2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Recurrent Subgraph Prediction

Saurabh Nagrecha
Dept. of Computer Science and Engg., iCeNSA
University of Notre Dame,
Notre Dame, IN, USA.
email: snagrech@nd.edu

Nitesh V. Chawla
Dept. of Computer Science and Engg., iCeNSA
University of Notre Dame,
Notre Dame, IN, USA.
email: nchawla@nd.edu

Horst Bunke
Institute of Computer Science and Applied Mathematics,
University of Bern,
Neubruckstrasse 10,
CH-3012 Bern, Switzerland.
email: bunke@iam.unibe.ch

***Abstract*—Interactions in dynamic networks often transcend the dyadic barrier and emerge as subgraphs. The evolution of these subgraphs cannot be completely predicted using a pairwise link prediction analysis. We propose a novel solution to the problem—"Prediction of Recurrent Subgraphs (PReSub)" which treats subgraphs as individual entities in their own right. PReSub predicts re-occurring subgraphs using the network's vector space embedding and a set of "early warning subgraphs" which act as global and local descriptors of the subgraph's behavior. PReSub can be used as an out-of-the-box pipeline method with user-provided subgraphs or even to discover interesting subgraphs in an unsupervised manner. It can handle missing network information and is parallelizable. We show that PReSub outperforms traditional pairwise link prediction for a variety of evolving network datasets. The goal of this framework is to improve our understanding of subgraphs and provide an alternative representation in order to characterize their behavior.**

***Index Terms*—Network Science, Methods and Algorithms, Frequent Pattern Mining, Mining Graphs**

I. Introduction

In many cases—like modeling influence within social networks [1], inferring attacks on anonymized social networks [2], functional discovery in biological networks [3]—it is of interest to find whether certain repeatedly observed interaction patterns between multiple network elements will reoccur or not in a given future network. It could be as simple as a friend sending out texts to multiple people to have brunch every Saturday morning, or a complicated hierarchical interaction nexus within a corporation occurring before every quarterly meeting. It is important to predict the occurrence of such *key* subgraphs within a given time window in the future and not just predicting individual links.

Consider the network of all the characters in the French literary classic, Les Miserables[1]. The connections between characters can be summarized as per Figure 1. These characters and story arcs appear at various times in the story. Often, it is desired to query the fate of the Thenardiers as a family, or perhaps that of Marius' family. Each of these represent two subgraphs of interest in an evolving network.

We can see that their behavior is influenced by the context they are placed in, and thus global factors are seen to play a role in subgraph occurrence. Prototypical behavior in this network would be characterized by points in time where something distinct happens, for example– Jean Valjean's interaction with Myriel, the parts of the story concerning Fantine, the June Rebellion etc. These prototypical events, when considered individually, have varying amounts of similarity with the state of the network when the subgraph of interest is present or absent; but a combination of a number of these key events can describe what circumstances provoked the appearance of the Thenardiers or that of Marius' family. If one attempts to predict their interaction on a pair-wise basis, it can become cumbersome and inaccurate.

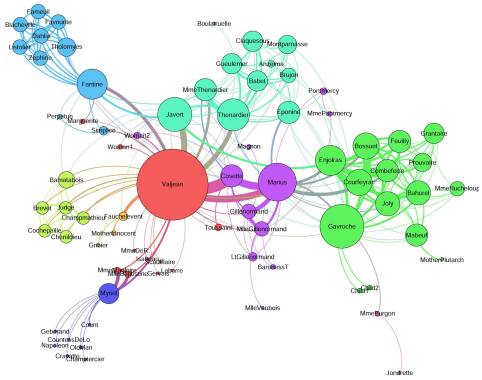

Fig. 1. **An example network**: The interaction between the characters of Les Miserables

The underlying idea in all the above examples is that there exist predictable patterns in networks. We would like to be able to identify *what* these patterns are, *when* they occur, and if we could find effective "early warning" methods to predict them. In this paper, we show that these interaction patterns are such that they cannot be predicted by mere component-wise analysis, and thus are an emergent phenomenon.

Traditionally, functional significance has also been attributed to individual links. But, in order to fully characterize them, studying just the dyadic interactions is insufficient—this motivates us to look at richer graph elements, such as subgraphs. Predicting the occurrence of subgraphs on a link-wise basis has the inherent disadvantage of scaling combinatorially, and as a result, its accuracy degrades correspondingly for an increasing number of constituent nodes (as we shall show later in this paper).

[1] V. Hugo. Les Miserables (abridged). 1862

ASONAM '15, August 25-28, 2015, Paris, France

ACM ISBN 978-1-4503-3854-7/15/08 $15.00
DOI: http://dx.doi.org/10.1145/2808797.2809283

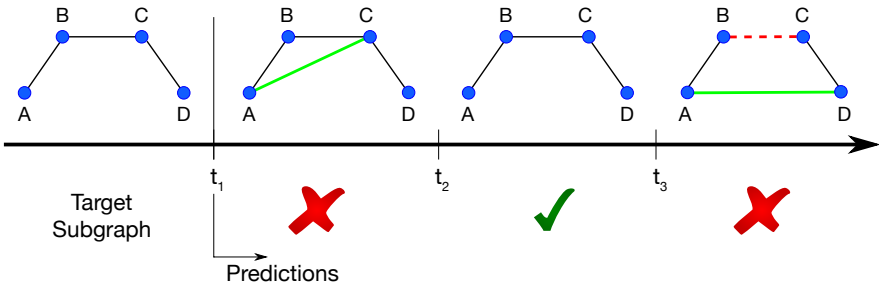

Fig. 2. **Predicting Recurring Subgraphs**: We predict when the target subgraph will appear in the network. Any deviations from the requisite target subgraph are considered as non-occurrence of that subgraph

Even if individual link prediction tasks are highly accurate, combining their results over multiple edges results in poor performance. This motivates us to think that subgraphs might be emergent structures in their own right and not just a composition of links. We propose our framework Prediction of Recurrent Subgraphs or "PReSub", which uses graph edit distances [4] of the network in order to predict the subgraph's occurrence.

Since the method we propose relies on inexact dissimilarity measures, we can use estimates of unknown network quantities and still be able to predict the occurrence of the target subgraph. This could be useful for networks where one is oblivious to certain regions of the network for some time period– like in the case of adversarial or terrorist networks. This predictive power is derived from using the Graph Edit Distance as a *contextual* feature of the graph state.

Problem Statement and Proposed Framework

Formally, given a labeled temporal network G, we are interested in predicting *if* a specific target subgraph S, which we have previously observed, will occur again in a given time-frame. This is illustrated in Figure 2. There is no additional restriction on the nature of the subgraph. We can either use a pre-specified subgraph, or infer recurring subgraphs using unsupervised methods. In this paper, we consider aggregated contact sequence networks [5], where a discrete set of non-overlapping time windows, within which a network instance or a "snapshot", G_t, is captured. Despite this, one can easily transform other temporal models to obtain equivalent predictions, like shifting time windows, and interval graphs.

A subgraph is considered "recurrent" when it is seen to appear and subsequently disappear in the training data. This idea is visited in detail in Subsection IV-A.

In PReSub, we consider the occurrence of the subgraph as an individual ground truth. The supervised observed instances in this case are just '1' for the subgraph being observed and '0' for it not being observed. We only report that the subgraph has been observed if all of the constituent links are present and *no other* links are present between the vertices of the subgraph. This is not the same as a link prediction problem, which is concerned with predicting links between two nodes. However, one may posit the problem of subgraph prediction as a prediction of links emanating from different nodes in a given subgraph. To that end, we compare PReSub with link prediction algorithms, as a baseline, which makes it a multilabel prediction task where each of the labels ('1' for presence of that link and '0' for absence) must be predicted correctly for the entire subgraph to be predicted correctly (Subsection V-B1). In this paper, we restrict the treatment to *exact* subgraph matching, but the above formalism can easily be modified to include inexact subgraph matching.

The "dataframe" in this paper is a table containing each time instance of the network as a row. Each row contains descriptive feature information about that particular instance and the instance's class value(s). As per the above descriptions, the class values in the baseline link prediction case are given as '1' or '0' for each constituent link's presence or absence, respectively. In the case of PReSub, it is simply '1' or '0' for the entire subgraph's presence or absence, respectively.

In PReSub, in order to get feature values of the network instance, we need a vector which describes what is happening at that particular time in the network. To that end, we leverage Graph Edit Distances (GED) for a vector space embedding, i.e. they encode a high dimensional object (here, the network instance) into a vector. Each value here represents the dissimilarity score from a reference network state, called a "prototype". In a more detailed form of PReSub, we consider "early warning subgraphs" (henceforth abbreviated as "EWS") as local indicators of structural behavior. EWS serve as pre-emptive structural indicators of the subgraph's recurrence in the future. When an EWS is seen, this value is '1', otherwise, it is '0'. In the dataframe, all these serve as feature values.

We investigate how EWS can be used in order to enhance our prediction. We follow the occurrences of subgraph structures which would eventually lead to the target subgraph. Aided by historical knowledge of their occurrence, we can

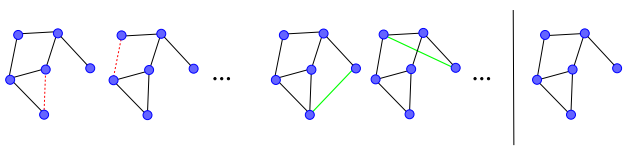

Fig. 3. **Potential Early Warning Subgraphs**. We can use the historical occurrences of subgraphs on the right to predict when the subgraph on the left may occur.

achieve better prediction performance for the target subgraphs. Prediction using these EWS is motivated by the concept of triadic closure; we aim to look at possible indicators of a given interaction pattern by looking at possible patterns which lead up to it. Even if information about parts of the network is unavailable for a certain time instance, the use of inexact graph matching enables us to get an approximation to the GED vector and hence a prediction for the desired subgraph's presence.

In the spirit of finding a minimal feature-set to describe the network state, in Subsection VI-B we show how varying the number of prototypes could lead to overfitting and would also defeat the purpose of dimensionality reduction. PReSub outputs the time range of the subgraph's occurrence, hence it is of interest to see how this granularity of prediction window can affect the prediction performance. As we show in Subsection VI-E, PReSub is parallelizable at every stage, and is thus ideal for very large graphs.

II. Related Work

To the best of our knowledge, subgraph prediction, as per our description in our Problem Statement, is an unexplored area.

Generating candidates for recurrent subgraphs can be achieved using techniques found in frequent subgraph mining. These techniques can be either transaction-based, or occurrence based [6]. Out of these, transaction based methods are of relevance to the problem of discovery within evolving labeled aggregated contact sequence networks. FP-Growth, a very basic association rule based method of frequent pattern generation is used in this paper [7]. It does not use expensive structures to grow candidate sets, but instead populates a scalable tree in order to generate suitable candidates, even under constraints. An efficient, parallel implementation of this is discussed in Parallel-FP-Growth [8].

SUBDUE [9], is a popular, inexact method which uses the minimum descriptor length principle to compress graph data, but it generates a limited set of patterns and it does not scale well; hence it is not considered suitable for this paper.

To quantify how much a network differs from another, we need a suitable dissimilarity measure. In PReSub, we use GEDs to achieve this. We use them for their versatility [4], error tolerance [10] and dimensionality reduction capability [11]. The problem of graph matching is NP hard, but in our case, we use labeled graphs and there are various efficient implementations available to obtain inexact graph matching [12]–[14]. Even though an inexact method of graph matching is used, PReSub predicts *exact* subgraph occurrences. These dissimilarity measures need to be taken from certain reference graph instances, for which we need to identify prototypical states of the network. Prototype selection in general has been introduced in [15], and [16] shows various techniques to use it for graphs and how to obtain a vector space embedding of the graph state. This is further discussed in Section IV and the implementation for the scope of this paper is as per Section V.

Graph embedding has been used in [17], but the scope of the embedding has been limited to the subgraph itself; whereas, our method embeds the entire network state. Structure prediction in temporal networks has been studied in [18], but it considers interactions which occur across time-steps; our approach aims to predict any arbitrarily defined subgraph for a future instance. Subgraph frequency mapping in [19] uses constrained size, edge-induced subgraphs– PReSub imposes no constraint on size and predicts exact subgraphs.

III. Data

We investigate a broad spectrum of real world networks which evolve over time in order to observe semantically meaningful recurrent subgraphs.

1) Commercial cellular phone calls (`mobile`): We use a stream of 712 million cellular phone calls from a major non-American phone service provider [20]. This is a directed network where an edge (e_{ij}) represents customer v_i calling v_j. We break this down into intervals of 30 minutes, consisting of average call volumes of the order of 100k calls per snapshot.
2) Wikipedia Co-authorship (`wiki`): Submissions and edits to the online collaborative encyclopedia, Wikipedia by its contributor community. The raw data is in the form of a bipartite graph obtained from http://dumps.wikimedia.org/. This bipartite graph contains information on the article and the user who contributed to it, with timestamps of each event. We collapse this into a massive co-authorship graph, in discrete time steps.
3) Enron Email Corpus (`enron`): The Enron email corpus is a large scale email collection from the organization bearing the same name. This paper uses the version of the [21]. Here, a directed edge e_{ij} corresponds to an email sent by v_i to v_j.
4) Facebook Wall Posts (`facebook`): The online social network data from Facebook for users in the New Orleans regional network was crawled by the authors of [22]. This paper uses the interactions between these users that manifested in the form of "wall posts". The relevant features of the network's stream contain information about the wall owner, the user who made the post, and a POSIX time stamp of the time when the post was made.

TABLE I
A summary of the data sources used

Dataset	# of Nodes	# of Edges	Time Span
mobile [20]	8,321,119	712M	65 days
wiki	25,323,882	266M	∼4 years
enron [21]	87,098	1,147,028	∼4 years
facebook [22]	46,715	803,744	∼2 years

In these sources, we have divided the data into equal-width discrete time windows which give a "snapshot" of the state of the network in that duration.

IV. METHODS

PReSub addresses the problem of subgraph prediction as per the modules defined in Subsection I. A preliminary step is to identify a subgraph of interest. If such a subgraph is already provided, then we can safely ignore this step. To build a contextual "story" around the evolution of the network over time, we identify several key prototypical graph instances using unsupervised methods [4]. We obtain the feature values using the techniques mentioned below and populate the dataframe. Each instance is labeled by occurrence or non-occurrence of the target subgraph. Using this calculated dataframe, we train on past data and predict whether a subgraph will reappear or not in the given test instance.

A. From "Frequent" Subgraphs to "Recurrent" Subgraphs

Frequent subgraph mining aims to discover subgraphs in a given dataset, whose occurrence exceeds a specified threshold [6]. We use FP Growth [7] in order to identify frequently occurring subgraphs.

It is to be noted that the most frequent links in the network need not be the most informative. These may as well be characterized as "background noise". We posit that the key to considering a particular link as background noise is not in the *occurrence* frequency, but in the *recurrence* frequency.

In order to obtain interesting subgraphs in an unsupervised manner, we propose a transformation of the frequent subgraph prediction problem. Instead of counting each item's occurrence, we count the item's "re-occurrence". We do this with the below example in mind:

Say we observe the network over various edge-list snapshots as below:

$$G_1 = \{l_1, l_2, l_3\} \qquad G_2 = \{l_2, l_3, l_4\}$$
$$G_3 = \{l_1, l_2, l_4\} \qquad G_4 = \{l_1, l_2, l_3\}$$

We see that l_1 disappears in time instance 2 and reappears in time instance 3. Similarly, l_3 disappears in instance 3 and reappears in instance 4. Since these re-occurrences are what interest us, we transform the itemsets to only contain those items which are reoccurring after deletion – i.e. convert the network snapshots to display only the links which are reoccurring after prior deletion. So, for the above toy example, we get

$$G'_1 = \{\} \qquad G'_2 = \{\}$$
$$G'_3 = \{l_1\} \qquad G'_4 = \{l_3\}$$

Notice that we have itemsets which are now smaller and more relevant to the problem of recurrence. We can now perform association rule mining on the links that we have now obtained. The most frequent itemsets that we get as a solution of this problem are really just the most "recurrent" subgraphs.

A subgraph's *recurrence frequency* is the recurrence frequency of the *least* frequent link in it. By this argument, from the distribution of link recurrence thus obtained in Figure 4, we extract three subgraphs with varying amounts of recurrence to demonstrate the power of PReSub. As we can see, very few links, and as a result subgraphs, reoccur frequently.

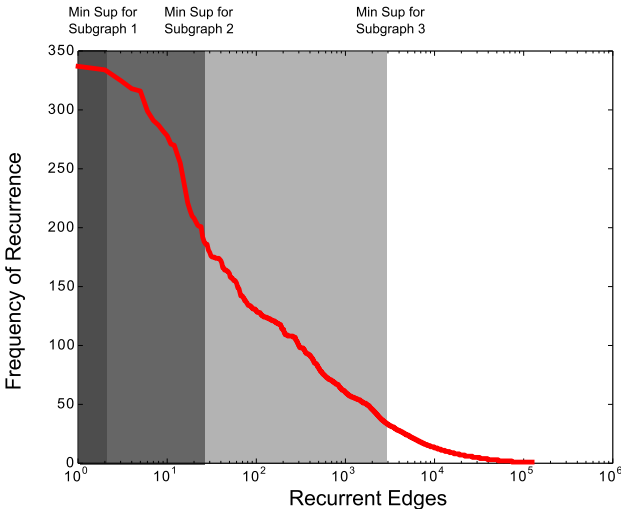

Fig. 4. **Distribution displaying the recurrence frequency of links**. The minimum support of a constituent link in a subgraph dictates the entire subgraph's recurrence frequency. We extract three such subgraphs from extreme values of minimum support. Since these subgraphs are diverse in their recurrence frequency, we see how our framework performs in terms of predicting their recurrence in Subsection VI-C.

B. Graph Edit Distance and Prototypes

In order to obtain a reduced dimensional picture of the graph, we use a graph kernel to implement vector space embedding [4]. Borrowing from the field of pattern recognition, Graph Edit Distances (GED) [10] from a selection of prototype graphs to the graph's state are used in order to characterize each graph's state in time. The GEDs thus obtained are then used as features to predict subgraph occurrence.

C. Graph Dissimilarity Measures for Embedding

To achieve a representation of the changing graph structure, in the domain of graphs, the notion of proximity is more fundamental than that of a feature or a class [15]. When dealing with structural data, like graphs, it is difficult and unmanageable to extract numerical features, whereas proximity measures can be evaluated from the data using dissimilarity measures. GED is a flexible, error tolerant mechanism to measure the dissimilarity between two graphs. Here we use it in the context of labeled directed graphs.

We consider a labeled set of sample graphs $T = \{g_1, ..., g_n\}$, and a graph dissimilarity measure $d : G \times G \to \mathbb{R}$. In this problem, we use GED as the dissimilarity measure as motivated by the previous section. From the set of n labeled graphs, we select a set $P = \{p_1, ...p_m\}$ of $m \leq n$ prototype graphs. This leads to m dissimilarities of the form $d_i = d(g, p_i)$, which can be arranged in an m-dimensional vector $[d_1, ..., d_m]$. So, in this way, an entire graph can be transformed into a vector of real numbers.

Calculating GED requires us to find the minimal common subgraph (MCS) between two networks. Finding an MCS is a known NP-hard problem; but in our case, we can make some simplifications based on the network having non-empty ($V \neq \phi$) instances with statically-labeled nodes and edges. Since

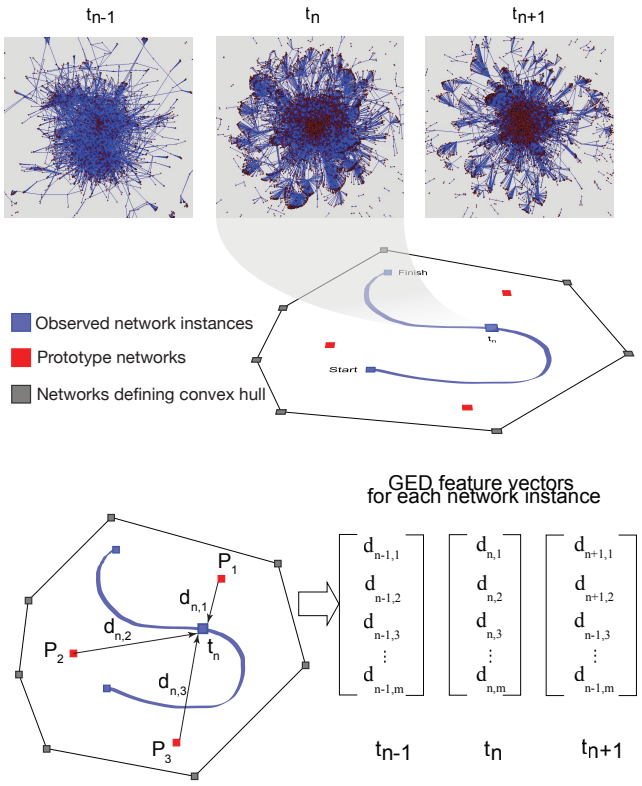

Fig. 5. **A pictorial summary of the vector space embedding**. Networks are extracted at consecutive time windows. The training set defines the convex hull of all possible graphs that the system can predict. "m" prototype graphs (denoted as P_i) are selected from within this space. At each instance, $t = t_n$, Graph Edit Distances (GEDs) of each of these networks from the set of prototype graphs are calculated and considered as features in the vector space embedding. The states represented by the GEDs can be used to predict the next state of the subgraph.

the set of nodes and node labels does not change, we have a relatively simplified problem.

We could also use a Levenshtein distance based implementation of finding GED [14]. This treats the edit distance as an equivalent problem to finding the dissimilarity between pairwise strings. This would be an inexact measure, but it operates in near-linear time and would thus be computationally favorable.

D. Prototype selection

For the purposes of PReSub, we use Border, Random or Targetsphere prototype selection.

Intuitively one can imagine the measures obtained against these prototypes to be "location triangulations" of the graph's (dynamic) state in the space of all graphs $\mathcal{G}$. The space of all graphs is the space defined in the $\mathbb{R}^{|V|\times|E|}$ domain which encompasses *all* the possible states the graph can attain. The prototypes are graphs within this space, against which the distance of the graph's state is measured so as to ascertain its state in the domain. A simple two dimensional analogue of this is as depicted in Figure 5.

V. Predicting Subgraphs on Real World Networks

We demonstrate the effectiveness of PReSub on a variety of real world social networks as described in Section III. These networks are diverse in terms of origin, task size[2], and temporal activity.

A. Features Used

a) PReSub: We see that GEDs are vector space embeddings of the entire network's state, which means they can provide global contextual information about the network's state. The specific components of the GEDs attempt to provide a minimum descriptor of the particular graph's state using distances from the prototypes. Each of these GEDs is used as a feature under PReSub.

b) PReSub + Early Warning Subgraphs: In addition to the above "vanilla" version, we can use the occurrence of "early warning" subgraphs (EWS) in order to predict that of the target subgraph. In the same spirit as that of "triadic closure", we suggest that we use the historic occurrences of those recurrent subgraphs which are a few edges short of the target structure. Here, we consider those subgraphs which differ from the original subgraph by one edge as candidates for being EWS. Whenever the target subgraph is present, the EWS' will be considered as 'present', since they are edge induced subgraphs of the target subgraph. For the instances where it is not present, these EWS' indicate a build-up leading to the appearance of the target subgraph. This is displayed in Figure 3. We use the historic occurrence of these EWS' in conjunction with the GEDs as the feature set.

B. Prediction Evaluation

The task is to predict whether a target subgraph occurs in a time-frame. We have three comparative approaches – baseline link prediction, PReSub, and PReSub+EWS.

1) Using Baseline Link Prediction: For the baseline approach, we use conventional link prediction methods with the entire target subgraph $S = (V_S, E_S)$ in mind. Consider the clique $G_{cl} = (V_S, E_{cl})$ formed by all of the vertices in S, this contains two types of links– those which occur in S, and those which do not. If we are to predict the occurrence of S, we have to predict not only the occurrence of E_S, but also the non-occurrence of the links in $E'_S = E_{cl} - E_S$. For this, we have a ground truth observation for each individual link. If any one link is misclassified, then the prediction is deemed to have been misclassified.

At each time instance, we perform a link prediction on each of the constituent links in E_S and E'_S. We now have the scores pertaining to both of these sub-problems over all the observed examples, henceforth referred to as *training instances*. For each of them, we only consider the subgraph to be predicted to occur if and only if all the links in E_S are present and those in E'_S are absent. In a correctly predicted case, we would expect

[2]Here, the *task size* of the subgraph refers to the number of link prediction tasks associated with it. This is equal to the number of edges in the clique formed by the constituent nodes of the subgraph.

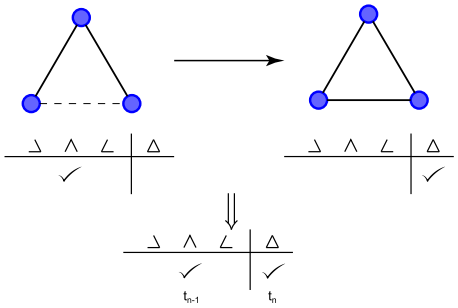

Fig. 6. **Triadic Closure as a special case**. Triadic closure could be explained using early warning subgraphs as features.

all the links in E_S to be predicted as present and those in E'_S as absent.

We use the comprehensive solution LPmade [23] in order to perform all the baseline experiments required for this paper. The LPmade paper also contains detailed information on particular link prediction techniques.

2) PReSub: We train on the initial instances in order to infer the prototypes. Each of the training instances contributes to defining the convex hull, from within which the prototypes are selected. Prototype selection is done using any of the various methods listed in Subsection IV-D. Using these prototypes, the GEDs to all the graphs at various time instances are evaluated. It is to be noted that GEDs to all the graphs within the training data all confirm to locations within the convex hull, by definition; but the test instances are not subject to this constraint. We treat these GEDs from the training data in conjunction with the historical subgraph information as supervised instances of time series data.

Along the lines of High Performance Link Prediction (HPLP) [20], we use random forests as our classification technique. It can be effectively parallelized and mitigates the problem of feature selection upfront.

C. PReSub + EWS

Leaning on the triadic closure theory, we choose our EWS as those which occur in a 'V'-shaped configuration between the three constituent nodes. Thus, our EWS features are the occurrences of these V-shaped subgraphs.

VI. Results

We report our results using ROC curves and AUROC. In addition to the breadth of datasets we cover, we explore the effects of varying the task size of the subgraph, prototype selection methods, and the number of prototypes used.

A. AUROC Performance

We see that when pitted against the best performing conventional link prediction method, PReSub shows a marked improvement overall which is reflected in the AUROCs. The addition of Early Warning Subgraphs makes the performance even better, since we now have a richer blend of local (EWS) and global (GEDs) features to learn on.

Our method, which relies on GEDs, is relatively immune to the effects of such degradation. As can be seen in comparison with the conventional link prediction method in Figure 7, it confirms that the degradation of performance for link prediction methods is markedly significant. This is due to the fact that as the subgraph increases in task size, the performance for the baseline pair-wise prediction plummets. Since the task size increases as $O({N_{nodes}}^2)$, where N_{nodes} is the number of nodes in the target subgraph, this is to be expected.

If our target subgraph is a triad, like in the case of the `enron` network, one can use various EWS in order to predict its occurrence. We use PReSub, equipped with EWS' which are these particular V-shaped subgraphs. These subgraphs should be able to predict when the triangle between the three nodes should close. We obtain an AUROC of 0.880 using PReSub, whereas PReSub+EWS results in an AUROC of 0.943 for the same subgraphs.

B. Choice of Number of Prototypes

Using a range of prototypes, we plot the AUROC performance obtained in the case of PReSub and PReSub+EWS with the goal of finding how the number of prototypes affects prediction performance and what is the number of prototypes one should consider. The latter is important to evaluate since our entire theme is motivated by finding a minimum set of GED features which best describes the network state. The "best" feature set to describe the network state is the one which gives us the best AUROC performance, since that is our end-goal.

From Figure 8 we see that for PReSub, the performance gives diminishing returns upon increasing the number of prototypes beyond 10 until a prohibitively high number of prototypes is used (in this case, 70). If one were to use 70 graphs out of 100 instances as prototype graphs, it would defeat the purpose of exploring subgraph evolution using just an initial prototype discovery phase. From the same figure, we also see that PReSub+EWS gives us the best performance

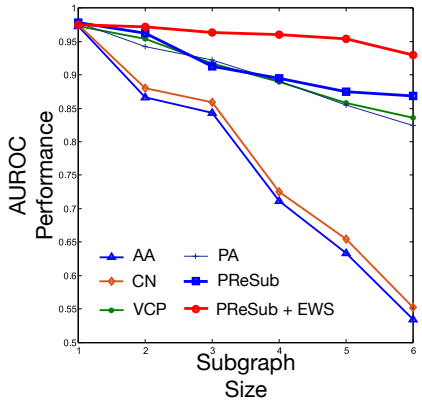

Fig. 7. **Performance of the proposed methods versus baseline techniques**. Only a representative set of baseline techniques has been displayed for clarity. The techniques are as per Tables II and III. Notably, VCP [24] comes closest to our performance.

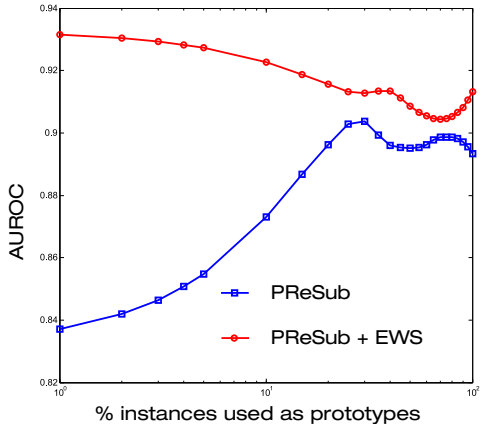

Fig. 8. **Performance as a function of the Number of Prototypes** in the Enron network. We would like to optimize the number of prototypes such that we get the best performance in terms of AUROC. We see that it is advantageous to use PReSub+EWS with a few prototypes.

using just a few prototypes (in this case, 4). If a greater fraction of the instances is used as prototypes, then the performance gives diminishing returns and then proceeds to perform almost as good as PReSub. From this example, it goes to show that the best strategy here is to use PReSub+EWS with a lower number of prototypes.

We clearly see that there exists a certain optimal number of prototypes one must consider to get the best performance. This number could be obtained by running a parameter sweep for PReSub over a small set of pilot instances. In case concept drift is detected concurrently, this parameter sweep may not hold and must be re-evaluated.

C. Choice of Subgraphs to be predicted

In order to show the effectiveness of PReSub in predicting *any* given subgraph, we select subgraphs with varying degrees of recurrence from a given network as shown in Figure 4. The subgraphs S_i thus chosen, are of equal task size so as to not compound the effect of link prediction degradation as observed in Figure 7, i.e. if the subgraph task size is kept variable along with the recurrence frequency, we cannot isolate the effects of the individual factors. These subgraphs have recurrence frequencies as listed below:

- S_1: greater than 95%ile
- S_2: greater than 50%ile
- S_3: greater than 25%ile

Thus, we see that the subgraphs all have a varied range of recurrence frequency and as a direct result– a correspondingly varied range of class imbalance. From Table II, *we can see that the choice of subgraph has no noticeable effect on the performance of PReSub, whereas link prediction methods are more sensitive to changes in class distribution.*

D. Effect of Granularity of timestamps

Timestamps contain a real number value of the time that each edge was recorded in the networks. In order to obtain a snapshot of the network, discrete windows of this real valued interval are created and links which exist within a given time window are said to have occurred during that particular discrete timestep. This discretization has a lower bound established in accordance with the Nyquist-Shannon Sampling Theorem. We investigate this for various time binning choices for the `enron` dataset as shown in Figure 9 and display the results in Table III. If a very broad window is chosen, the fine-grain aspects of recurrence may be lost, and if a very narrow window is chosen, there may be very little difference between two consecutive snapshots.

TABLE II
CHOOSING SUBGRAPHS WITH DIFFERENT RECURRENCE MODALITIES: AUROC PERFORMANCE FOR VARIOUS SUBGRAPHS IN THE ENRON NETWORK. ALL SUBGRAPHS S_i ARE OF THE SAME TASK SIZE, BUT HAVE DIVERSE RECURRENCE FREQUENCIES.

	Subgraphs		
Technique	S1	S2	S3
Adamic Adar (AA)	0.3803	0.3968	0.3656
Common Neighbor (CN)	0.3764	0.3885	0.3480
Preferential Attachment (PA)	0.3934	0.4175	0.3703
VCP	0.8021	0.8253	0.7889
PReSub	0.8408	0.8514	0.8349
PReSub+EWS	0.9017	0.9094	0.8653

TABLE III
GRANULARITY OF TIMESTAMPS: AUROC PERFORMANCE FOR VARIOUS DISCRETIZATION SCHEMES FOR TIMESTAMPS IN THE ENRON NETWORK. A HIGHER NUMBER OF DISCRETIZATION BINS CORRESPONDS TO A NARROWER WINDOW FOR SNAPSHOTS OF THE NETWORK.

	# Discretization bins		
Technique	100	1000	10000
Adamic Adar (AA)	0.3803	0.4173	0.2556
Common Neighbour (CN)	0.3764	0.3987	0.2602
Preferential Attachment (PA)	0.3934	0.4199	0.2865
VCP	0.8021	0.8202	0.6348
PReSub	0.8408	0.9183	0.8210
PReSub+EWS	0.9017	0.9276	0.8472

E. Computational Complexity

We analyze the complexity of the problem by breaking it down into its constituent sub-problems. This is broken down as per Table IV. If the target subgraph is known beforehand, the most computationally intensive task is that of evaluating graph dissimilarity using edit distances; however each of these steps are amenable to parallelization. In comparison, for the baseline pair-wise link prediction, the computational time complexity

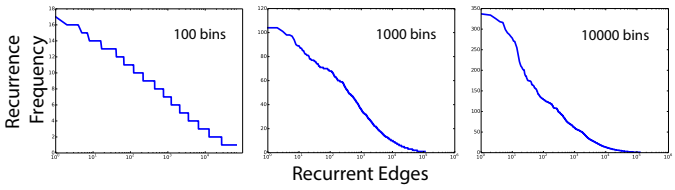

Fig. 9. **The effect of choosing discrete timesteps**, illustrated with various values of the number of bins. Timestamps can be discretized with varying granularity, which is reflected in the recurrence of edges.

TABLE IV
THE COMPUTATIONAL COMPLEXITY OF THE PROPOSED FRAMEWORK COMPARED TO THE BASELINE TASK OF LINK PREDICTION. THE BOTTLENECK STEPS FOR EACH ARE HIGHLIGHTED IN BLUE. THIS SHOWS THAT THE PROPOSED FRAMEWORK IS PARALLELIZABLE AT EVERY STEP.

Approach	Component	Time Complexity ($O(...)$)	Notes
Proposed	Recurrent Subgraph Discovery	$\|G_{train}\|/P$ $\|I\| \times max(\#itemRelatedPatterns)$ $\times log(K)/P$	Map and reduce steps for FP-Growth as per [8] P = # of parallel computation nodes
	Prototype Selection	trivial	if using Random Prototype Selection
	GED computation	$n_{test} \times \#Prots \times \|E\|^{1+\epsilon}/P$	$E = E_t \cup E_{prot}$
	Prediction	$N_{trees} \times \mathcal{F} \times n_{train} log(n_{train})/P$ $LPMethod$	Using N_{trees} in a Random Forest classifier In general, if one does not use Random Forests
Baseline Link prediction		${N_{nodes}}^2 \times LPMethod$	$\binom{N_{nodes}}{2}$ reduces to N_{nodes}^2

is simply that of recurrent subgraph discovery plus that of the prediction task (i.e. ${N_{nodes}}^2 \times O(LPMethod)$, where N_{nodes} is the number of nodes in the target subgraph and $LPMethod$ is the link prediction technique). Note that the ${N_{nodes}}^2$ term comes from there being $\binom{N_{nodes}}{2}$ links need to be predicted.

VII. CONCLUSION

We demonstrate that the behavior of subgraphs cannot be fully modeled using just their constituent components; instead, we provide a comprehensive analysis into how one could address this challenge. In PReSub, we find prototypical behaviors from the training set. We then express graph instances as a combination of these prototypical states. Based on these global trends, we learn how the subgraph reacts to the evolving network. From our training data, we wish to find how similarity (or dissimilarity) to a particular prototype affect the presence or absence of the subgraph we are interested in.

Using a variety of distinct prototypes enables us to characterize such behavior in a more fine-tuned manner, but increasing the number of prototypes to an arbitrarily large amount defeats the purpose of the dimensionality reduction that we aim to achieve.

ACKNOWLEDGMENTS

Research was supported in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, National Science Foundation (NSF) Grant OCI-1029584, and by the grant FA9550-12-1-0405 from the U.S. Air Force Office of Scientific Research (AFOSR) and the Defense Advanced Research Projects Agency (DARPA).

REFERENCES

[1] J. Leskovec, A. Singh, and J. Kleinberg, "Patterns of Influence in a Recommendation Network," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2006, pp. 380–389.

[2] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 181–190.

[3] H. Hu, X. Yan, Y. Huang, J. Han, and X. J. Zhou, "Mining Coherent Dense Subgraphs across Massive Biological Networks for Functional Discovery," *Bioinformatics*, vol. 21, no. suppl 1, pp. i213–i221, 2005.

[4] K. Riesen and H. Bunke, "Graph Classification Based on Vector Space Embedding," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 06, pp. 1053–1081, 2009.

[5] P. Holme and J. Saramäki, "Temporal Networks," *Physics reports*, vol. 519, no. 3, pp. 97–125, 2012.

[6] C. Jiang, F. Coenen, and M. Zito, "A Survey of Frequent Subgraph Mining Algorithms," *KER*, vol. 28, no. 1, pp. 75–105, 2013.

[7] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," in *ACM SIGMOD Record*, vol. 29, no. 2, 2000, pp. 1–12.

[8] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "PFP: Parallel FP-Growth for Query Recommendation," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 107–114.

[9] D. J. Cook and L. B. Holder, "Substructure Discovery Using Minimum Description Length and Background Knowledge," *arXiv cs/9402102*, 1994.

[10] H. Bunke and K. Shearer, "A Graph Distance Metric based on the Maximal Common Subgraph," *Pattern recognition letters*, vol. 19, no. 3, pp. 255–259, 1998.

[11] K. Riesen and H. Bunke, "Reducing the Dimensionality of Dissimilarity Space Embedding Graph Kernels," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 1, pp. 48–56, 2009.

[12] M. Neuhaus, K. Riesen, and H. Bunke, "Fast Suboptimal Algorithms for the Computation of Graph Edit Distance," in *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2006, pp. 163–172.

[13] S. Fankhauser, K. Riesen, and H. Bunke, "Speeding Up Graph Edit Distance Computation through Fast Bipartite Matching," in *Graph-based representations in pattern recognition*. Springer, 2011, pp. 102–111.

[14] A. Andoni and K. Onak, "Approximating Edit Distance in Near-Linear Time," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1635–1648, 2012.

[15] E. Pekalska, R. Duin, and P. Paclik, "Prototype Selection for Dissimilarity-based Classifiers," *Pattern Recognition*, vol. 39, no. 2, pp. 189–208, 2006.

[16] K. Riesen, M. Neuhaus, and H. Bunke, "Graph Embedding in Vector Spaces by means of Prototype Selection," in *Graph-Based Representations in Pattern Recognition*. Springer, 2007, pp. 383–393.

[17] V. Vacic, L. M. Iakoucheva, S. Lonardi, and P. Radivojac, "Graphlet Kernels for Prediction of Functional Residues in Protein Structures," *Journal of Computational Biology*, vol. 17, no. 1, pp. 55–72, 2010.

[18] M. Lahiri and T. Y. Berger-Wolf, "Structure Prediction in Temporal Networks Using Frequent Subgraphs," in *IEEE Symposium on CIDM*, 2007, pp. 35–42.

[19] J. Ugander, L. Backstrom, and J. Kleinberg, "Subgraph Frequencies: Mapping the Empirical and Extremal Geography of Large Graph Collections," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 1307–1318.

[20] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New Perspectives and Methods in Link Prediction," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 243–252.

[21] J. Shetty and J. Adibi, "The Enron Email Dataset Database Schema and Brief Statistical Report," *Information Sciences Institute Technical Report, University of Southern California*, vol. 4, 2004.

[22] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the Evolution of User Interaction in Facebook," in *Proceedings of the 2nd ACM workshop on Online social networks*, 2009, pp. 37–42.

[23] R. N. Lichtenwalter and N. V. Chawla, "LPmade: Link Prediction Made Easy," *Journal of Machine Learning Research*, vol. 12, pp. 2489–2492, 2011.

[24] R. N. Lichtenwalter and N. V. Chawla, "Vertex Collocation Profiles: Subgraph Counting for Link Analysis and Prediction," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 1019–1028.