Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)

Structure Prediction in Temporal Networks using Frequent Subgraphs

Mayank Lahiri
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: mlahiri@cs.uic.edu

Tanya Y. Berger-Wolf
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: tanyabw@cs.uic.edu

***Abstract*— There are several types of processes which can be modeled explicitly by recording the interactions between a set of actors over time. In such applications, a common objective is, given a series of observations, to predict exactly when certain interactions will occur in the future. We propose a representation for this type of temporal data and a generic, streaming, adaptive algorithm to predict the pattern of interactions at any arbitrary point in the future. We test our algorithm on predicting patterns in e-mail logs, correlations between stock closing prices, and social grouping in herds of Plains zebras. Our algorithm averages over 85% accuracy in predicting a set of interactions at any unseen timestep. To the best of our knowledge, this is the first algorithm that predicts interactions at the finest possible time grain.**

I. Introduction

In many applications, a common objective is to model the dynamic behavior of a process and to predict its state in the future. Several types of processes are represented by a set of entities interacting over a period of time. Examples which have been studied extensively are the evolution of bibliographic databases [1]–[5], the web and other information networks [6]–[13], disease transmission paths in epidemiological simulations [14]–[19], influence and information spread in a social network [20]–[27] and the pattern of correlations between stock prices [28], [29]. Temporal networks[1] are a powerful generic model for such processes [32]. A temporal network consists of a sequence of graphs, each being a snapshot of interactions at a particular instant or over a small time interval. Vertices in each graph represent entities and edges between them represent interactions, which can either be directed or undirected (bi-directional). The flexibility of this definition allows temporal networks to be used to model a variety of processes while maintaining the explicit order and concurrency of interactions.

There are many questions that can be posed for processes represented as temporal networks. We focus on the task of predicting the structure of the temporal network at *each unseen timestep* by computing a model of the evolution of the process under consideration. A closely related problem from the field of network analysis is that of *link prediction* in a graph, which aims to rank all possible edges (interactions) by the likelihood that they will occur at some point in the future [4]. Our work extends that definition to temporal networks by predicting for each timestep the structure of the graph which is not noise. The predictions are based solely on prior observations. Unlike the link prediction problem, we are concerned with the ability to predict exactly *when* groups of interactions will occur, not to predict the likelihood of every possible interaction occurring at *any* point in the future.

In this paper, we present a generic, accurate, adaptive, streaming algorithm for efficient structure prediction in temporal networks. Our algorithm does not rely on any domain-specific features. The data is assumed to be a stream of graphs, and the algorithm adaptively learns a model for the dynamics of the process. Our algorithm uses the idea that certain interactions signal the occurrence of others at some point in the future. By probabilistically measuring the delay between interactions, one can predict exactly when certain interactions are going to occur based on past and current observations. However, directly using this approach requires computing on pairs of edges and becomes intractable for even medium-sized graphs. We propose the use of frequent subgraphs in order to aid the tractability of the algorithm as well as to filter out insignificant interactions.

We tested our algorithm on three diverse examples of real-world processes, ranging from stock price correlations to social groupings in animal populations. Our algorithm averages 85.7% accuracy when predictions are allowed a single slack timestep. This is the first algorithm, to the best of our knowledge, that predicts interactions between entities at the finest possible time grain.

This paper is organized as follows. In the rest of this section, we review relevant prior work from computer science as well as other fields and list some preliminary definitions. In Section II, we formally define the structure prediction problem and describe our algorithm, as well as an equivalence between temporal networks and the traditional transactional or 'market basket' data that is common in data mining research. This is followed by a discussion of the datasets and experimental results in Section III. Finally, we end with conclusions and future research directions in Section IV.

[1]Similar representations are also known as dynamic networks [30], time-series networks, and longitudinal data in social network analysis [31].

1-4244-0705-2/07/$20.00 ©2007 IEEE

A. Related Work

From a theoretical point of view on the evolution of networks, there have been several papers that characterize the evolution of large, complex networks such as bibliographic databases [1], [5], [33], [34] in terms of measures such as changes in vertex degree distributions over time. Behavior such as preferential attachment [34] and structural properties like the 'small-world' effect [1] have been observed in several real-world networks. Other evolving networks that have been studied are the World Wide Web and a subset of it commonly called 'blogspace'. There is a large amount of literature dealing with the various aspects of the evolution of the former [6], [9] Kumar et al. [10] examine the evolution of blogspace using *time graphs*, focusing specifically on community structure. Snijders [31] proposes a sociological behavior model for entities, coupled with Markov Chains in order to build models of temporal dynamics in social networks. The models are evaluated using statistical measures, but not empirically.

Most work in explicit link prediction has used static graphs which aggregate temporal data in some way. The survey by Getoor and Diehl [35] outlines some approaches to link mining in general, and link prediction in particular. Similarly, Desikan and Srivastava [7] outline a different three-tier approach to mining temporal networks with an emphasis on temporal networks generated from web usage logs. Liben-Nowell and Kleinberg [4] test the accuracy of several structural measures from social network analysis and web mining on ranking potential future collaborations in a bibliographic database. Although the relatively simple common neighbors indicator was found to be about as effective as more advanced techniques, the accuracy of link prediction on the aggregated test interval was still quite low.

Given a single graph with a partially known link structure, Popescul and Ungar [36] try to infer which edges are missing from the graph. Test data is generated by removing links from a bibliographic database, following which effective features for classification are extracted by searching over the space of database queries. This is similar to the problem of graph inference, where a partial structure of a single graph is given and the remaining structure has to be inferred based on topological features alone. Vert and Yamanishi [37] describe a supervised learning approach to this problem. Note that this is a different line of research that aims to reconstruct partially specified biological networks in biologically meaningful ways.

There have been few papers that explicitly use temporal data. Kempe, Kleinberg and Kumar [38] define temporal networks as static graphs where every edge is labeled with the time that the interaction took place. They define the inference problem on temporal networks as that of reconstructing time labels for unlabeled edges, given a partial labeling of the graph. O'Madadhain, Hutchins and Smyth [39] outline a method to perform explicit temporal prediction of interactions. Domain-dependent features are extracted from the temporal data, following which a probabilistic classifier is used to predict future interactions. However, their focus is on predicting future interactions which have not occurred before, whereas our method predicts *when* frequent groups of interactions will occur at the finest possible time grain, given that they have occurred at some point in the past.

Finally, our method also relies on results from the field of frequent pattern mining. Recent advances in frequent graph and itemset mining have resulted in efficient algorithms for mining both maximal and closed frequent subgraphs from a database of graphs [28], [40]–[42], as well as specialized frequent structures like cliques [28]. Efficient algorithms have also been proposed for mining frequent closed itemsets [43], [44] from large datasets, which are used in our algorithm.

B. Definitions

In this section, we formally define temporal networks and frequent subgraphs. We assume that a temporal network is being used to model interactions amongst a set of uniquely labeled entities.

Definition 1: A *temporal network* is a sequence of graphs $\mathcal{G} = \langle G_1, ..., G_T \rangle$, where $G_t = (V_t, E_t)$ is the unweighted graph of pairwise interactions at time $t \in [1, T]$. We let $V = \bigcup_t V_t$ be the universal set of entities.

Note that not all entities have to be present in all timesteps, thus $V_t \subseteq V$ for all t. Each entity $v \in V$ is uniquely labeled, and each v can appear only once at each timestep. This restriction derives from the assumption that vertices represent entities and timesteps are atomic. An entity present more than once in a single timestep therefore invalidates our definition of a temporal network. Furthermore, for any $v \in V_t$, if v is not connected to any other vertex at a timestep t, then the self-edge $(v, v) \in E_t$ is implied.

Definition 2: For any arbitrary graph $G' = (V', E')$ such that $V' \subseteq V$, we define the *support set* of G' as $S(G') = \{t \mid G' \subseteq G_t\}$, that is, the set of timesteps for which G' is a subgraph of G_t. The cardinality of the support set, $|S(G')|$, is called the *support* of G' in $\mathcal{G}$.

Definition 3: A graph G' is *frequent* in $\mathcal{G}$ if $|S(G')| \geq minsup \times T$, where $0 < minsup \leq 1$ is a fixed minimum support threshold.

Since the vertices in each graph G_t are uniquely labeled and appear only once per graph, determining whether an arbitrary graph is a subgraph of another graph does not incur the complexity of checking for graph isomorphism. This is in contrast to most research in graph mining [40], [41], [45], [46], where there is no restriction on the vertex labels of a graph and, consequently, the need to check for isomorphism among frequent subgraphs.

Definition 4: Let $\mathcal{F}$ be the set of all frequent subgraphs of $\mathcal{G}$ given a particular $minsup$. A graph $f \in \mathcal{F}$ is *maximal* if $\forall g \in \mathcal{F}$, $f \not\subset g$. A graph $f \in \mathcal{F}$ is *closed* if $\forall g \in \mathcal{F}$, such that $g \neq f$, either $f \not\subset g$ or $S(f) \not\subseteq S(g)$.

II. Prediction in Temporal Networks

We now formally define the structure prediction problem for temporal networks and state our approach.

A. Problem Statement and Approach

We define the structure prediction problem for temporal networks as follows: given a temporal network $\mathcal{G} = \langle G_1, ..., G_T \rangle$, predict a set of edges (interactions) for each G_t where $t > T$.

Our approach is based on the assumption that some interactions are good predictors for others occurring in the future. For example, an interaction x is a good predictor for y if y commonly occurs a fixed number of timesteps after x. For each ordered pair of interactions $\langle x, y \rangle$, where $x, y \in E$, we treat the temporal interval between x and y as a random variable $\Phi(x, y)$. Since the interval is temporal rather than a distance, $\Phi(x, y) \neq \Phi(y, x)$ if $x \neq y$. The former measures the interval from an occurrence of x to an occurrence of y, and the latter vice versa. Note that $\Phi(x, x)$ is valid and measures the delay between two occurrences of a particular interaction.

The approximate distribution of each random variable can be measured empirically and used to estimate the most likely temporal delay for a given pair $\langle x, y \rangle$. We shall denote such a delay as $\delta_{x,y}$. Combining the timestep in which x was last observed with the estimate for $\delta_{x,y}$, we obtain the timestep at which y is next expected to occur. The advantage of this method is that it can easily be implemented as a streaming algorithm, a requirement for most practical applications with very large graphs.

A major drawback of this approach, however, is that considering all ordered pairs of interactions has $O(|V|^4 T)$ time complexity[2]. This complexity is unacceptable even for relatively small graphs with $|V| = 100$ and $T = 100$. Moreover, the computation of the distributions of $\Phi(x, y)$ over the whole space of edge pairs would include those pairs which have no real correlation and are possibly just noise.

We propose a modified version of this approach using frequent subgraphs instead of individual edges. There are two reasons for using frequent subgraphs to reduce the input space. The first is that we aim to retain statistically significant groups of interactions while reducing the number of edge pairs for which delay distributions have to be calculated. Frequent subgraphs can be used as heuristic approximations of groups of statistically significant concurrent interactions, and algorithms for their extraction are specifically designed to operate efficiently on large databases. Furthermore, the number of closed or maximal frequent subgraphs is typically asymptotically smaller than the number of edges in a graph [40], [41].

Using frequent subgraphs, an 'interaction' is redefined to be the occurrence of a frequent subgraph at a timestep. Thus, for an ordered pair $\langle x, y \rangle$, $x, y \in \mathcal{F}$ instead of $x, y \in E$ as defined earlier. Instead of iterating over all pairs of edges at each timestep, we now iterate over all pairs of frequent subgraphs and the complexity of calculating the distributions of all $\Phi(x, y)$ drops to $O(|\mathcal{F}|^2 T)$. The problem is therefore redefined as one of predicting *frequent subgraphs* at each timestep.

Finally, we combine the use of the estimated delay with an adaptive learning component to reduce the effect of noisy predictions based on small-sample approximations of $\Phi(x, y)$. The details of this part of the algorithm are discussed in Sec. II-C.

[2]There are $O(|V|^4)$ pairs of edges in an arbitrary graph and T possible temporal delays.

B. Frequent Subgraphs

Since frequent subgraphs are being used to filter the data stream, they have to be selected carefully so as not to discard significant patterns of interaction. The obvious approach to achieve this goal is to extract all frequent subgraphs from a temporal network. However, according to the downward closure property [47] applied to graph mining, every subgraph of a frequent subgraph is also frequent. This often leads to a massive number of frequent subgraphs, limiting the utility of using frequent subgraphs instead of edge pairs.

An alternative approach is to mine only maximal frequent subgraphs [41]. No maximal frequent subgraph is a subgraph of any other maximal frequent subgraph. Thus, if a small part of a maximal frequent subgraph exists independently of its parent, it is not considered and omitted from the data stream. This approach can thus be overly restrictive because of the emphasis on maximizing the structural size of subgraphs instead of retaining common interactions.

A compromise between these two approaches is to use frequent closed subgraphs [40], which are fewer than the total number of frequent subgraphs, but not as structurally biased as maximal frequent subgraphs. Fig. 1 illustrates the difference between the three types of frequent subgraphs.

Finally, temporal networks with the constraints that we have imposed in Sec I-B have an equivalence with the more traditional transactional, or 'market basket', type of data used in frequent itemset mining [47]. Market basket data consists of a set of transactions where each transaction is a set of items drawn from a universal set. In order to transform a temporal network into a transactional database, for each graph $G_t \in \mathcal{G}$, each edge $e \in E_t$ can be uniquely represented as a concatenation of the labels of its vertices. Singleton vertices are represented by self-edges and will be included in frequent subgraphs. With any mapping from the concatenated label to the set of integers, each $e \in E_t$ becomes an 'item' and each timestep becomes a transaction of items. In this way, tools for mining frequent itemsets can be used to mine frequent subgraphs in temporal networks. Itemset mining does not incur the cost of checking for subgraph isomorphism that is required to varying degrees in general graph mining algorithms, where vertex labels do not have to be unique.

C. The Algorithm

Our algorithm consists of two main components: learning the distributions of the random variables $\Phi(x, y)$ for pairs of frequent subgraphs (steps 1 and 2 below) and an adaptive prediction module (steps 3 and 4 below). Step 0 below is a preprocessing step; the streaming prediction algorithm starts from step 1.

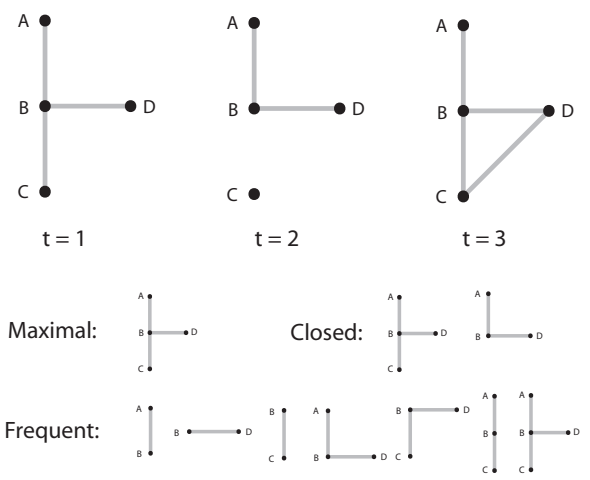


Fig. 1. A temporal network with three timestep and various types of frequent subgraphs at minimum support 2.

Step 0: Extract frequent closed subgraphs (preprocessing): We assume that all or part of the network is given and is used for extracting frequent subgraphs at a fixed minimum level of support $minsup$. We use the itemsets correspondence and MAFIA [44], a frequent itemsets algorithm, to achieve this goal.

Step 1: Update the distribution for each $\Phi(x, y)$: The algorithm maintains the timestep at which each frequent subgraph was last seen. For a frequent subgraph x, let this value be $L(x)$, initially set to 0. For each timestep t, the graph G_t is represented by a set of frequent subgraphs $F_t = \{f_1, f_2, ...\}$ rather than edges occurring in that timestep. When a new timestep $G_t \simeq F_t$ is read, for each pair of frequent subgraphs $x \in \mathcal{F}$ and $y \in F_t$, the distribution of $\Phi(x, y)$ is updated with $t - L(x)$, the observed delay between x and y. That is, the count for $\delta_{x,y} = t - L(x)$ is increased by one, and is only done if $L(x) \neq 0$. This implicitly builds the distribution matrix D, shown in Fig. 2. Finally, $L(x) = t$ if $x \in F_t$.

Step 2: Update estimates of when each frequent subgraph will occur next: The distributions of the variables $\Phi(x, y)$ are used to estimate the temporal intervals $\delta_{x,y}$ between x and y. We use the mode[3] of the distribution, denoted $\widehat{\Phi}$, as the estimate. The distribution matrix D is reduced to an estimator matrix Δ that maintains the predicted delay from x to y. This is done by reducing every entry in D with $\widehat{\Phi}(x, y)$:

$$\delta_{x,y} = \widehat{\Phi}(x, y)$$

In the case of a multimodal distribution, the mode of the smallest delay is chosen.

Step 3: Assign reliability values to each delay estimate: In order to suppress predictions which turn out unreliable, the algorithm dynamically assigns a reliability value to the elements of Δ. In addition to the mode $\delta_{x,y} = \widehat{\Phi}(x, y)$, we also have the size of the mode $|\widehat{\Phi}(x, y)|$, i.e. the number of

[3]mode: most frequent value in a discrete distribution.

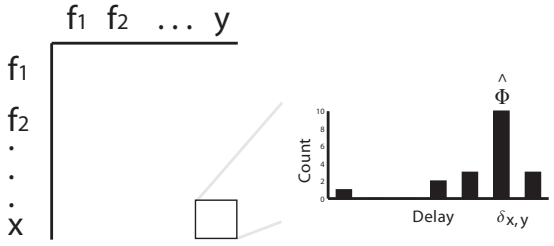


Fig. 2. The matrix D of distributions of $\Phi(x, y)$

times that the interval $\delta_{x,y}$ has occurred in the data stream. Whenever a prediction is made (detailed in step 4) using $\delta_{x,y}$ we say that it is incorrect if the predicted frequent subgraph is absent. We consider that as a negative example of $\delta_{x,y}$ being an accurate estimate of the true interval. Let $\overline{\Phi}(\delta_{x,y})$ be the number of times that a prediction based on $\delta_{x,y}$ turned out to be incorrect. The reliability value of $\delta_{x,y}$ is then defined as:

$$R(\delta_{x,y}) = \frac{|\widehat{\Phi}(x, y)|}{|\widehat{\Phi}(x, y)| + \overline{\Phi}(\delta_{x,y})}$$

Step 4: Make predictions for the next timestep: Given the current timestep t, we make predictions for the frequent subgraphs that will occur at timestep $t+1$. We predict an occurrence of y in G_{t+1} if for some $x \in \mathcal{F}$, $L(x) + \delta_{x,y} = t + 1$ with a reliability above a fixed minimum threshold τ. We compare the prediction against actual data if it is available.

We also consider a relaxed version by adding a slack variable S. A prediction for a frequent subgraph y at time t is considered correct if y is observed anywhere in the time interval $[t - S,\ t + S]$. At $S = 0$, all predicted frequent subgraphs have to occur exactly at the timestep at which they are predicted.

III. Experimental Results

In this section, we describe the implementation of our algorithm and the datasets used to test its performance and accuracy. Experimental results are presented in Sec. III-B, followed by a discussion of the results.

A. Datasets

We use four real-world temporal networks to test the accuracy of our prediction algorithm. The datasets range from networks that are relatively sparse but with a large number of vertices and timesteps, to more dense networks with fewer vertices and timesteps. Each dataset is described below and summarized in Table I. The number of timesteps listed is the number of graphs in the temporal network. Unique interactions represent a distinct instance of two individuals interacting, that is, multiple instances of two individuals interacting are counted once. The sums of all interactions over the whole network are listed under total interactions.

1) Enron e-mails: The Enron e-mail corpus is a publicly available database of e-mails sent by and to employees of the now defunct Enron corporation [48], [49]. The corpus was made available in 2003 by the Federal Energy Regulatory Commission during their investigation of the company. We use a cleaner version of the original dataset with fewer integrity issues [50]. Timestamps were extracted from message headers for each day of e-mail activity. Entities are identified by their e-mail address, and an interaction is said to occur if an e-mail was sent between two entities on a given day. Only one interaction per day is incurred even if multiple emails are sent between any two entities. The direction of the email is discarded in our experiments.

2) Plains zebras: Social interactions between Plains zebra (*Equus burchelli*) in Kenya were recorded by direct observations made by behavioral ecologists from Princeton University [51]. The data is made from visual scans of the populations, typically once a day, over periods of several months. Each entity is a zebra, uniquely identified by the pattern of its stripes. Each spatially proximate group of animals, as determined by GPS coordinates, represents a complete set of interactions amongst those individuals.

3) Stock market: Correlations between closing prices of stocks over a time period can be represented as a temporal network. Using publicly available historical data for the New York Stock Exchange (NYSE) ranging from 1988 to 2006 [52], we obtained daily closing prices for 3,416 stocks. For every pair of stocks (S_1, S_2), the correlation coefficient ρ is defined as $\rho = \dfrac{cov(S_1, S_2)}{\sigma_1 \times \sigma_2}$, where $cov(S_1, S_2)$ is the covariance of the daily closing prices of S_1 and S_2 over a sampling period and σ_1 and σ_2 are the standard deviations of the stock prices in the same period. For every successive sampling period, an edge (S_1, S_2) exists if $\rho(S_1, S_2) \geq \alpha$, where α is a minimum $apriori$ chosen correlation value. This form of the stock graph has been shown to have the scale-free property common to many real-world social networks [29], as well as frequently occurring patterns [28].

We use two versions of the stock graph, one with a sampling period of 30 days and $\alpha = 0.9$ (Stock Market 1) and another with a sampling period of 5 days and $\alpha = 0.98$ (Stock Market 2). Furthermore, starting from the most recent quotes, we scan backwards and chose as entities a subset of stocks that have been continuously traded throughout the entire period. We thus obtain a set of aligned stock time series of identical length. This removes what would otherwise have been a bias in the frequent subgraphs toward patterns involving stocks that have been traded for longer periods of time. The fixed minimum number of stocks chosen are 1,000 (Stock Market 1) and 2,500 (Stock Market 2).

B. Experiment Parameters and Results

We designed a series of experiments to evaluate the performance of our algorithm and test its sensitivity to parameters. The parameters for the streaming component of the algorithm are τ, the minimum reliability required to make a prediction, and S, the slack variable. However, the use of frequent subgraphs introduces the parameters $minsup$, the minimum support for mining frequent subgraphs, and $train\_size$, the size of the initial network from which frequent subgraphs are extracted.

TABLE I
TEST DATASET CHARACTERISTICS.

Dataset	Timesteps	Entities	Interactions	
			total	*unique*
Enron	1,213	75,026	936,032	295,233
Plains Zebras	228	1,002	435,926	168,787
Stock Market 1	104	1,000	1,415,815	501,499
Stock Market 2	121	2,500	4,910,962	2,482,911

TABLE II
SUMMARY OF EXPERIMENTAL PARAMETERS.

Parameter	Description	Range
$minsup$	Minimum support for mining	[0, 1]
$train\_size$	Number of initial timesteps to use for extracting frequent subgraphs	[2, T]
τ	Minimum reliability for making a prediction	[0, 1]
S	Slack interval for validating predictions	[0, T]

Each dataset is first converted to a transactional itemset representation using the transformation described in Section II-B. Frequent closed subgraphs are then extracted from the first $train\_size$ timesteps using the MAFIA algorithm. The (fractional) $minsup$ value only applies to this initial portion of the network, i.e. the absolute minimum support for any frequent closed subgraph is $minsup \times train\_size$. We decreased $minsup$ values from 1 until the number of frequent subgraphs produced became too large, typically around supports of 1-3 timesteps. In general, note that at a $minsup$ of 1 timestep, every timestep is a frequent closed subgraph. Table II summarizes the experimental parameters.

After mining frequent closed itemsets, our prediction algorithm is run on the entire network. The inclusion of the 'training' timesteps does not bias the accuracy of the algorithm because their purpose is to build a filter and not to learn a model, which is done continuously. The algorithm is run at a range of $minsup$, $train\_size$, S and τ values, with the slack being varied in the range $[0, 2]$. The algorithm was implemented in C and run on a dual-core Pentium D system (32-bit) running at 3.2 GHz with 4 GB of RAM and Linux kernel 2.6.12, and did not exceed a run time of 5 minutes in any instance.

Fig. 3 shows the accuracy of the algorithm on each dataset for different values of $minsup$ and slack. The τ parameter for these runs was fixed at 0.9 to filter out unreliable predictions. The $train\_size$ parameter was set to 20% of the total number of timesteps for each dataset. The range of $minsup$ was chosen separately for each dataset based on the number of

timesteps and frequent subgraphs. At the highest $minsup$ for each dataset, there were only a few frequent subgraphs, whereas the lowest $minsup$ represents the largest number of frequent subgraphs that were tractable on our machine, typically ranging in the thousands. The average accuracy across datasets at $S = 1$ was 85.7%.

In order to judge the contribution of the adaptive prediction module to the overall accuracy (as opposed to just the distributions of $\Phi(x, y)$), we ran a series of experiments to determine the effect of the τ parameter. Fig. 4 shows the effect of increasing τ on average accuracy across all datasets. The $minsup$ used for each dataset was the middle of the ranges given in Fig. 3.

Fig. 5 shows the effect of increasing the proportion of the network from which frequent subgraphs are extracted. The $minsup$ used for each dataset is the same as that of Fig. 4. The τ value was fixed at 0.9.

C. Discussion

The most prominent result of our experiments is that the proposed network structure prediction algorithm is highly accurate (with one slack time step). However, predicting structure precisely at each timestep ($S = 0$) is a difficult task. It is highly unlikely that real-world processes would have completely and precisely predictable patterns of interaction, an aspect which is exacerbated by an arbitrary quantization into timesteps. The arbitrariness of timesteps is emphasized by the dramatic improvement in accuracy when a single slack timestep is allowed for structure prediction, and marginally thereafter for larger slack intervals.

By increasing the slack value S, we increase the time interval in which a prediction is considered accurate and thus increase the probability that an arbitrary prediction is found to be correct. As S increases to infinity, it is expected that the accuracy increases towards 100%. However, as we have mentioned, 100% accuracy is quite unlikely when the dataset concerned is a real-world processes and would require the algorithm to make correct predictions even with very small samples for the distributions of $\Phi(x, y)$.

A trend in the results shown in Fig 3 is that accuracy increases as the minimum support for frequent subgraph mining is increased. This is partially caused by the number of frequent subgraphs dropping as $minsup$ is increased. Delays between frequent subgraphs at a higher support seem to be more predictable, as indicated by the increase in accuracy with higher $minsup$ values. The degree to which the accuracy increases varies across datasets. For example, Stock Market 1 shows relatively little change until $minsup$ is increased to around 0.4 (about 10 timesteps, or 300 business days), whereas the Enron dataset shows a smoother increase in accuracy. Intuitively, this suggests that a small number of frequent patterns of interaction are very predictable, whereas a larger number of less frequent patterns are less predictable. The use of frequent subgraphs thus seems to be effective as a filter for statistically significant interactions.

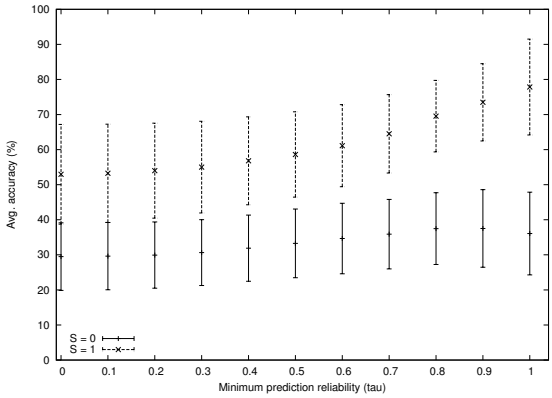


Fig. 4. Effect of varying minimum prediction reliability τ.

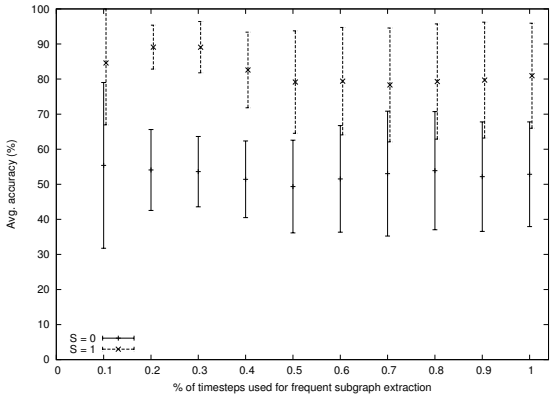
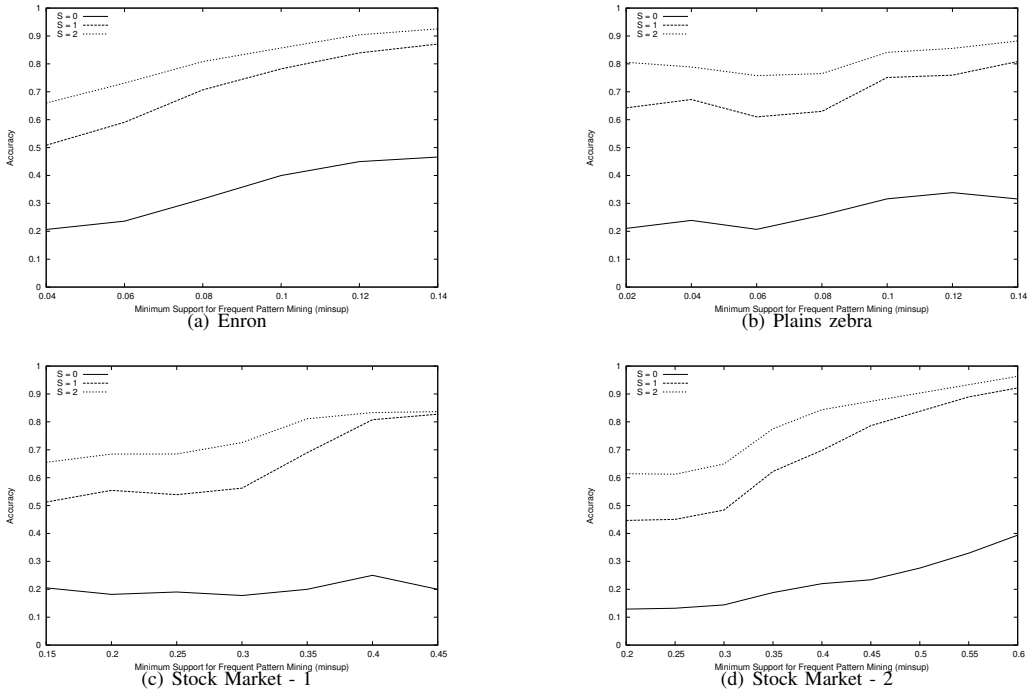


Fig. 5. Effect of varying size of training network.

The use of frequent subgraphs is only one possible method to reduce the space of entities being considered for temporal dependence. The tradeoff is that fewer frequent subgraphs are likely to cover a smaller subset of individuals in the population. Noting that our algorithm predicts the occurrence of frequent subgraphs, this leads to predictions covering fewer individuals overall. A simple way to relax this restriction is to maintain a buffer of the most recent timesteps and continually extract local, as opposed to global, frequent subgraphs. This, and other approaches, will be addressed in future work.

The adaptive prediction module is instrumental for high accuracies. Fig. 4 shows the effect of varying τ for representative values of $minsup$ and $train\_size$. At $\tau = 0$, the adaptive module is effectively switched off. Every possible prediction that can be made is made at each timestep with no filtering whatsoever. At $\tau = 1$, only $\delta_{x,y}$ values which have never proven to be wrong are used to make predictions. Naturally, this makes the prediction more sensitive to noise, as illustrated by the slight drop in accuracy at $\tau = 1$ for $S = 0$. At a higher slack value, predictions are less likely to be wrong with a small amount of noise in the data and increasing τ improves performance significantly.

Finally, efficiency is a major motivating factor in our design. With large networks, and moreover, streaming network data



(a) Enron (b) Plains zebra

(c) Stock Market - 1 (d) Stock Market - 2

Fig. 3. Accuracy at different levels of minimum mining support and slack.

becoming available, many theoretical approaches do not scale well. Our work takes advantage of mature, mainstream data mining algorithms which are specifically designed to handle large datasets. Our method is best suited to streaming data and applications that require real-time predictions.

IV. Conclusion and Future Work

We have presented a generic, online algorithm for predicting a partial structure of a temporal network. Our algorithm is accurately able to predict the structure of a temporal network at an arbitrary unseen timestep with a slack of one timestep. It can work as an online, streaming algorithm by continually updating its model of the underlying process, or be used to predict the next occurrence of each interaction based on a static learned model. It is thus suitable for applications with streaming data where real-time predictions are required, as well as for more analytical applications where the evolution of a process needs to be modeled precisely. Although the noise associated with precise predictions holds for all our datasets, predictions made for a particular timestep, give or take a timestep, allow us to model, with a high level of confidence, when interactions will occur in a diverse range of domains. We were able to accurately predict, within one timestep, correlations in stock prices, e-mail activity in a corporation, and social groupings in populations of Plains zebra, with the resolution of a timestep varying from one day for the e-mail domain to 30 days for a version of the stock market graph.

While the prediction of a partial structure of the network at each timestep has many applications, there is also a lot of scope for future work. Perhaps most importantly, in our current approach, frequent subgraphs are extracted once from a training network. This precludes the possibility of capturing groups of interactions which become frequent at a later point. What is needed is a dynamic filter for statistically significant interactions. One possible solution is to use locally frequent subgraphs, or subgraphs that have been frequent in the last t timesteps. These could be dynamically extracted from the data stream, possibly replacing older subgraphs.

There are other possible extensions and variations of the algorithm specific to a given application domain. However, the proposed algorithm provides an efficient and accurate basis for those extensions.

Acknowledgments

We would like to thank Dan Rubenstein, Ilya Fischhoff, and Siva Sundaresan of the Department of Ecology and Evolutionary Biology at Princeton University for sharing the Plains Zebra data. Their work was supported by the NSF grants CNS-025214 and IOB-9874523. We are grateful to the anonymous reviewers for their constructive comments. This work is supported by the Microsoft award 14936.

References

[1] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A: Statistical Mechanics and its Applications*, vol. 311, no. 3-4, pp. 590–614, August 2002. [Online]. Available: http://dx.doi.org/10.1016/S0378-4371(02)00736-7

[2] K. Börner, J. Maru, and R. Goldstone, "The simultaneous evolution of author and paper networks," *PNAS*, vol. 101, no. Suppl 1, pp. 5266–5273, 2004.

[3] K. Börner, L. DallAsta, W. Ke, and A. Vespignani, "Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams," in *Complexity, Special issue on Understanding Complex Systems*, 2006, in press.

[4] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

[5] J. Ramasco, S. Dorogovtsev, and R. Pastor-Satorras, "Self-organization of collaboration networks," *Physical Review E*, vol. 70, no. 3, p. 36106, 2004.

[6] Z. Bar-Yossef, A. Broder, R. Kumar, and A. Tomkins, "Sic transit gloria telae: Towards an understanding of the web's decay," in *Proc. WWW '04*.

[7] P. Desikan and J. Srivastava, "Mining temporally evolving graphs." New York, NY, USA: ACM Press, 2004, pp. 13–22.

[8] J. Kleinberg, "Temporal dynamics of on-line information streams," draft chapter for the forthcoming book Data Stream Management: Processing High-Speed Data Streams (M. Garofalakis, J. Gehrke, R. Rastogi, eds.), Springer.

[9] W. Koehler, "A longitudinal study of web pages continued: a consideration of document persistence," *Information Research*, vol. 9, no. 2, p. paper 174, 2004, [Available at http://InformationR.net/ir/9-2/paper174.html].

[10] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins, "On the Bursty Evolution of Blogspace," *World Wide Web*, vol. 8, no. 2, pp. 159–178, 2005.

[11] A. Ntoulas, J. Cho, and C. Olston, "What's new on the web? The evolution of the web from a search engine perspective," in *Proc. WWW'04*.

[12] R. Pastor-Satorras and A. Vespignani, *Evolution and structure of the Internet*. Cambridge: Cambridge University Press, 2004.

[13] D. Spinellis, "The decay and failures of web references," *Communications of the ACM*, vol. 46, pp. 71–77, 2003.

[14] S. Eubank, V. Kumar, M. Marathe, A. Srinivasan, and N. Wang, "Structural and algorithmic aspects of massive social networks," *Proc. 15th ACM-SIAM Symp. on Discrete algorithms (SODA)*, pp. 718–727, 2004.

[15] S. Eubank, H. Guclu, V. Kumar, M. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang, "Modelling disease outbreaks in realistic urban social networks," *Nature*, vol. 429, pp. 429:180–184., Nov 2004, supplement material.

[16] M. Keeling, "The effects of local spatial structure on epidemiological invasions," *Proc. R. Soc. Lond. B*, vol. 266, pp. 859–867, 1999.

[17] M. Kretzschmar and M. Morris, "Measures of concurrency in networks and the spread of infectious disease," *Math. Biosci.*, vol. 133, pp. 165–195, 1996.

[18] L. A. Meyers, M. Newman, and B. Pourbohloul, "Predicting epidemics on directed contact networks," *Journal of Theoretical Biology*, vol. 240, pp. 400–418, 2006.

[19] J. M. Read and M. J. Keeling, "Disease evolution on networks: the role of contact structure," *Proc. R. Soc. Lond. B*, vol. 270, pp. 699–708, 2003.

[20] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. Wallace, "Discovering hidden groups in communication networks," in *Proceedings of the 2nd NSF/NIJ Symposium on Intelligence and Security Informatics*, 2004.

[21] A. Broido and K. Claffy, "Internet topology: connectivity of ip graphs," in *Proceedings of SPIE ITCom*, 2001.

[22] K. Carley, "Communicating new ideas: The potential impact of information and telecommunication technology," *Technology in Society*, vol. 18, no. 2, pp. 219–230, 1996.

[23] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *Proc. WWW '04*, pp. 491–501.

[24] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.

[25] M. Tsvetovat, K. Sycara, Y. Chen, and J. Ying, "Customer coalitions in electronic marketplaces," in *Agent-Mediated Electronic Commerce III, Lecture Notes on Artificial Intelligence*, U. C. Frank Dignum, Ed. Springer-Verlag, 2003.

[26] J. Tyler, D. Wilkinson, and B. Huberman, "Email as spectroscopy: Automated discovery of community structure within organizations," in *Proc. 1st International Conf. on Communities and Technologies*, 2003.

[27] B. Wellman, "An electronic group is virtually a social network," in *Culture of the Internet*, S. Kiesler, Ed. Mahwah, NJ: Lawrence Erlbaum, 1997, pp. 179–205.

[28] J. Wang, Z. Zeng, and L. Zhou, "Clan: An algorithm for mining closed cliques from large dense graph databases." Los Alamitos, CA, USA: IEEE Computer Society, 2006, p. 73.

[29] H. Kim, I. Kim, Y. Lee, and B. Kahng, "Scale-Free Network in Stock Markets," *J. Korean Physical Society*, vol. 40, no. 6, pp. 1105–1108, 2002.

[30] R. Breiger, K. Carley, and P. Pattison, Eds., *Dynamic Social Network Modeling and Analysis*. Washington, D.C.: The National Academies Press, 2003.

[31] T. Snijders, "The Statistical Evaluation of Social Network Dynamics," *Sociological Methodology*, vol. 31, no. 1, pp. 361–395, 2001.

[32] D. Kempe, J. Kleinberg, and A. Kumar, "Connectivity and inference problems for temporal networks," *J. Comput. Syst. Sci.*, vol. 64, no. 4, pp. 820–842, 2002.

[33] M. E. J. Newman, "From the Cover: The structure of scientific collaboration networks," *PNAS*, vol. 98, no. 2, pp. 404–409, 2001. [Online]. Available: http://www.pnas.org/cgi/content/abstract/98/2/404

[34] ——, "Clustering and preferential attachment in growing networks," *Physical Review E*, vol. 64, no. 2, p. 25102, 2001.

[35] L. Getoor and C. P. Diehl, "Link mining: a survey," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 3–12, 2005.

[36] A. Popescul and L. Ungar, "Statistical relational learning for link prediction," *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[37] J. Vert and Y. Yamanishi, "Supervised graph inference," *Advances in Neural Information Processing Systems*, vol. 17, pp. 1433–1440, 2005.

[38] D. Kempe, J. Kleinberg, and A. Kumar, "Connectivity and inference problems for temporal networks," *Proc. 32nd ACM Symp. on Theory of Computing (STOC)*, pp. 504–513, 1999.

[39] J. O'Madadhain, J. Hutchins, and P. Smyth, "Prediction and ranking algorithms for event-based network data," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 23–30, 2005.

[40] X. Yan and J. Han, "Closegraph: mining closed frequent graph patterns," in *Proc. 9th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining (KDD)*. New York, NY, USA: ACM Press, 2003, pp. 286–295.

[41] J. Huan, W. Wang, J. Prins, and J. Yang, "Spin: mining maximal frequent subgraphs from graph databases," in *Proc. 10th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining (KDD)*. New York, NY, USA: ACM Press, 2004, pp. 581–586.

[42] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi, "Scalable mining of large disk-based graph databases," in *Proc. 10th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining (KDD)*. New York, NY, USA: ACM Press, 2004, pp. 316–325.

[43] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets," *Proc. IEEE ICDM03 Workshop FIMI03*, 2003.

[44] D. Burdick, M. Calimlim, and J. Gehrke, "MAFIA: A maximal frequent itemset algorithm for transactional databases," *Proc. 17th International Conference on Data Engineering*, pp. 443–452, 2001.

[45] X. Yan and J. Han, "gSpan: graph-based substructure pattern mining," 2002, pp. 721–724.

[46] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," 2001, pp. 313–320.

[47] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pp. 487–499, 1994.

[48] B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research," *Proc. European Conf. on Machine Learning*, 2004.

[49] G. Kolata, "Ideas and trends; enron offers an unlikely boost to e-mail surveillance," *New York Times*, May 22 2005.

[50] J. I. Adibi, "Enron email dataset." [Online]. Available: {http://www.isi.edu/~adibi/Enron/Enron.htm}

[51] I. R. Fischhoff, S. R. Sundaresan, J. Cordingley, H. M. Larkin, M. J. Sellier, and D. I. Rubenstein, "Social relationships and reproductive state influence leadership roles in movements of Plains zebra (*Equus burchelli*)," *Animal Behaviour (in press)*.

[52] "Yahoo! Finance," accessed June 20, 2006. [Online]. Available: http://finance.yahoo.com/