space of all connected dynamic subgraphs and extract only the minimally discriminative dynamic subgraphs. Experimental results validate that DynaMinds meets the expected purpose of the mining task.

Future directions include:

1) Summarizing the output space: While DynaMinds is able to find minimally discriminative dynamic subgraphs, the number of such subgraphs is still huge, especially when there is too much noise in the dataset, leading to many false positives. Therefore, it would be beneficial to summarize the output space of DynaMinds and to present only some of the best and the most robust patterns to users. Such task depends largely on the availability of a substantial dataset, whereas, the datasets we have at the moment are quite small in the number of samples. A pattern might be considered robust if it does not occur too often in some background random dataset. Another possible extension is to summarize the output space by extracting only the frequent evolutional rules.

2) Finding other types of patterns: Our problem definition relies on the division of a dataset into positive and negative samples, which limits its usage significantly when such division is either hard to obtain or noisy. The technique we proposed, promisingly, can be applied to a more general case when only a single dataset is available. In such case, a new dynamic subgraph scoring function is needed, which can quantify the relevance of a subgraph given only the available dataset. A sampler can then explore the subgraph search space to maximize
this scoring function and find the best patterns. For example, in the case of the traffic network, we can find anomalous dynamic subgraphs indicating traffic jams by assigning weights in \(-1, 1\) to the nodes or edges. More specifically, a road (edge) will have a weight of 1 if the speed of the vehicles on it is significantly slower than normal, and -1 if the speed is close to normal allowed speed. Such weight assignment can be done by using the notion of p-value of speed. An anomalous pattern will then be a connected dynamic subgraph, i.e., a set of connected roads in time and space, that maximize the sum of the weights on the edges.

3) A better comparison with other dynamic pattern mining techniques: While the specific problem we are solving is basically new, there are similar tools that mine different types of dynamic subgraphs in the literature. A comprehensive comparison of our technique with these tools could be beneficial, especially given the fact that DynaMinds does not guarantee to mine all possible patterns, but only a subset of them.

4) Finding relative-time dynamic patterns: In this paper, we focus on absolute-time dynamic patterns, i.e., the patterns must start and end at the same points in time in all of the samples. This is a very strong assumption. In the future, we can investigate a more flexible and practical case where the time duration of the patterns varies between samples.

REFERENCES


