Measuring and Mining Dynamic Networks

BY

MAYANK LAHIRI
B.S. (Hons.), Lafayette College, 2005

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2011

Chicago, Illinois

Dedicated to my maternal grandfather, Kalyan – the first Ph.D. in my immediate family, who passed away less than three months before my defense, and my maternal grandmother Anita, who passed away a month later.

ACKNOWLEDGMENTS

This thesis would not have been possible without my family. Certainly, I would not have existed without the cooperation of my parents, but in a larger sense, I could not have asked for a better support structure, more encouragement in the face of unending education and perpetual whining induced by ramen-poisoning, less concern about gainful employment forsaken for an incomprehensible drive to study zebras and dots and squiggly lines year after passing year, from a wonderful family – who never once asked about graduation, who joined in the highs and lows of the fog of research, and who were totally and completely happy knowing that I was busy tickling my brain in esoteric and impractical ways.

This thesis would also not have been possible without my advisor, Tanya Berger-Wolf. She has failed to show any of the hallmarks of a 'traditional' Ph.D. advisor, and I would be remiss not to mention these eccentricities: we are still on talking terms despite the fact that she has *already signed* my thesis approval form; we are writing research papers together despite the fact that I *have already graduated*; I have *never* been frustrated enough to consider quitting my Ph.D. in a fit of anger and disgust; she has *not once* forbidden me from working on a pet research topic or going on a wild goose chase, and perhaps most shocking of all, she has been a *constant* source of encouragement and advice, *even* after receiving tenure. This is probably because she is *also* my Ph.D. advisor, in addition to equally prominent roles as a true mentor, confidante, and friend. I was truly lucky that she chose to hire me, an event that goes beyond mere good fortune to winning the lottery.

ACKNOWLEDGMENTS (Continued)

Over the years, a number of wonderful people have graced my life in Chicago and softened the Ph.D. experience. I am immensely grateful to Ishaan Joshi for the freewheeling conversation over many pints at the Drum and Monkey pub on Taylor Street, to Kapil Thadani for being a source of tremendous support and inspiration over many, many years, to my friends and colleagues Habiba, Rajmonda Sulo, Arun Maiya, Chayant Tantipathananandh, and Paul Varkey in the Computational Population Biology lab for some unforgettable experiences, including safaris in Kenya and an eventful salsa night at a small bar in Pisa, Italy. All this has been made possible by the amazing staff at the CS department at UIC, who deftly weave arcane university policy into practicality. Finally, a giant thank you to my girlfriend Nova, for putting up with incessant complaining and being a great source of support during the production of this thesis.

ML

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

FIGURE **PAGE**

SUMMARY

A dynamic network is a mathematical graph representation for time-varying natural systems that consist of many independently interacting entities. In particular, dynamic networks are most useful when the system exhibits extremely complex behavior and is continually changing over time, and when only a record of the interactions between entities are observed. In these cases, the goal is to learn about the underlying natural system by studying the dynamic behavior of the entities in it. By dealing with a graph representation, we can abstract away the increasingly diverse sources of dynamic network data, and extract information from the dynamics of interactions alone.

We start by analyzing how the change in structure of networks are quantified over time. This is a fundamental technique used in a variety of fields: for example, can we determine if the average shortest-path length between Internet routers is gradually decreasing from samples of its structure over time? By analyzing a bibliographic database like DBLP, can we conclude that publishing scientists are forming increasingly more connected collaboration networks? We survey existing techniques, and make a case for more sophisticated measurement and mining methods.

The first of these new methods is an efficient Fourier-like decomposition for dynamic network data that allows us to analyze the periodicities and periodic patterns present in a dynamic network dataset. Using this tool on several datasets, we find that the method is both efficient and able to recover a spectrum of plausibile periodicities in real dynamic

networks. Given that there is interesting information that can be extracted from network dynamics, the second new technique proposed is a data mining technique for finding interactions in a dynamic network where the occurrence of one predicts the other. This allows us to tease apart the parts of a network that are regular and predictable.

All the methods we propose serve the broader goal of characterizing and quantifying network dynamics, as advanced forms of measurement that make a minimal set of assumptions about the underlying system.

CHAPTER 1

INTRODUCTION

This thesis is about *dynamic network data*, with an individual emphasis on each word. In the most general sense, a dynamic network is a very powerful mathematical representation for time-varying systems that consist of many interacting entities. In particular, dynamic networks are most useful when the system exhibits extremely complex behavior and is continually changing over time, and when only a record of the interactions between entities are observed. One of the best known instances of network representation applied to a real-world system is the *social network* (Wasserman and Faust, 1994), where the relationships between people are mapped into a web-like network. The structure of this web of human connections has been used for decades to gain insight into the structure of human societies and behavior, and more recently animal societies and behavior as well. A dynamic network is a much more recent and powerful representation than a canonical social network that allows one to explicitly map the changing structure of such a web over time. It is a generic, graph-theoretic representation that can be applied to systems much more diverse than human and animal social connections. In this thesis, we will describe correspondingly generic methods for analyzing any dynamic network dataset in successively more complex ways.

The focus of this thesis is on developing generic analytical methods, rather than focusing on the analysis of networks from a specific domain. Since the diversity of sources of dynamic network data has been steadily increasing (Newman, 2003), the problems in this thesis are

driven by the need for analytical tools that are powerful under a minimal set of assumptions, and as agnostic as possible to the characteristics of any single class of domains. As examples of this data diversity, dynamic network datasets can be produced by massive industrial logging systems attached to communications switches, or by researchers scraping the World Wide Web from a laptop; by social scientists painstakingly interviewing human subjects over many years, or by automatically processing the log files from a web server. Where this data might have earlier been condensed into simpler formats, new techniques for handling very large dynamic networks can now enable more powerful network analysis. In particular, this thesis examines and builds tools for analyzing the *dynamics* of networks, *i.e.,* the way in which its structure is changing over time, to extract information about the underlying system. The tools developed here fit between the statistician John Tukey's original notion of exploratory data analysis (Tukey, 1980), where the focus is on probing data to generate questions and hypotheses for subsequent investigation[1], and modern data mining, where the focus is on extracting the statistically 'interesting' parts of massive datasets under various definitions of what constitutes interesting.

Returning to the three emphasized words in the opening statement, a *network* is a graph-theoretic representation of a system where individually identifiable entities, either physical or abstract, interact with each other. Almost any interconnected system can be represented as a network, the most common representation for which is a graph with labeled vertices.

---

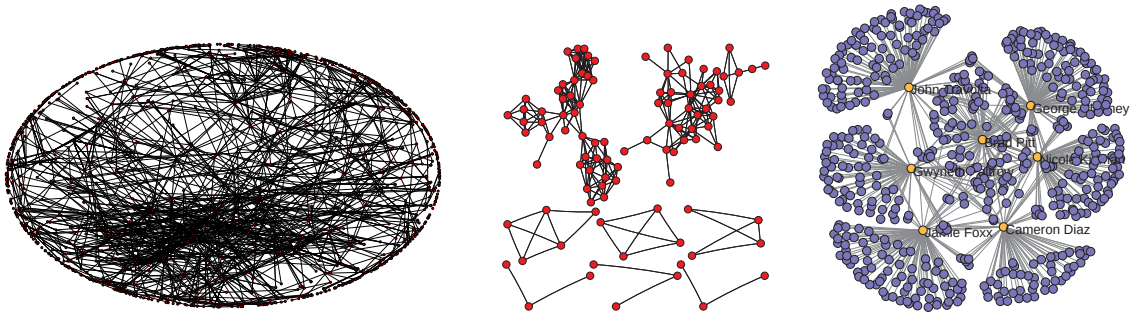
[1]As opposed to *confirmatory data analysis* methods like hypothesis testing, *etc.*, which presume the existence of a hypothesis.

Figure 1 shows an example of a graph drawing of three different network datasets. Figure 1a is a scientific co-authorship network, in which vertices represent individual researchers in theoretical computer science, and an edge between any pair of researchers indicates that they have published an academic paper together. Collaboration and citation networks can be easily extracted from computerized bibliographic databases. Figure 1b is a network representation of a type of animal association data that is routinely collected by ecologists and field biologists. Ecologists recorded social associations between wild horses on Shackleford Island, North Carolina over a period of three months.[1] Ordinarily, this association data might have been summarized into simpler statistics such as the mean group size, but networks offer a powerful new way to visualize and analyze all the information present in association data. Finally, Figure 1c illustrates the fact that network datasets can emerge from unusual places. The Internet Movie Database maintains a large online repository of publicly available photographs of actors, musicians, movie directors, and other individuals associated with the entertainment industry in the United States, taken at various times when the individuals in question appear in public. These photographs lead to a natural approximation of the professional association network between these individuals, and how it changes over time.[2]

Moving on to the second emphasized word in the opening statement of this chapter, we focus on *dynamic* networks in this thesis. It should be clear that the systems shown

---

[1]Data courtesy of Prof. Daniel I. Rubenstein of the Princeton Equid Research Group.



(a) Co-authorships of academic papers between researchers in theoretical computer science.

(b) Social associations between wild horses on Shackleford Island, North Carolina.

(c) Social associations of the top 7 Hollywood celebrities by PageRank, determined by co-appearance in publicly available photographs.

Figure 1: Three examples of physical systems represented as networks.

in Figure 1 are not frozen in time, but are in essence 'network snapshots' of continually changing systems.[1] New researchers are continually born, and existing researchers form new collaborations, so the co-authorship network in Figure 1a is certainly evolving over time, as are the other two networks depicted. There are therefore two components to most networks: its structure at any point in time, such as the visualizations in Figure 1, and the dynamics of the underlying system that drive the formation and evolution of this structure

[2]The IMDB data is obtained using a method (photographing public sightings) that is curiously similar to the one used to collect social association data on the Shackleford horses and other wild animals (Lahiri et al., 2011), and indicative of the increasing use of technology to collect association and interaction data using technologies such as short-range wireless links and GPS positioning (Juang et al., 2002).

[1]The reason why these static snapshots allow us to draw any inferences about the system at all is a network analogue of the concept of statistical consistency: under certain assumptions, the larger our snapshot gets, the more likely it is to be an accurate representation of the underlying system.

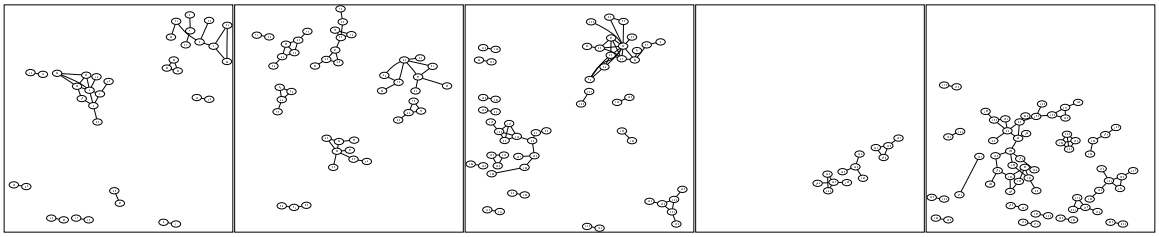


Figure 2: Individual network snapshots from five consecutive days of the IMDB photo-based dynamic network. A different portion of the larger, aggregate network is shown in Figure 1c.

through time. These dynamics have generally been difficult to study because of the limited availability of data with a temporal component. However, this has changed with recent technological advances, resulting in large and detailed datasets that depict the structures shown in Figure 1 through the time dimension.

Given a dynamic network dataset, we first address the most basic of analytical tasks:

- *How are network properties changing over time?* Graph theoretic measures summarize the structure of a graph into numeric scalar or vector values. Some examples are the basic counts of vertices and edges in a graph, the empirical distribution of shortest-path lengths between all pairs of connected vertices, and the spectrum of a graph. Since a dynamic network is a graph that changes over time, a fundamental question to ask is if and how these measures are changing as the network changes. For example, are scientific co-authorship networks getting denser over time, possibly indicating that collaboration between scientists is on the rise? Is the average number of hops between routers on the Internet decreasing over time, even at the rate that the Internet is

currently growing? And could a trend in its clustering properties over time suggest, for example, that simpler decentralized packet routing algorithms could be used for efficient routing of packets (Kleinberg, 2000)?

In Chapter 2, we start with a methodological description of how basic structural properties of dynamic networks are measured from a collected dataset. Among other uses, these measurements form the basis of a very large class of *generative stochastic models* for the dynamics of the underlying physical system (Chakrabarti et al., 2010), and which are also used to generate realistic synthetic data to test network-based algorithms (Bilgic and Getoor, 2008). We note that dynamic network datasets are almost always susceptible to various kinds of non-trivial measurement errors and missing data issues, so it is important to select graph measures that are robust under such circumstances. While the effects of sampling biases and missing data have been extensively studied in the context of a single static network (Ghani et al., 1998; Costenbader and Valente, 2003; Kossinets, 2006; Achlioptas et al., 2009), this has not been the case with dynamic measurements of networks. We show that the temporal properties of dynamic networks as reported in the literature use one of two basic aggregation methods, and a small set of simple graph theoretic measures. We also show that almost all of these measures are sensitive to incomplete data under one of the two aggregation methods, and can erroneously suggest extremely prominent trends where none, or contrary ones, truly exist.

We therefore demonstrate that even the task of determining how basic network measures like diameter are changing over time is not necessarily a settled issue. We suggest

some alternatives, but conjecture that there is no generally applicable way to correct or even estimate the measurement bias without making additional assumptions about network dynamics and the size and gross structure of the missing data. This suggests the need to look laterally at a network through the time dimension directly to analyze dynamics. Noting that a dynamic network is analogous to a time series of changing graphs, we choose approaches that are inspired by methods for dealing with numeric time series. In signal analysis, the Fourier transform is a standard tool that is used to decompose a time-varying signal into a sum of sinusoidal components; we develop a similar tool for dynamic networks in Chapter 3 to answer the following question.

- *What are the periodicities and periodic patterns present in a dynamic network?* In systems like communications networks, there are likely to be patterns of interactions between specific individuals that recur or a periodic basis. If we can extract all such patterns in a tractable, succinct representation, we can determine a spectrum of periodicities in the interactions within a physical system similar to the frequency domain representation of time-varying numerical signals. For example, at what frequency do research groups publish in different fields, and do animal associations exhibit cyclical behavior? Alternatively, the periodic patterns themselves could be used for algorithmic tasks like clustering individuals into communities of individuals that interact periodically.

Our formulation of the problem above incorporates the notion of *local periodicity*, where a pattern of connections between individuals may exhibit periodic behavior only temporarily

relative to the time extent of the dataset. Although there have been a number of approaches to mining periodic patterns in related types of data, we develop an asymptotically more succinct formulation of the problem that does not lose any information present in other formulation. Furthermore, our formulation of the problem naturally yields a polynomial time and space algorithm in the size of the input dataset, stemming from a worst-case output size that is provably polynomial in the size of the input. Our algorithm efficiently mines periodic patterns at all possible periodicities (a number of other mining algorithms require the user to input 'likely' periodicities), performing orders of magnitude faster on real datasets than its theoretical worst-case bound, and the frequency domain spectra we obtain reveal very plausible principle periodicities in various physical systems.

Since we were able to extract an appreciable amount of information from our periodic pattern mining formulation, a natural extension would be to attempt to detect more sophisticated forms of temporal relationships than periodicity. In doing so, we have to abandon the frequency-domain spectrum of periodicities, and focus solely on structural patterns instead:

- *Which interactions in a dynamic network are strongly correlated in time?* In Figure 2, it is difficult to tell whether any edges are temporally correlated with each other. In networks with hundreds of thousands or millions of edges that may appear entirely chaotic, is it possible to wean out the few that are temporally correlated? For example, can we look through the plethora of interactions and tell that a particular pair of scientists publishing a paper in the current year is a good predictor of a different pair

of scientists publishing a paper in the following year, or that an e-mail from a student to their Ph.D. advisor at any time is a reliable predictor of a reply within an hour?

Given the sheer size of typical dynamic networks, this problem can easily become intractable. Although the underlying physical system being represented by a dynamic network can sometimes arise from applying simple local rules, going in the other direction without knowing those rules is a formidable problem. We propose a formulation of this problem in Chapter 4 to mine a rich but limited set of predictably coupled interactions in a dynamic network. When posed as a data mining problem, the patterns of interest are specific interactions that reliably predict future occurrences of themselves or other interactions. Our contribution consists of a novel method of modeling and evaluating dependencies between interactions, in order to yield data mining results with a degree of generalization. In the later part of Chapter 4, we demonstrate some practical applications of mining strong relationships.

In the next section, we briefly survey the historical development of dynamic network data for context, as well as other related representations. Each subsequent chapter deals with one of the three problems described earlier, and more detailed surveys of the literature specific to each problem are contained within individual chapters. Section 1.2 contains a summary of the technical contributions of this thesis, and Section 1.3 briefly describes some of the datasets used throughout the thesis.

1.1 A brief history of network data

In the introductory paragraph of this thesis, we mentioned that there would be an individual emphasis on data, since all the methods presented in this thesis deal with inferring aspects of the underlying system from a dynamic network dataset. A number of developments in network analysis have been directed by the availability of large, comprehensive datasets, and so we briefly summarize the history of dynamic network data in this section, as well as the theoretical developments that accompanied the availability of datasets with greater detail.

Prior to the late 1990s, the analysis of real networks was an endeavor that was largely restricted to the fields of sociology, bibliometrics, and computer networks. In sociology, social network analysis had precedents as far back as the 1930s (Moreno, 1934). However, the only practical way to collect network data was interviews with subjects. This severely limited the size of datasets and the ability to make inferences at a large scale. For example, a benchmark network dataset in social network analysis, called the *Southern Women* dataset, consists of just 18 individuals and can be printed in its entirety in a publishable table. A meta-analysis of this dataset identified 21 published attempts at analyzing it, each with different methods (Freeman, 2003). Another sociological dataset, called the *Zachary karate club*, consists of approximately 100 individuals (Zachary, 1977), and is still used to test modern analytical methods (Newman and Leicht, 2007; Tong et al., 2010). By modern standards, these would be woefully inadequate datasets, but have nonetheless helped

develop a comprehensive literature on analytical techniques that use graph theory to gain sociological insights (Wasserman and Faust, 1994).

The current interest in network analysis, both static and dynamic, can probably be largely attributed to two developments in the late 1990s. The first is that physicists discovered that real networks look quite similar in certain ways, but at the same time quite different from purely random graphs. The best-known precursor to modern dynamic graph models is the 1960 publication of Erdős and Rényi (Erdős and Rényi, 1960), which analytically describes the structural properties of randomly generated graphs as the number of vertices (and edges) is increased. The so-called Erdős-Rényi (ER) random graph model was not intended to be realistic models, with the authors noting that "if one aims at describing ... a real situation, one should replace the hypothesis of equiprobability of all [edges] by some more realistic hypothesis" (Erdős and Rényi, 1960). In 1998, Watts and Strogatz (Watts and Strogatz, 1998) showed that many types of real networks are highly clustered, unlike ER graphs, with short average path lengths relative to comparable ER graphs. This lead to the first realizations of Erdős and Rényi's "more realistic hypothesis" for networks, in the form of the *small-world hypothesis* (Watts and Strogatz, 1998) and the *preferential attachment* model (Barabási and Albert, 1999). Bibliographic databases were some of the earliest testbeds for these hypothesis. The research questions, however, were rooted in statistical mechanics, and could broadly be classified into the development of generative models to explain observed network characteristics (Newman, 2003), and the analysis of 'critical points'

in parameters governing the formation of network, at which networks exhibit sudden drastic changes in properties (Dorogovtsev et al., 2008).

The second development in the late 1990s was the wide-scale growth of the World Wide Web and the adoption of search engines. Commercial web crawlers were indexing the Web at continually increasing scales, leading to possibly the largest dynamic network dataset in existence. In 1998 and 1999, two of the most successful algorithms for ranking webpages were based on network analysis: the HITS algorithm (Kleinberg et al., 1999), and the PageRank algorithm (Brin and Page, 1998). As a result, a number of algorithmic questions dealing with either static or dynamic network came to the forefront. In addition to the ranking algorithms powering search engines, some of the focus in computer science was on algorithmic issues such as routing in decentralized networks (Kleinberg, 2000; Kumar et al., 2005), targeting influential nodes in a network for marketing or immunization (Kempe et al., 2003; Aspnes et al., 2007; Domingos and Richardson, 2001), and characterizing specific computer networks such as the Internet (Faloutsos et al., 1999), the Web (Kleinberg et al., 1999), and recently, online social networks (Kumar et al., 2006; Hu and Wang, 2009; Ahn et al., 2007).

Finally, we note that at approximately the same time in the mid 1990s, the field of data mining was maturing rapidly, and although it would not be acknowledged till later, a number of techniques developed in data mining would be applicable to dynamic network data. Mining for frequent patterns in large *transactional databases*, also known as 'market basket data', was one of the earliest problems in data mining (Agrawal and Srikant, 1994).

A transactional database records co-occurrences of items, with the standard example being a supermarket retail database that records the items in each customer's 'basket'. The database can then be thought of as a set of subsets drawn from a universal set of 'items'. Although there is generally no notion of order between the transactions in such a database, some algorithms that operate on transactional databases do require an ordering (Özden et al., 1998; Tung et al., 1999).

Dynamic networks are graphs with unique node labels, a property that can be mapped to transactional databases for certain tasks (Lahiri and Berger-Wolf, 2008; Lahiri and Berger-Wolf, 2007). Since each node within the graph of a single timestep is unique, an edge between any pair of nodes can be identified uniquely. Therefore, all nodes and edges can be uniquely mapped to the set of integers. Each timestep in the dynamic network becomes a transaction in the database, with each vertex and edge in the timestep being converted to an item. The entire dynamic network can then be treated as a transactional database, with a direct mapping between algorithmic tasks such as maximal common subgraph and set intersection (Dickinson et al., 2003; Lahiri and Berger-Wolf, 2008; Lahiri and Berger-Wolf, 2007). Note that the mapping is not valid for tasks where graph properties that have no equivalent mapping in set notation, such as connectivity, are required. However, as a benefit, frequent subgraphs can be mined using current tools for frequent itemset mining.

Similarly, sequences of symbols can be treated as discrete time series if they are scanned in a temporal direction. Although sequence data is almost ubiquitous in bioinformatics, it is also used to represent event logs, such as hardware and networks alert logs (Domeniconi

et al., 2002; Vilalta and Ma, 2002). Event logs (or to be precise, *multievent* logs) are slightly more general than sequences, because event logs can often contain more than one symbol or event at each position (Oates and Cohen, 1996; Oates et al., 1997). With this property, multievent logs can be considered equivalent to an ordered transactional database. An advantage of this mapping is that it also forms a link between dynamic networks, transactional databases, and the well-studied area of machine learning in sequences (Dietterich, 2002). Although some assumptions made for learning in sequences might not hold for dynamic networks[1], the general techniques are still be applicable to networks. This connection also allows one to capitalize, if needed, on well established algorithms for mining frequently occurring sequential patterns (Pei et al., 2004).

1.2 Summary of contributions

The following is a summary of contributions in this thesis.

1. *(Chapter 2)* A survey of how dynamic networks are measured over time to yield time series of various graph theoretic properties. Specifically, we find that almost all literature on measuring dynamic networks uses one of two aggregation methods, and a handful of simple graph theoretic measures.

   (a) When the commonly *growing network* aggregation method is used, many common trends observed in dynamic networks can be explained by a doubly stochastic

[1]In particular, the scale and dimensionality (*i.e.,* of the adjacency matrix) of dynamic networks is several orders of magnitude larger than typical multi-event sequences, and the dimensionality may not even be known in advance.

sampling process involved in the collection of data, and not as an intrinsic feature of the network itself.

(b) Using simulations on several network models, we show that networks that exhibit certain temporal properties, when subjected to even a small amount of missing temporal data, erroneously display either a contrary trend, or no trend at all.

(c) Given a dynamic network dataset, we propose a method using statistical randomization (permutation) tests to determine how likely it is that the properties of the underlying network are in flux under various assumptions.

(d) We note that measuring dynamic network datasets in the presence of noise and missing data is a difficult issue, and suggest some alternatives. If networks must be measured over time in the presence of noise and missing data, then it is important to either pick a measurement method that is less prone to biases, or to choose measures or methods that are more robust.

2. *(Chapter 3)* The development of a Fourier-like decomposition for detecting periodicity and periodic patterns in dynamic networks.

(a) We propose a new mining problem for dynamic networks that involves periodicity detection and is well grounded in theory.

(b) We prove a polynomial upper bound in the size of the input on the number of patterns in a dynamic network that can satisfy our mining criteria. This is in

contrast to related periodic pattern mining formulations that are intractable in the worst case and include redundant information in the output.

(c) We describe an efficient polynomial-time algorithm that makes a single pass over the data.

(d) We show how our algorithm extracts both a spectrum of periodicities from the network, as well as the basis patterns that comprise the spectrum.

3. *(Chapter 4)* The development of a technique for finding strong temporal correlations between edges in dynamic networks. A correlation is defined as strong if it holds a degree of predictive power on unseen data.

(a) We approach the intractability of the general problem by capitalizing on the graph-theoretic properties of real networks. Specifically, the skewed degree distribution of real networks is used to build a tractable dependency structure.

(b) We describe a problem formulation that works on a continuous time stream of interaction data, without requiring it to be quantized into discrete timesteps. We also describe an evaluation framework for our continuous-time formulation.

(c) We describe a novel Hidden Markov Model (HMM) formulation that models the time delay between any pair of edges. Using the dependency structure we described earlier, we mine pairs of edges that are best modeled by this HMM.

1.3 Datasets used in this thesis

Dynamic networks can often appear in different guises. For example, 'call graphs' are collected by telecommunications companies in real time, even though the number of customers can number in the millions (Nanavati et al., 2006). GPS and radio collars allow ecologists to tag wild animals and the social interactions between them (Juang et al., 2002; Fischhoff et al., 2007), resulting in continuous streams of proximity data. In humans, a similar effect is achieved by monitoring connections between Bluetooth-equipped cellphones (Eagle and Pentland, 2006), manually annotated photographs (Lahiri and Berger-Wolf, 2008) or the headers of email traffic (Chapanond et al., 2005; Diesner and Carley, 2005). In computer networks research, a time-series of labeled graphs representing network traffic has been used as the basis for network intrusion detection (Bunke, 2003; Bunke et al., 2005). The following is a description of the datasets that are used in various parts of this thesis.

1. The **Enron email network** is inferred from the mailboxes of about 150 employees of the former Enron Corporation (Shetty and Adibi, 2004). The contents of the mailboxes were publicly released by the Federal Energy Regulatory Commission in the course of investigations into the workings of the company, and consist of the full text and headers of emails, both sent and received. This allows us to construct a partial, dynamic view of email communications within a large, complex organization like Enron, and has spurred much research in various areas (Diesner and Carley, 2005; Chapanond et al., 2005; Shetty and Adibi, 2004). Each vertex represents an email address, with a directed edge from the sender of an email to all its recipients.

The timestep quantization is one day, although arbitrarily smaller quantizations are also possible due to the presence of full email header information.

2. The **Plains Zebra** and **Grevys Zebra** networks are observations of social associations in two species of wild Zebra in Kenya (Fischhoff et al., 2007; Sundaresan et al., 2007). They are currently collected by direct visual observations made by ecologists, although more advanced and accurate methodologies like radio and GPS collars are being investigated (Juang et al., 2002). The manual collection of interaction data results in missing data, but for Grevys Zebra, the missing data rate is estimated to be under 50%. Both species of Zebra are *fission-fusion* species, which means that they come together in groups that subsequently dissolve to form new groupings. An analysis of dynamic communities has confirmed differences in the grouping habits of the two species (Tantipathananandh et al., 2007). Each vertex represents an individual Zebra, identified by the pattern of stripes on specific parts of its body, and an edge represents a social association as determined by ecologists. The timestep quantization is one day, which corresponds to the frequency of the ecologists' observation rounds.

3. The **IMDB Photo Network** is collected from metadata about people tagged in photographs on the Internet Movie Database (IMDB) (Lahiri and Berger-Wolf, 2008). The photographs are generally of actors, musicians, directors and other people associated with the entertainment industry, and may either be candid or professional shots. Since IMDB is not a general photo-sharing site and its pictures are labeled by staff members, one might reasonably assume that the people tagged in photos are

'celebrities' of some sort and that a degree of social association exists between them. This methodology is quite similar to the Zebra sighting datasets, and the observed structure is a partial view of the true set of interactions. Metadata on a total of 194,430 photographs were collected, with about 75,000 photographs containing more than one person. Each vertex corresponds to a manually identified and disambiguated person (conducted either by IMDB or professional photo agency staff), with an edge representing co-appearance in a photograph. The discretization timestep is one day.

4. **Reality Mining** was an experiment conducted at MIT to collect a variety of data related to movements and social dynamics in humans (specifically, students at MIT) (Eagle and Pentland, 2006). It involved equipping volunteers with cellphones augmented with special tracking software. Among the recorded data was physical proximity data inferred from two cellphones being able to establish a direct Bluetooth connection, with the maximum range for such a connection being approximately 30 ft. (Bluetooth SIG, Inc., 2009) Each vertex represents a study participant with a Bluetooth-equipped cellphone, and an edge represents physical proximity of less than 30 ft. The quantization timestep is four hours.

5. **Call Detail Records-C** is a Call Detail Record (CDR) dataset that is collected whenever a phone subscriber makes a call to another telephone number. In the process, information such as the originating and destination number (encrypted), and the time and date of the call are logged. We obtained 4 months of CDRs from mobile phone subscribers in a dense urban area. For this dataset, we only considered subscribers

that made at least 3 phone calls per day, and included *all* successful phone calls made or received during the observation period. We treat a phone call between two subscribers as an undirected edge, because phone conversations, as opposed to phone calls, are inherently bi-directional. The number of nodes in the network and the number of phone calls recorded are on the order of 10^5 and 10^7 respectively. We are unable to disclose further details due to privacy considerations.

6. **Call Detail Records-J** is similar to the CDR-C dataset, but includes CDRs from all phone users of a large geographical region (an entire state) of a particular country, for a period of 5 months, without any sampling bias as in CDR-C. It is also unlikely that there is significant overlap between the customers in CDR-J and CDR-C due to the geographical separation between the regions. The number of nodes and interactions are on the order of 10^6 and 10^7 respectively.

1.4 Declaration of prior published work

Parts of Chapters 3 and 4 appear in the following publications:

1. M. Lahiri and T.Y. Berger-Wolf. *Periodic subgraph mining in dynamic networks.* Knowledge and Information Systems, Volume 24, Issue 3 (2010), p. 467.
2. M. Lahiri and T.Y. Berger-Wolf. Mining Periodic Behavior in Dynamic Social Networks. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, Pisa, Italy. December 2008.
3. M. Lahiri and T.Y. Berger-Wolf. Structure Prediction in Temporal Networks using Frequent Subgraphs. In *Proceedings of the IEEE Computational Intelligence and Data Mining Conference (CIDM 2007)*, Honolulu, Hawaii. April 2007.

CHAPTER 2

STRUCTURAL PROPERTIES OF NETWORKS OVER TIME

A fundamental question in dynamic network analysis is to determine how the graph-theoretic properties of a network are changing over time when the network is not completely visible. A number of diverse questions depend on our ability to accurately measure structural changes in real networks: for example, is the average number of hops between Internet routers increasing or decreasing as the Internet grows, and does the pattern of collaborations between publishing scientists indicate that scientists are forming an increasingly tighter-connected social network? These issues are important in computer network analysis as well as physics and sociology, and in general, any domain where a physical system can be represented as a partially observable *graph structured process* that generates a changing network over time. In these cases, the change in the structure of the physical system between any two time points is quantified by the change in some graph-theoretic measure evaluated at those points.

There are at least two reasons to look for trends in the time evolution of various graph measures.

- *Generative network models.* Since the true dynamical processes that generate real dynamic network datasets are too complex to model, a line of research aims to develop simple dynamical processes that can approximate the true process by generat-

ing networks with a given trend over time in some graph measure (Leskovec et al., 2007; Akoglu et al., 2008; Akoglu and Faloutsos, 2009; Bonato et al., 2009; Chakrabarti et al., 2010; Du et al., 2010). These dynamical processes can very generally be viewed as probabilistic algorithms driven by uniform random noise that convert one graph into another. Inferring the parameters of these processes from data is generally intractable, so a common method of justifying various dynamical graph generation models is to show that they reproduce temporal trends in measures comparable to an average temporal trend found in multiple real datasets. Naturally, this presumes that the temporal trends measured from empirical data were accurate and not artifacts of the data sampling process, an issue that we show is not trivial.

- *Scientific datasets.* In scientific datasets, a sudden fundamental change in the structure of a network can sometimes be detected by tracking a correspondingly large change in an appropriate graph measure. In animal association networks from ecology, for example, differences in network measures are used to determine behavioral differences between species (Sundaresan et al., 2007).

In an ideal world, we would simply obtain a sequence of snapshots of a graph structured process over time and periodically measure properties of interest in order to determine a trend. In the real world, however, our observational capabilities are severely limited in terms of how far back in time datasets reach, how complete each snapshot is, and the mechanisms by which network data is collected. In this chapter, we examine how these issues are tackled by surveying the growing literature on dynamic network measurements. Several

recent empirical studies suggest that the structures of a variety of otherwise disparate real networks are evolving in similar ways (Leskovec et al., 2005; Ahn et al., 2007; Menezes et al., 2009). We describe in detail the experimental methodology used to reach these conclusions, and investigate how sensitive they are to various kinds of missing data.

This chapter is organized as follows. In the next section, we present a methodology survey describing typical procedures for measuring graphical properties of a network over time, as well as a characterization of dynamic network datasets into two broad classes: *interaction* networks, and *citation* networks, depending on the nature and permanence of interactions (edges). Section 2.2 presents a literature review and summary of published empirical results on the graphical properties of real dynamic networks over time. All the published results on networks that we survey can be classified as either interaction or citation networks. In order to assess the significance of the many common temporal trends in network properties (*e.g.,* , decreasing average shortest path length over time), we experimentally analyze whether these trends necessarily reflect the reality of the underlying system in the presence of missing data and observational limitations in the data collection process. Section 2.3 deals with citation networks, and Section 2.4 deals with the more common class of interaction networks. In Section 2.5, we summarize our findings and list some suggestions for future research.

2.1 Networks, measurement, and error

Dynamic networks are explicit representations of the change in the structure of a physical system over time. These systems are generally complex real-world phenomena that can be

modeled as a set of uniquely identifiable entities interacting with each other over time. The entities can act independently of each other, and the interactions between them can occur in arbitrary ways. The global structure of which entities have interacted with which other entities can therefore be a complex network-like structure that (presumably) evolves over time. Quantifying the change in the structure of such a system over time is the first step in analyzing a dynamic network.

The most immediate way of quantifying the change in a dynamic network between two time points is to simply measure its structure at both time points and compute the difference. There are numerous ways to measure a graph, but in a very general sense that covers the most popular methods, a measure M can be thought of as a function from a graph to a real number.

$$M : G \mapsto \mathbb{R} \quad \text{where } G = (V, E)$$

Some simple measures include the number of nodes and edges in a graph, the average shortest path length between all pairs of connected nodes, and the largest eigenvalue of the adjacency matrix. These are *static* measures that operate on a single graph; an example of a measure that operates on a graph through time is the graph edit distance measure between consecutive graphs used in anomaly detection in computer networks (Bunke, 2000). Given a dynamic network dataset, one could simply construct a graph of all interactions up to a given time point, measure the graph, and subsequently produce a time series of measure values by repeating the process, which would then presumably characterize the change in the structure of the system.

A number of assumptions need to hold before this is true. Clearly, the measure M that is used must adequately represent true change in the physical system, at least with respect to the reason for measuring the dynamic network. For example, if we want to quantify the change in a computer network to study the effect of decentralized routing protocols, the average shortest path length might be a good measure, whereas the raw number of nodes might not be. Secondly, if there are sampling and incomplete data issues in the dataset (as is the case with almost all empirical time-series datasets), then the chosen measure must be robust to sampling errors. In particular, if we are to measure a dynamic network at successive time points in the presence of sampling error, then the resultant time series of measure values should be similar to the true underlying time series, at least qualitatively.

There also exist two common methods for aggregating dynamic network data into a single graph that can be measured. For example, given a dynamic network dataset, we might have identified two time points at which we wish to measure the structure of the underlying system. Both aggregation methods can be applied to the same data under certain conditions, and yield very different results because they make different assumptions about the data. Adding to the subtleties of choosing an aggregation method, we have also identified two different classes of network data that require different treatment. This distinction between aggregation methods and network data classes does not appear to have been explicitly made, particularly in terms of measurement biases involved in mismatching aggregation methods, graph measures, and network data classes. We start this analysis

with a description of network data classes in the next subsection, and network aggregation methods in Section 2.1.2.

2.1.1 Network data classes

All the dynamic network datasets described in the literature that we will later review in Section 2.2 fall into one of two categories depending on the type of physical system being modeled: *interaction* networks and *citation* networks. The overall characteristics of each class are as follows.

- *(Interaction networks)* An edge, identified by the labels of its adjacent nodes, can appear at multiple timesteps. Examples include email networks built from email headers (Leskovec et al., 2005; Diesner and Carley, 2005), where people continually send emails to each other over time, and similar networks built from phone records (Nanavati et al., 2006), logs from physical proximity sensors (Eubank et al., 2004; Eagle and Pentland, 2006), and co-authorship of published scientific articles (Newman, 2001b; Barabási et al., 2002; Leskovec et al., 2005). Each edge in the dynamic network dataset is therefore a record of a single instance of an interaction between nodes, out of many other possible instances at different times. Network structure is determined through the proxy of interactions, which can be regulated by an independent dynamic process.
- *(Citation networks)* Unlike interaction networks, citation networks only grow in size over time with the addition of new nodes and edges, and each edge appears only once in the observation stream. Thus, in a dynamic network dataset, the appearance of an

edge at time t implies that the underlying physical system grew by at least one edge at time t. Their name comes from the canonical example of the directed network of citations between scientific papers; once a scientific paper is published, its citations never change, and thus each edge in the dynamic network appears only once over all of time (Bilke and Peterson, 2001; Nerur et al., 2005; Leskovec et al., 2007). Note that interaction networks where edges and nodes are deleted very infrequently relative to growth can also be seen as citation networks at short times. This covers some types of online social network data where the rate of node and edge addition far outstrips its removal, to the point where the online 'friendship' networks can be seen as citation networks (Kumar et al., 2006; Krishnamurthy et al., 2008). For the same reason, it also includes domains such as blog inter-linking networks (Adar and Adamic, 2005).

The distinction between the two classes is important when we use network measures to represent the change in the underlying physical system. In particular, when a citation network dataset contains a record of edge (u, v) at time t, we know that the underlying physical system has increased in size by at least one edge. However, with interaction networks, particularly when we do not have the entire temporal history of the network, we cannot tell if the record of edge (u, v) at time t represents true growth in the underlying system, or is just the re-occurrence of an interaction that had also occurred some time before. In other words, there is a process that generates interactions along a growing network structure, and using interactions as a proxy for network structure requires that we account for the sampling error induced by the dynamics of the interactions, particularly

at short times and without full temporal history. This is closely tied to the aggregation method (or lack thereof) used to assemble interaction data into a time series of graphs, which is the topic of the next subsection.

2.1.2 Network aggregation methods

We can view *citation* and *interaction* network data types through the lens of a sampling process (generally involved in data collection) on a true underlying network that is growing over time. The sampling process generates a sequence of graphs drawn from the true underlying process, which we call the *observations*. In the case of interaction networks, the sampling process reveals a set of interactions each time a group of entities interacts. In the case of citation networks, observations consist solely of (subsets of) new edges and vertices that are added to the network, with the constraint that the same edge must not appear twice. These samples are generally revealed over time, starting from an arbitrary time relative to the underlying process, from which we must infer both an initial structure as well as any change in it.

The earliest approach to network data was to treat the underlying process as being in a *steady state* with respect to some graph measure M.

Definition 2.1.1. *(Steady state)* A physical system is in equilibrium or a *steady state* with respect to a graph theoretic measure M if the partial derivative of M measured on the system with respect to time is zero.

$$\frac{\partial M}{\partial t} = 0$$

Vertices and edges may therefore be added to or deleted from the system, but the dynamic behavior of a measure of interest M is independent of time.[1] If we assume that an underlying system is in a steady state with respect to measure M, and assuming a consistent sampling process, we can simply let the observation sequence converge to a large enough graph and assume that M measured on this graph is representative of the true value of M. These steady state and consistency assumptions are the foundation of *static network analysis*, which approximates partially stationary dynamic processes as large networks. Static network analysis has produced a large number of important advances in network theory, such as the observations that real-world networks tend to have heavily skewed degree distributions and low average pairwise shortest path lengths (Albert and Barabási, 2002; Newman, 2003; Boccaletti et al., 2006). A number of studies also analyze dynamical aspects of the sampling process, such as how long typical systems must be observed till various graph theoretic properties converge to limiting values (Latapy and Magnien, 2006), and how sensitive measured properties are to different types of sampling error (Costenbader and Valente, 2003; Kossinets, 2006).

There has also been interest in the dynamic behavior of some measure M of the underlying system, with the implicit assumption that the underlying system is not in a steady state with respect to M (Barabási et al., 2002; Leskovec et al., 2005). Intuitively, the steady

---

[1]For example, a growing k-regular graph, one one in which every vertex has exactly k adjacent edges, is in equilibrium with respect to its degree distribution or any statistic of it, but not necessarily with respect to other measures.

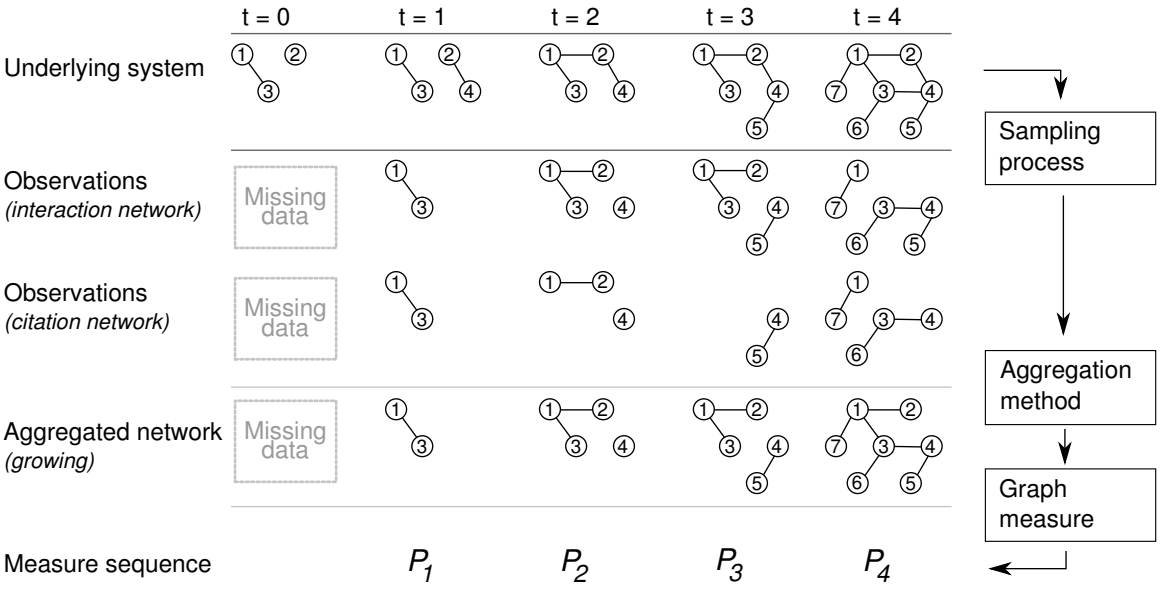


Figure 3: Example of a true underlying network, a sequence of observations of it starting at $t = 1$, and growing and dynamic interaction representations of the observations. Edge thickness represents edge weight.

state assumption might not appear to be appropriate for systems as large and chaotic as the Internet (for example). In this case, observations are aggregated over time, either as a *growing* network or a *fully dynamic network*, into a single graph that is measured with M at fixed time intervals. Figure 3 depicts how a sampling process on the true underlying network generates observations, depending on whether the underlying network is a citation or interaction network. These samples of network structure are aggregated using the growing network method to yield a time series of graphs with measure values $P_1, ..., P_n$. We describe each aggregation method in more detail below.

Definition 2.1.2. *(Growing network)* Observations of the physical system are made at discrete timesteps. Each observation is accumulated into a single growing graph, which is measured at fixed time intervals. Aggregating in this manner implicitly assumes that the underlying process can be represented as a citation network. Let $\mathbf{G}$ be the true, underlying network of the physical system being modeled, which is only partially observable and growing over time with the addition of new vertices and edges (by definition, vertices and edges are never removed in a citation network). Since $\mathbf{G}$ is growing over time, we let its structure at time t be denoted by an element in the sequence $\langle \mathbf{G} \rangle$.

$$\langle \mathbf{G} \rangle = \langle \mathbf{G}_0, ... \rangle \quad \text{where } \mathbf{G}_t = (\mathbf{V_t}, \mathbf{E_t})$$

where $\mathbf{G}_t$ is the *complete* network structure at time t, consisting of a set of labeled vertices $\mathbf{V_t} = \{\mathbf{v}_i : \mathbf{v}_i \in \mathbb{N}\}$ and a set of directed or undirected edges $\mathbf{E_t} = \{(\mathbf{v}_i, \mathbf{v}_j) : \mathbf{v}_i, \mathbf{v}_j \in \mathbf{V}_t\}$. Since vertices and edges are only added to the system, we have a monotone property on the vertex and edge sets over time.

$$\mathbf{V_t} \subseteq \mathbf{V_{t+1}} \tag{2.1}$$

$$\mathbf{E_t} \subseteq \mathbf{E_{t+1}}$$

Since the true underlying network is only partially observable, we instead obtain finite samples of its graph structure over time, called the observation sequence $\langle O \rangle$, starting at some arbitrary time $t_0 > 0$.

$$\langle O \rangle = \langle G_{t_0}, ..., G_{t_n} \rangle \quad \text{where } G_{t_i} = (V_{t_i}, E_{t_i}),\ V_{t_i} \subseteq \mathbf{V}_{t_i}, E_{t_i} \subseteq \mathbf{E}_{t_i}$$

Due to limitations of the observation process, each observed graph G_t may be a small fraction of the size of the physical system at that point in time $\mathbf{G_t}$, *i.e.,* $|V_t| \ll |\mathbf{V_t}|$ and $|E_t| \ll |\mathbf{E_t}|$. The growing network aggregation method approximates $\mathbf{G_t}$ by aggregating all samples from the start of observations t_0 up to the current timestep into a single growing network $\langle G^+ \rangle$:

$$\langle G^+ \rangle = \langle G^+_{t_0}, ..., G^+_{t_n} \rangle \quad \text{where } G^+_t = (\bigcup_{i=t_0}^{t_n} V_i, \bigcup_{i=t_0}^{t_n} E_i)$$

Any graph measure M can then be measured on G^+_t at fixed intervals, yielding a time-series that (presumably) approximates the trend of the same property on the underlying network $\langle \mathbf{G} \rangle$.

A key assumption in the growing network methodology in practice is that the aggregated growing network at any point in time is a good approximation of the underlying physical system, including at the (necessarily) arbitrarily chosen time of the first observation t_0. Particularly, the assumption that we can accurately track trends in the underlying process, either quantitatively or qualitatively, by measuring an aggregated graph of observations

periodically requires that the measured trend accurately track the trend in the underlying system.

Definition 2.1.3. *(Fully dynamic network)* A fully dynamic network approach treats the underlying system as an interaction network. Observations are aggregated within successive, non-overlapping intervals of time, and the graph obtained from each interval is measured independently of the other intervals. When the interval of time is fixed, it is called the *window* of the aggregation. It can be seen as a growing network where vertices and edges are removed after a fixed period of time (the window) unless they appear again within the window. Thus, depending on the relative rates of vertex and edge addition and removal, a dynamic network may be structurally growing, in equilibrium, or shrinking. The underlying network $\langle \mathbf{G} \rangle$ no longer necessarily has the monotonicity property of growing networks (Equation Equation 2.1).

A variant of this approach weights each edge with a time-decaying function, to emphasize the impact of more recent changes in graph structure. This is usually practiced in one of two related ways:

1. Weight each edge by the amount of time that has elapsed since it was last seen, and use algorithms that take the weight of each edge into account (*e.g.,* (Sharan and Neville, 2007; Acar et al., 2009; Barrat et al., 2004)).
2. Remove edges from the network after a certain period of time if they have not appeared again, where the amount of time is determined by a decay function (*e.g.,* (Kossinets and Watts, 2006)).

The issue of time-decaying relationships in social networks has also received attention in sociology, such as the formulation of sociologically meaningful decay functions suggested by data (Burt, 2000).

A practical issue with the fully dynamic aggregation method is that it presumes knowledge of an appropriate window length to aggregate observations over. In practice, this is usually determined by taking some 'natural' quantization of the dataset, such as a window length of a year for scientific publication data, but this is difficult to determine for some datasets (*e.g.,* animal association data). Furthermore, some measures M might be sensitive to the window length, in effect showing trends that are a function of the quantization window. However, more rigorous methods are needed to determine a good aggregation window for a dataset, and a complementary set of approaches attempts to find a time aggregation that minimizes some notion of error in measured dynamic trends (Sun et al., 2007; Sulo et al., 2010). The growing network method has the practical advantage of not requiring any additional parameters, and we focus more on this method for the remainder of this chapter.

To illustrate the different situations that might call for growing networks over dynamic networks, consider the analysis of e-mail transmission records to deduce the structure of the underlying association network (Shetty and Adibi, 2005; Kossinets and Watts, 2006; Leskovec et al., 2007). Treating each day as a timestep, the observation corresponds to the graph structure of e-mails that are sent and received on a given day. This has definite meaning, as we can investigate the time stream for periodic or other temporal patterns (Lahiri and Berger-Wolf, 2008; Lahiri and Berger-Wolf, 2010; Sun et al., 2007; Acar et

System	*Static*	*Growing*	*Fully dynamic*
Co-auth.	(Newman, 2001b)	(Barabási et al., 2002)	(Moody, 2004)
E-mail	(Ebel et al., 2002)	(Leskovec et al., 2007)	(Kossinets and Watts, 2006)[a]
Online	(Mislove et al., 2007)	(Holme et al., 2004)	(Hu and Wang, 2009)

[a] Time-weighted dynamic network.

TABLE I: EXAMPLES OF PHYSICAL SYSTEMS PAIRED WITH DIFFERENT NETWORK MODELS.

al., 2009). On the other hand, consider the discovery of Autonomous Systems (AS) routes on the Internet through the use of `traceroute` probes and similar methods (Andersen et al., 2002; Chen et al., 2002; Vázquez et al., 2002; Leskovec et al., 2007). The temporal sequence of samples is a product of the observation technique and has no inherent meaning to the object of interest (the AS topology). Graph theoretic analysis of the sampled graph would be more meaningful than, for example, looking for periodic patterns. However, since AS routes are frequently deleted in addition to being added, growing networks might not be the best representation.

Table Table I illustrates how the same physical system can be represented as different types of networks, often starting with the same initial data.

2.1.3 Network measures

A number of graph theoretic measures are used to characterize the structure networks. In this subsection, we survey the measures that are commonly used for characterizing the graphical properties of networks over time. We describe these measures to demonstrate

that they are non-trivial and diverse, and to motivate the form of randomized sensitivity analysis that we propose in later sections, which is agnostic to the chosen measure.

In addition to classical graph theoretic measures like the average and maximum shortest path lengths between vertices, sociologists have developed a variety of *vertex centrality* measures to assess the importance of individual vertices within the larger network (Wasserman and Faust, 1994), physicists and graph theoreticians have proposed looking at *distributions* of fundamental properties (Newman, 2003; Albert and Barabási, 2002; Erdös and Rényi, 1959), and computer science has seen the wide-scale adoption of ranking techniques like PageRank (Brin and Page, 1998; Langville et al., 2008) and tensor factorization (Sun et al., 2007; Acar et al., 2009) for analysis, and algorithmic questions posed on real networks, such as routing (Kleinberg, 2000).

We focus on three basic categories of network properties that have been widely used to characterize networks that change over time: (1) *connectivity*, in terms of statistics of the distribution of shortest path lengths between all pairs of vertices, (2) the *density* of the network, both local and global, in terms of the relative numbers of vertices and edges, and (3) *spectral* properties of the graph's adjacency matrix or transformations of it.

2.1.3.1 Connectivity

The *distance* between two vertices u and v in a graph is generally the length of the shortest path between the vertices, *i.e.,* the minimum number of edges that need to be traversed to connect u to v. Let σ_{uv} be the length of the shortest path between vertices u

and v, where $\sigma_{uv} = \infty$ if there is no path between u and v (note that σ_{uv} and σ_{vu} are not necessarily equal for directed networks).

Definition 2.1.4. The *connectivity* of a network can be expressed in terms of summary statistics of the pairwise shortest path length distribution, *i.e.,* the distribution of σ_{uv} for all distinct, ordered vertex pairs in a directed network, and for all distinct unordered vertex pairs in an undirected network. In the following definitions, let $k = 1$ for directed networks and $k = 2$ for undirected networks.

$$\bar{l} = \frac{k}{V(V-1)} \sum_{u,v \in V} \sigma_{uv} \qquad \text{(Average path length (West, 2001))}$$

$$\bar{l}^{-1} = \frac{k}{V(V-1)} \sum_{u,v \in V} \frac{1}{\sigma_{uv}} \qquad \text{(Efficiency (Latora and Marchiori, 2001))}$$

$$d_{\max} = \max_{u,v \in V} \sigma_{uv} \qquad \text{(Diameter (West, 2001))}$$

$$d_{90} \approx \arg_i [P(\sigma_{uv} \leq i) = 0.9] \qquad \text{(Effective diameter (Leskovec et al., 2005))}$$

The distribution of all-pairs shortest path lengths between vertices is only well defined for (strongly) connected graphs, since $\sigma_{uv} = \infty$ when there is no path between u and v. In order to handle disconnected graphs, the average path length distribution is generally only computed over pairs of vertices that are reachable from each other. The *efficiency* measure introduced by Latora and Marchiori (Latora and Marchiori, 2001) is another way to overcome this problem by considering the inverse of the shortest path length, so that

when $\sigma_{uv} = \infty$, then $l_{uv}^{-1} = 0$. A third method is to only compute shortest paths between vertices in the largest (strongly) connected component.

The *diameter* of a graph, although unambiguously defined in graph theory (Bollobás, 1998; West, 2001), has a slightly different meaning in other disciplines, particularly physics. For example, in their seminal paper on estimating the 'diameter' of the World Wide Web, Albert *et al.* (Albert et al., 1999) are referring to the average shortest path length and not the maximum shortest path length, as is standard in graph theory. In computer science, Leskovec *et al.* (Leskovec et al., 2005; Leskovec et al., 2007) and subsequently Ahn *et al.* (Ahn et al., 2007) use a smoothed notion of the conventional graph-theoretic definition of diameter – the 90^{th} percentile of the shortest path length distribution – to estimate the length of the 'almost' longest shortest path length in a network. The distinction between these properties is particularly important when considering trends over time, since it is possible to construct examples where, say, the average path length is increasing while the diameter or effective diameter is decreasing.[1]

2.1.3.2 Density

Irrespective of the structure of a graph, it is sometimes useful to quantify the relative numbers of vertices and edges to get some sense of the 'crowdedness' of the network, either at a global or a local scale. Although density measures are trivial to compute, the ratio of nodes to edges can be particularly informative of structure of the network. For example, a classic

---

[1] As a real world example of such a case, see the early stages of the evolution of the *Cyworld* network (Ahn et al., 2007).

paper by Erdös and Rényishowed that the connectivity properties of random graphs undergo an abrupt phase change at certain critical densities that can be analytically computed (Erdös and Rényi, 1959).

Definition 2.1.5. The density of edges in a network $G = (V, E)$ can be quantified in the following ways. In the following definitions, let $N(v)$ be the set of neighbors of vertex v, *i.e.,* the set of vertices that are adjacent to vertex u, and let $k = 1$ for directed networks and $k = 2$ for undirected networks.

$$D = \frac{k|E|}{|V|(|V|-1)} \qquad \text{(Density (Bollobás, 1998; West, 2001))}$$

$$\overline{N} = \frac{1}{|V|}\sum_{v \in V}|N(v)| \qquad \text{(Average degree)}$$

$$= \frac{D}{|V|-1}$$

$$\overline{CC_1} = \frac{1}{|V|}\sum_{v \in V}\frac{k|M(v)|}{|N(v)|(|N(v)|-1)} \qquad \text{(Clust. coefficient (Watts and Strogatz, 1998))}$$

where $M(v)$ is the set of edges between the neighbors of v:

$$M(v) = \{(i, j) : i, j \in N(v) \text{ and } (i, j) \in E\}$$

and $\overline{CC_1}$ is only computed for vertices with $N(v) > 1$.

The *clustering coefficient* measure proposed by Watts and Strogatz (Watts and Strogatz, 1998) is a measure of the local density of edges around a particular vertex. The clustering coefficient of the whole network is usually expressed as the mean clustering coefficient

computed over all vertices. Soffer and Vazquez (Soffer and Vázquez, 2005) show that the clustering coefficient of a vertex, as defined above, is inherently correlated with the degree of the vertex, since high-degree vertices will have more possible edges between them, and a generally lower clustering coefficient. They propose an alternative definition that removes this bias. However, to the best of our knowledge, it has not been widely used.

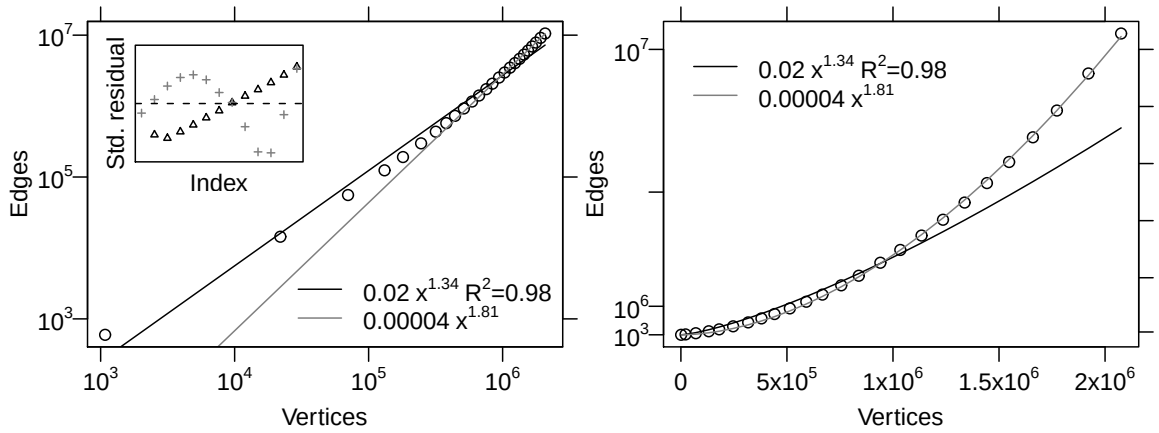
Finally, a property that has attracted considerable interest in computer science is the *Densification Power Law* (DPL), first described in a pair of seminal papers (Leskovec et al., 2005; Leskovec et al., 2007), examined in a number of subsequent studies (Shi et al., 2007; Menezes et al., 2009; Latapy and Magnien, 2008; Pallis et al., 2009), and even used as the basis of a network sampling algorithm (Leskovec and Faloutsos, 2006).

Definition 2.1.6. An evolving network that obeys a *Densification Power Law* (DPL) contains a number of edges E that, at any point in time, is related to the number of vertices V in a power function of the form:

$$E(t) = k \cdot V(t)^{\alpha} \tag{2.2}$$

where $1 < \alpha < 2$ is called the *densification exponent*.

The DPL is a statement about the relative number of vertices and edges over time. Note that the area of fitting power-law *distributions* to the degrees of vertices measured in a dataset has been well studied (Clauset et al., 2009; Goldstein et al., 2004), but in the case of the DPL, we are not dealing directly with a statistical distribution, but rather a power



(a) Log-log scale

(b) Linear scale

Figure 4: DPL plot from a patent citation dataset on doubly logarithmic and linear scales, illustrating the difference between regression assumptions. The black line is a fit of the *Multiplicative* DPL equation, and the gray line is a fit of the *Additive* DPL equation. The inset shows standardized residuals of both fits against a $y = 0$ line, revealing errors that are unlikely to be independent.

function of the number of edges to the number of nodes that is invariant over time. A practical issue, however, is that there are at least two ways to fit a power function to empirical data, each of which requires different assumptions about the underlying relationship, and each of which leads to a different statistical formalization of Definition 2.1.6:

1. *Linear regression on log-transformed data.* This is the regression method that has commonly been used in the literature, including in the original definition (Leskovec

et al., 2005). The DPL function is assumed to have a multiplicative, centered, *i.i.d.*, normally distributed error term ϵ (Seber and Lee, 2003), specifically:

$$E(t) = k \cdot V(t)^{\alpha} \cdot \epsilon$$

2. *Nonlinear regression.* This method assumes a more common additive error term, and requires the use of an algorithm like nonlinear least-squares (Ryan, 2008):

$$E(t) = k \cdot V(t)^{\alpha} + \epsilon$$

Figure 4 illustrates this distinction on a dataset of U.S. patent citations (Hall et al., 2001) (see (Leskovec et al., 2005) for details on preprocessing). The densification exponent of $\alpha = 1.34$ obtained using linear regression is significantly different from the nonlinear regression fit of $\alpha = 1.81$, with the magnitude of the standardized residuals shown in the inset (*i.e.,* differences between the data and fitted curve, centered around the mean difference and divided by the standard deviation in differences). Later in this chapter, we show that each form of DPL exhibits different sensitivities to the same missing data, making it important to qualify the regression technique (and thus statistical noise model) used.

2.1.3.3 Spectral properties

Spectral graph theory uses the algebraic properties of a graph's adjacency matrix, or transformations of it, to reveal an astonishing amount of information about the structure of the graph (Biggs, 1993; Chung, 1997). A number of diverse questions can be answered

by examining either the set of eigenvalues of the graph's (possibly transformed) adjacency matrix, or specific eigenvectors associated with them. For example, the eigenvector corresponding to the largest eigenvalue is the basis of a number of vertex importance ranking algorithms (Bonacich and Lloyd, 2001; Perra and Fortunato, 2008). The popular PageRank algorithm can be seen as a variant of this method that uses a transformation of the adjacency matrix (Brin and Page, 1998). We briefly describe some interesting spectral graph properties, focusing on eigenvalues as a global characterization of the graph rather than on eigenvectors as characterizations of individuals vertices.

Definition 2.1.7. The *spectrum* of a matrix is the set of its eigenvalues. The spectral properties of a graph $G = (V, E)$ depend upon which matrix representation is chosen for it. Starting with the basic adjacency matrix of a graph, the *combinatorial Laplacian matrix* is defined as follows, where $N(v)$ is the degree of vertex $v \in V$:

$$L_{uv} = \begin{cases} N(v) & : u = v \\ -1 & : (u, v) \in E \\ 0 & : \text{otherwise} \end{cases}$$

We denote $\lambda_1 \leq ... \leq \lambda_V$ as the ordered set of eigenvalues of an adjacency matrix A, where V is the number of vertices in the graph and λ_V is the *principal eigenvalue*. Similarly, the spectrum of the Laplacian matrix is denoted $\mu_1 \leq ... \leq \mu_V$.

The following are some properties of the spectra of each matrix representation of a graph.

1. *(Adjacency)* The diameter of the graph $d_{\max}$ is less than the number of distinct eigenvalues (West, 2001).

2. *(Adjacency)* For processes spreading on graphs, such as viruses in computer networks, the *epidemic threshold* is a critical rate of infection beyond which a virus can become an epidemic. The epidemic threshold can be estimated by the quantity $1/\lambda_V$ (Wang et al., 2003).

3. *(Adjacency)* In a scale-free graph, the maximum degree grows as $\sqrt{V}$, so the principal eigenvalue can be expected to grow as $\lambda_V \sim V^{1/4}$ (Farkas et al., 2001).

4. *(Laplacian)* The number of zero eigenvalues is equal to the number of connected components.

5. *(Laplacian)* The largest eigenvalue is bounded by twice the maximum degree (Almendral and Díaz-Guilera, 2007):

$$\mu_V \leq 2N_{\max}$$

6. *(Laplacian)* The diameter of an undirected graph $d_{\max}$ is bounded by the following expression involving the largest and second-smallest eigenvalue μ_2 (Chung et al., 1994):

$$d_{\max} \leq \left\lfloor \frac{\cosh^{-1}(V-1)}{\cosh^{-1}(\frac{\mu_V+\mu_2}{\mu_V-\mu_2})} \right\rfloor + 1,$$

Spectral properties have rarely been use to characterize the time evolution of networks, perhaps because interpreting temporal trends in eigenvalues is less intuitive than inter-

preting trends in classical graph theoretic properties like density and diameter. Plotting the spectrum of a graph can be used as an effective tool to summarize the structure of the graph (Banerjee and Jost, 2009), but robust and intuitive methods for tracking this summary over time are an open problem.

2.2 Dynamic properties of real networks

In the previous section, we outlined network data classes, aggregation methods, and graph measures used to characterize network structure. Using this framework, we now survey the literature to summarize commonly reported temporal properties of real dynamic network datasets. We also aim to classify the experimental methodology of each study into the conceptual framework described in the previous section, in an attempt to characterize what the *de-facto* experimental procedure is. Reported empirical results are categorized by graph-theoretic properties and the type of network dataset being studied, and then presented in increasing chronological order of year of publication. Section 2.2.1 describes the literature in terms of global and local density measures in the graph, whereas Section 2.2.2 describes distance measurements in terms of the distribution of shortest paths over time. Due to the ordering of this section, the networks of each study are described in most detail in Section 2.2.1.

2.2.1 Global and local density

We first examine three measures related to the global and local density of a network: the conventional definition of density D (the ratio of the number of edges to the maximum possible number of edges), the average degree $\overline{N}$, and the average clustering coefficient $\overline{CC_1}$.

Of these, $\overline{CC_1}$ measures the local density of edges centered at each vertex in the network, on average, as specified in Definition 2.1.5.

2.2.1.1 Bibliographic networks

Barabási *et al.* (Barabási et al., 2002) analyzed the time evolution of two co-authorship datasets consisting of publications in Neuroscience and Mathematics. They found that the average degree $\overline{N}$ in both datasets are monotonically increasing over time, although the rate of increase is faster in Neuroscience than in Mathematics. They speculate that this is because of the differences in collaboration culture between the fields. The average clustering coefficient is also decreasing in both datasets, although this could possibly be explained by the degree-correlation bias in the $\overline{CC}$ measure (Soffer and Vázquez, 2005).

Elmacioglu and Lee (Elmacioglu and Lee, 2005) presented an analysis of co-authorship in the database research community by aggregating co-authorship information from 100 "hand-picked" publication venues listed in the DBLP database (Ley, 2002). We obtained two trends related to density from their analysis. The first is the average degree over time considering the DBLP database as a dynamic network, expressed as the average number of collaborators per author each year. This trend is shown to be increasing, and is a lower bound on the trend in a growing version of the same dataset. The second trend in the paper is a slow, linear increase in the clustering coefficient over time computed in the growing (cumulative) version of the dataset. The apparent disagreement with the trend found by Barabási *et al.* in other co-authorship datasets could possibly be explained by the fact that Elmacioglu and Lee only compute the clustering coefficient for vertices in the

largest connected component of the network, whereas Barabási *et al.* consider all vertices in the network. At the end of the observation period, less than 60% of all vertices in Elmacioglu and Lee's dataset are included in the giant component.

Menezes *et al.* (Menezes et al., 2009) analyzed the structure and evolution of research collaborations in computer science from co-authorship data. They manually selected academic institutions from three geographical regions – 16 in the United States and Canada (Ca-US), 6 in Europe (Fr-Sw-UK), and 8 in Brazil (Br) – and extracted names of faculty from departmental homepages, and co-authorship data for those faculty from DBLP. In the Brazil and Europe networks, the clustering coefficient appears to be decreasing at a uniform rate. However, the North American network appears to be largely stable, subject to regular fluctuations. This is in contrast to the fraction of vertices in the giant component, where both Brazil and Europe exhibit sharp increases.

2.2.1.2 Online social networks

Holme *et al.* (Holme et al., 2004) obtained an almost exhaustive history of user interactions in *Pussokram*, an online dating social network popular in Sweden. This network is somewhat unusual because it contains full temporal information about multiple modes of user interactions. Users can, for example, list each other as 'friends', or write private or public messages to each other, or 'flirt' with one another, and each type of interaction generates a different network. We report the authors' results on the aggregate network of all interactions and on friendship links alone, since the other trends are very similar. All interaction types yield networks that exhibit an initially increasing average degree, but in

all cases, the trend appears to be quickly converging to a constant. In the case of friendship links, the average degree is approximately constant for more than 400 out of 500 days in the observation period. The authors note that this appears to agree with sociological constraints, such as empirical findings that maximum social network sizes in humans appear to be bounded (Hill and Dunbar, 2003).

Local density in the *Pussokram* network also exhibits some interesting trends. The authors compute both directed and undirected clustering coefficient in the network over time, and show that the network of all interactions exhibits a decreasing clustering coefficient, similar to earlier findings in bibliographic networks by Barabási *et al.* (Barabási et al., 2002). The network consisting only of friendship links, however, displays a non-monotonic trend, and it is hard to draw any conclusions about long-term behavior. The authors note that since *Pussokram* is primarily a dating site, conventional social norms for introducing one's friends to each other, and thus increasing local density, might not apply.

Kumar *et al.* (Kumar et al., 2006) obtained complete temporal information on the evolution of two online social networks - *Yahoo 360* and *Flickr*, both of which are treated as growing networks. A key finding, which can perhaps be generalized to other online social networks where complete temporal information is available, is that there appear to be multiple regimes of growth. The authors note that these stages correspond to an "initial euphoria" peak as early adopters sign up for the service, a subsequent decline of the initial enthusiasm, followed by steady "organic growth". Both networks exhibit these phenomena

in terms of the average degree of nodes, with the long-term trend appearing to be an increasing one.

Beyene *et al.* (Beyene et al., 2008) examine the structure of a *trust network*, built from binary feedback ratings on the online auction site eBay. These ratings are assigned to users by other users on the completion of a transaction, and can be either positive or negative, although they are generally found to be positive. Beyene *et al.* show that the average degree in this network increases almost linearly with time, from 1999 to 2005. Note that the data was collected by crawling user profiles on the eBay site, which brings up issues of incomplete data, and consequently, the missing past.

Hu *et al.* (Hu and Wang, 2009) analyze the temporal evolution of *Wealink*, an online social network popular in China, as a dynamic network. Perhaps the most interesting feature of their dataset is the S-shaped trend, resembling a logistic function, of both the number of vertices and edges over time. Since the dataset is an exhaustive history of the evolution of the network, the authors speculate that the sharp increase in the number of vertices and edges corresponds to a sudden burst of popularity, when the membership of the network grew from under 10,000 vertices to over 200,000 in less than 5 months out of a 27 month history, similar to earlier findings by Kumar *et al.* (Kumar et al., 2006). Both the global density D and the average clustering coefficient $\overline{CC_1}$ appear to decrease towards a constant once the growth of the network has stabilized. A fit to the logistic function suggests that the number of vertices has an asymptotic limit of $V \sim 224,000$ and $E \sim 272,100$.

2.2.1.3 Subsets of the World Wide Web

Buriol *et al.* (Buriol et al., 2006) analyzed the evolution of the structure of hyperlinks between articles in *Wikipedia*, an online, publicly-editable encyclopedia. Using snapshots of the link structure that model Wikipedia as a dynamic network, they find that the average out-degree $\overline{N_{out}}$ is increasing at a constant rate of approximately one new out-link every 100 days. The average clustering coefficient remains approximately constant in the last year of network evolution, after periods of non-monotonicity. This is somewhat surprising, since it implies that the local density around vertices remains constant even though the average number of neighbors increases.

Shi *et al.* (Shi et al., 2007) analyzed four temporal snapshots of a partial crawl of blogspace released as part of the TREC Blog-Track 2006 dataset. They report that that the average degree $\overline{N}$ of the network increases from 1.5 to 2.657 over a period of 40 days. Similarly, $\overline{CC_1}$ increases monotonically from 0.034 to 0.052 over the same period.

Latapy and Magnien (Latapy and Magnien, 2008) analyze a crawl of the `.uk` WWW domain, where each hyperlink is labeled with the time that it is discovered. Note that the motivation for the study was not to study the evolution of the network, but to determine how large a network sample should be in order for network properties to stabilize. In order to achieve this, the authors add edges sequentially to a network in increasing order of the timestamp of the edge. Thus, their methodology essentially builds a growing network, allowing us to compare their results, at least qualitatively, to studies that explicitly analyze network properties over time. Although the timestamp on the hyperlinks

in the WWW dataset correspond to the link's discovery time and not the link's creation time, we include the datasets in this survey since the same methodology has been used in other papers (Leskovec et al., 2007). The average degree grows quickly as a function of the number of nodes added to the network, but the density appears to decrease smoothly and the clustering coefficient is extremely non-monotonic. The Web dataset also seems to show sharp discontinuities in time, perhaps implying problems in the underlying crawling process. Thus, the trends in this particular dataset should not be given too much importance.

2.2.1.4 Internet Routers and Autonomous Systems

Dhamdhere and Dovrolis (Dhamdhere and Dovrolis, 2008) analyze the structure and evolution of a specific type of link between Autonomous Systems, namely 'customer-provider' (CP) or 'paid transit' links instead of all links. They perform extensive filtering and smoothing on the dataset, and show that the limited nature of AS datasets nonetheless allows reasonable estimation of the topology of CP links, but not all links. The average degree $\overline{N}$ over CP links is shown to be steadily increasing over time.

2.2.1.5 Dynamic Interaction Networks

Kossinets and Watts (Kossinets and Watts, 2006) analyzed a dynamic network of e-mail communications between university students, staff, and faculty. Since e-mail constitutes a dynamic interaction network, they used decay windows of various lengths in order to

smooth network structure. Specifically, each edge between vertices i and j is weighted at time t according to the following function[1]:

$$w_{ij} = \frac{\sqrt{m_{ij}m_{ji}}}{\tau}$$

where τ is the length of a chosen smoothing time window, and m_{ij} is the number of messages sent from i to j in the period $(t - \tau, t]$. For vertices i and j, if $w_{ij} > 0$ at time t, then the edge (i, j) exists in the network at that time. This introduces a form of edge decay over time, since e-mails allow the observation of the creation of links, but not of their dissolution. It also implements the intuitive idea that an e-mail should not constitute a social tie in a network indefinitely.

Using this edge decay scheme and window sizes of $\tau = 30, 60, 90$ days, the authors find that the average degree $\overline{N}$ is approximately constant during the semester, decreasing sharply during the semester break and in summer. Similarly, the average clustering coefficient $\overline{CC}$ appears to be largely in equilibrium during the semesters, its trend only increasing slightly during summer. The drops in average degree during summer, for example, can be explained by students not being on campus. Although the authors were only able to obtain data for one academic year, their results raise a number of interesting points: namely, that a single, global smoothing parameter appears to show that the network is largely in equilibrium

---

[1]Found in the supporting online material for (Kossinets and Watts, 2006), available at `www.sciencemag.org`.

during the semesters. Whether this phenomenon could be observed in other networks using a similar smoothing mechanism is an open question.

Pallis *et al.* (Pallis et al., 2009) analyze the structure of dynamic vehicular ad-hoc networks, *i.e.,* temporary wireless networks created when specially-equipped vehicles on open roads are within transmission range of each other. Their study is notable because it does not use actual interaction data, but rather the results of a large-scale realistic traffic simulation (Raney et al., 2002; Naumov et al., 2006). The authors simulate traffic in the center of the city of Zurich, Switzerland for 3 hours in the morning rush period, and do not use any form of edge decay or sliding window, as in Kossinets and Watts (Kossinets and Watts, 2006). Since the time evolution of the network reflects instantaneous traffic patterns, it is not surprising that the average degree increases gradually as traffic starts to build, and then falls off. The clustering coefficient, on the other hand, remains constant throughout the simulation period, making this an example of a graph where local and global density appear to uncorrelated.

Pallis *et al.* (Pallis et al., 2009) also illustrate a somewhat unusual usage of the DPL relationship in their analysis of communication links in dynamic vehicular ad-hoc networks. We have mentioned that the authors do not use any form of edge decay or smoothing, resulting in an instantaneous picture of communication links at each timestep in the evolution of the network. The authors fit the Multiplicative DPL to the model and find a densification exponent of $\alpha = 1.77$, although the interpretation of this value is difficult.

Year	Reference	Network	Type	Category	$\overline{N}$	D	$\overline{CC}$
2002	(Barabási et al., 2002)	Mathematics	Biblio.	growing	increasing	-	decreasing
		Neuroscience	Biblio.	growing	increasing	-	decreasing
2004	(Holme et al., 2004)	Pussokram: *All*	Online	growing	increasing[a]	-	decreasing
		Pussokram: *Friends*	Online	growing	constant	-	[b]
2004	(Park et al., 2004)	RouteViews	AS	dynamic	increasing[b]	-	increasing
		Extended	AS	dynamic	-	-	-
2005	(Elmacioglu and Lee, 2005)	DBLP-DB	Biblio.	growing	increasing[c]	-	increasing[b]
2006	(Kossinets and Watts, 2006)	University	Email	interaction	periodic[d]	-	periodic[d]
2006	(Buriol et al., 2006)	Wikipedia	WWW	dynamic	increasing	-	constant[b]
2006	(Kumar et al., 2006)	Flickr	Online	growing	increasing[b]	-	-
		Yahoo 360	Online	growing	increasing[b]	-	-
2007	(Shi et al., 2007)	TREC	WWW	dynamic	increasing	-	increasing
2008	(Dhamdhere and Dovrolis, 2008)	AS-CP	AS	dynamic	increasing[b]	-	-
2008	(Latapy and Magnien, 2008)	INET	Router	growing	increasing	[b]	increasing[a,b]
		eDonkey	P2P	growing	increasing	decreasing[a,b]	decreasing[b]
		Metrosec	Router	growing	constant[b]	decreasing[a,b]	decreasing[a,b]
2008	(Beyene et al., 2008)	eBay	Online	growing	increasing	-	-
2009	(Menezes et al., 2009)	Brazil	Biblio.	growing	-	-	decreasing
		Ca-US	Biblio.	growing	-	-	decreasing
		Fr-Sw-UK	Biblio.	growing	-	-	constant[b]
2009	(Hu and Wang, 2009)	Wealink	Online	dynamic	-	constant[b]	constant[b]
2009	(Pallis et al., 2009)	Vehicle traffic	Ad-hoc	interaction	[b]	-	constant

[a] Trend suggests convergence to asymptote.

[b] Trend exhibits non-monotonic behavior.

[c] Trend measured using fully dynamic network aggregation.

[d] Uses smoothing, sliding windows, or edge decay with fully dynamic network aggregation.

TABLE II: OVERVIEW OF LOCAL AND GLOBAL DENSITY MEASUREMENTS. NOTE THAT THESE ARE QUALITATIVE ASSESSMENTS BASED ON GRAPHICAL DATA PRESENTED IN EACH PAPER.

Year	Reference	Network	Type	Category	$\bar{l}$	$d_{\max}$	d_{90}
2002	(Barabási et al., 2002)	Mathematics	Biblio.	growing	decreasing[a]	-	-
		Neuroscience	Biblio.	growing	decreasing[a]	-	-
2003	(Nascimento et al., 2003)	SIGMOD	Biblio.	growing	[b]	constant[b]	-
2004	(Holme et al., 2004)	Pussokram: *All*	Online	growing	decreasing[a]	-	-
		Pussokram: *Friends*	Online	growing	increasing[b]	-	-
2004	(Park et al., 2004)	RouteViews	AS	dynamic	decreasing	-	-
		Extended	AS	dynamic	[b]	-	-
2005	(Elmacioglu and Lee, 2005)	DBLP-DB	Biblio.	growing	constant[b]	-	-
2006	(Kumar et al., 2006)	Flickr	Online	growing	constant[b]	-	increasing[b]
		Yahoo 360	Online	growing	decreasing[b]	-	decreasing[b]
2006	(Kossinets and Watts, 2006)	University	Email	interaction	periodic[d]	-	-
2007	(Ahn et al., 2007)	Cyworld	Online	dynamic	decreasing[b]	-	decreasing[b]
2008	(Dhamdhere and Dovrolis, 2008)	AS-CP	AS	dynamic	constant[b]	-	-
2008	(Latapy and Magnien, 2008)	INET	Router	growing	decreasing[a]	decreasing[a]	-
		eDonkey	P2P	growing	decreasing[a]	decreasing	-
		Metrosec	Router	growing	constant	constant	-
2009	(Menezes et al., 2009)	Brazil	Biblio.	growing	[b]	-	-
		Ca-US	Biblio.	growing	decreasing	-	-
		Fr-Sw-UK	Biblio.	growing	[b]	-	-
2009	(Hu and Wang, 2009)	Wealink	Online	dynamic	decreasing[ab]	constant[b]	-
2009	(Pallis et al., 2009)	Vehicle traffic	Ad-hoc	interaction	noisy	-	-

[a] Trend suggests convergence to asymptote.

[b] Trend exhibits non-monotonic behavior.

[c] Trend measured using fully dynamic network aggregation.

[d] Uses smoothing, sliding windows, or edge decay with fully dynamic network aggregation.

TABLE III: OVERVIEW OF AVERAGE AND MAXIMUM DISTANCE MEASUREMENTS. NOTE THAT THESE ARE QUALITATIVE ASSESSMENTS BASED ON GRAPHICAL DATA PRESENTED IN EACH PAPER.

Year	Reference	Network	Category	Type	Time span	Data	Miss. past
2002	(Barabási et al., 2002)	Mathematics	Biblio.	growing	1991-1998	8	yes
		Neuroscience	Biblio.	growing	1991-1998	8	yes
2003	(Nascimento et al., 2003)	SIGMOD	Biblio.	growing	1975-2002	28	yes
2004	(Holme et al., 2004)	Pussokram	Online	growing	512 days	large	yes
2004	(Park et al., 2004)	RouteViews	AS	dynamic	1997-2002	large	yes
		Extended	AS	dynamic	∼7 weeks	9	yes
2005	(Elmacioglu and Lee, 2005)	DBLP-DB	Biblio.	growing	1968-2003	36	yes
2005	(Leskovec et al., 2005)	arXiv	Biblio.	growing	1993-2003	124	yes
		Patents	Citation	growing	1975-1999	25	yes
		AS	AS	dynamic	1997-2000	735	yes
		Affiliation	Biblio.	growing	1992-2002	10	yes
2006	(Kossinets and Watts, 2006)	University	Email	interaction	∼1 year	large	yes
2006	(Buriol et al., 2006)	Wikipedia	Web	dynamic	2002-2006	17	yes
2006	(Kumar et al., 2006)	Flickr	Online	growing	100 weeks	100	no
		Yahoo 360	Online	growing	40 weeks	40	no
2007	(Ahn et al., 2007)	Cyworld	Online	dynamic	2002-2006	8	yes
2007	(Shi et al., 2007)	TREC	Blogs	growing	40 days	4	yes
2007	(Leskovec et al., 2007)	Email	Email	growing	18 months	18	yes
		IMDB Actors-Movies	General	growing	1890-2004	114	no
2008	(Latapy and Magnien, 2008)	INET	Router	growing	16 months	large	yes
		eDonkey	P2P	growing	47 hours	large	no
		Metrosec	Router	growing	8 days	large	yes
2008	(Dhamdhere and Dovrolis, 2008)	AS-CP	AS	dynamic	1998-2007	40	yes
2008	(Huang et al., 2008)	CiteSeer	Biblio.	growing	1980-2005	25	yes
2008	(Beyene et al., 2008)	eBay	Online	growing	1999-2005	7	yes
2009	(Menezes et al., 2009)	Brazil	Biblio.	growing	1994-2006	13	yes
		Ca-US	Biblio.	growing	1994-2006	13	yes
		Fr-Sw-UK	Biblio.	growing	1994-2006	13	yes
2009	(Hu and Wang, 2009)	Wealink	Online	dynamic	2005-2007	27	no
2009	(Dong et al., 2009)	China	Airport	dynamic	1983-2006	3	(?)
2009	(Pallis et al., 2009)	Vehicle traffic	Ad-hoc	interaction	3 hours	large	no

TABLE IV: AN OVERVIEW OF THE CHARACTERISTICS OF EVOLVING NETWORK DATASETS.

2.2.2 Connectivity

2.2.2.1 Bibliographic networks

Barabási *et al.* (Barabási et al., 2002) were among the first to find a decreasing average shortest path length $\bar{l}$ in a growing network, which does not agree with network growth models like Preferential Attachment (Newman, 2001a). For both the Mathematics and Neuroscience datasets that they examine, $\bar{l}$ is decreasing and apparently converging to an asymptote. The authors note that a longer observation interval might indicate that $\bar{l}$ approaches a stationary value, but the relative novelty of co-authorship datasets at the time of publication resulted in just 8 data points, with each representing an additional year of cumulative co-authorships. Note that although the authors refer to 'diameter', the only quantity studied is the average shortest path length $\bar{l}$.

Nascimento *et al.* (Nascimento et al., 2003) analyzed the co-authorship graph of the SIGMOD conference as a growing network. The average path length in the largest connected component in the network varies considerably over the years, but eventually settles down into what appears to be a decreasing trend. We have not listed any trend in Table Table III because of the extremely short time period that any trend is visible at all. The authors note that computing path lengths within the largest component only has its caveats: until 1980, only 16 authors were in the largest connected component, out of the 1,683 that eventually appear. The trend is $\bar{l}$ is therefore predictably noisy. The authors also report that $d_{\max}$ in the network has been constant at a value of 15 between 1996 and 2002.

Elmacioglu and Lee (Elmacioglu and Lee, 2005) analyzed the growing network of publications in the database research community, constructed from a set of manually selected publication venues reported to the DBLP database. They report the value of the average shortest path length $\bar{l}$ over time, computed over all reachable pairs of vertices in the network. After an initial increasing burst, the trend in $\bar{l}$ appears to stabilize to a constant value around 6 from approximately 1989 to 2003. While it is certainly possible that database research was particularly energized between 1973 and 1983, resulting in an infusion of new authors and sharply increasing $\bar{l}$, it is also possible that this spike is an artifact caused by missing past issues, or known limitations in the indexing process of DBLP for early data (Ley, 2002). However, the fact that $\bar{l}$ appears to stabilize for a number of years would suggest that the network has reached an equilibrium.

Menezes *et al.* (Menezes et al., 2009) analyzed three co-authorship networks constructed from faculty publications at manually selected universities in North American, Europe, and Brazil. The average shortest path length $\bar{l}$ in the North American network displays a relatively uniform decreasing trend, but both the Brazilian and European networks exhibit sharp increases by doubling between 1998 and 2001. Following these sharp increases, the Brazilian network appears to settle into a decreasing trend, whereas the European network continues increasing. It should be noted that in the Brazilian research network, the doubling of $\bar{l}$ is correlated with a sharp increase in the fraction of nodes in the giant component.

2.2.2.2 Online social networks

In their analysis of the *Pussokram* online dating social network, Holme *et al.* (Holme et al., 2004) find different trends for networks built from different types of interactions. Note that although the authors have complete history of the network, registered users of a different service had their accounts "automatically transferred to pussokram.com" on its inception (Holme et al., 2004), which implies that this dataset also has a version of the missing past issue. The authors report different trends in the average shortest path length $\bar{l}$, computed only within the largest giant component as opposed to between all pairs of reachable nodes, depending on the type of interaction used to build the network. The aggregate network built from all possible user interactions, for example, shows a decreasing trend in $\bar{l}$, appearing to converge to a constant, but the network built only from friendship links displays an *increasing* trend after a very brief period of initial decrease. One possibility is that each type of user interaction on the social network is governed by a different process, which results in different trends in each network, but it is also possible that the differences are caused by disparities in the amount of data on each type of interaction.

Kumar *et al.* (Kumar et al., 2006) analyzed social networks of the *Yahoo 360* and *Flickr* services as growing networks. Of these, the *Flickr* network is somewhat unusual in terms of both the $\bar{l}$ and d_{90} measures. In the "organic growth" phase, it is difficult to discern a long-term trend in either measure from graphical data. Although the authors state that both measures decrease over time, there appears to be a slight, monotonic *increase* in both measures for approximately the last 30 weeks out of 100. The *Yahoo 360* network

also exhibits noisy trends, although the long-term trend in both $\bar{l}$ and d_{90} appear to be decreasing. Both these networks are exception because complete temporal information is available for every network-altering activity. They also illustrate the problems associated with characterizing the properties of real networks as simple, monotonic trends.

Ahn *et al.* (Ahn et al., 2007) present an analysis of the evolution of friendship links in *Cyworld*, an online social network popular in South Korea. Although they claim to have obtained the complete topology of the network directly from the service provider, the amount of temporal information is limited to about 42% of the data. Thus, the missing past issue is a consideration in the temporal analysis of this dataset. The authors find that the average shortest path length $\bar{l}$ increases almost linearly for the first three and a half years of data, but then gradually starts to drop. The effective diameter d_{90}, on the other hand, stays approximately constant during the early period, before dropping off sharply and almost converging to the $\bar{l}$ trend. Note that the drop in both $\bar{l}$ and d_{90}, and particularly the reversal of the trend in $\bar{l}$, coincides with a sharp increase in the number of nodes in the network, which could indicate a change in the evolution dynamics of the network caused by perhaps some sort of external event or marketing effort. In either case, it reinforces the need to consider the underlying processes reflected in the dataset.

Hu *et al.* (Hu and Wang, 2009) note that a complete temporal history of the *Wealink* online social network exhibits non-monotonic behavior in both the average shortest path length $\bar{l}$ as well as the diameter d, *i.e.,* the maximum shortest path length. However, the network underwent a rapid and extremely pronounced burst of growth, after which network

properties seem to stabilize. The long-term trend in $\bar{l}$ appears to be slowly decreasing, whereas the diameter d stays constant. As Leskovec *et al.* (Leskovec et al., 2005) point out, the diameter is extremely sensitive to outlier structures, so the effective diameter d_{90} or 'almost' longest path might follow a different trend.

2.2.2.3 Internet Routers and Autonomous Systems

In their analysis of the evolution of CP links in Autonomous Systems, Dhamdhere and Dovrolis (Dhamdhere and Dovrolis, 2008) show that the average shortest path length $\bar{l}$ has remained approximately constant over the last 9 years.

Dhamdhere and Dovrolis (Dhamdhere and Dovrolis, 2008) analyze the structure and evolution of a specific type of link between Autonomous Systems, namely 'customer-provider' (CP) or 'paid transit' links instead of all links. They perform extensive filtering and smoothing on the dataset, and show that the limited nature of AS datasets nonetheless allows reasonable estimation of the topology of CP links, but not all links. The average degree $\overline{N}$ over CP links is shown to be steadily increasing over time.

2.2.2.4 Dynamic Interaction Networks

Kossinets and Watts (Kossinets and Watts, 2006) analyzed a university's e-mail records for one academic year as a smoothed dynamic interaction network, which we have previously described in Section 2.2.1.5. The authors report two interesting findings. The first is that with smoothing windows of 60 and 90 days, the average shortest path length $\bar{l}$ stays constant, except after the onset of summer, when it rises. This implies that the network is in equilibrium until the summer, when presumably a large fraction of the student population

leaves campus and uses e-mail less frequently. The second finding of interest is that for the shortest smoothing window of 30 days, intended to capture fast-changing dynamics, there is a strong correlation between the fraction of vertices in the largest component and the average shortest path length $\bar{l}$ with the largest component. The first phenomenon can again be explained by its occurrence during semester breaks, but the correlation between $\bar{l}$ and the size of the giant component raises the question of the underlying process responsible for the increase or decrease in $\bar{l}$ in other studies that compute $\bar{l}$ within the largest component only.

2.2.3 Summary

In conclusion, empirical evidence seems to suggest the following common trends in real networks, as measured using the growing network methodology:

1. The average shortest path length is decreasing over time, often appearing to converge to a fixed value.
2. The effective diameter decreases over time.
3. The average degree grows over time, apparently without a bound.

In the next two sections, we investigate how sensitive these trends might be to various kinds of sampling error. The next section deals with citation-type networks, and the subsequent section with interaction-type networks.

2.3 Sensitivity of measured trends in citation networks

Recall that citation networks grow over time with the addition of nodes and edges, and that each individual edge can only appear once in the timestream, to represent true growth in the underlying system. When the growing network method is used to aggregate observations of a citation network, an implicit assumption is that every aggregated observation is representative of the underlying system at that point in time, with respect to a measure M, *i.e.,* that the magnitude of change in the underlying system from time t_1 to time t_2 is proportionally represented in a change in measure M from t_1 to t_2, and vice versa.

There is, however, a condition under which this assumption does not hold: when the dataset does not contain a full temporal history of the underlying process. This is called a *missing past*, and a version of it has been briefly described before in (Barabási et al., 2002) and (Leskovec et al., 2005). Assume that a physical system is continuously in a state of flux, but that the observations in a citation network dataset of it necessarily start at an arbitrary time $t_0 > 0$. Furthermore, assume that due to limitations in the observation process, the partial picture we have of the underlying system at time t_0, with respect to measure M, is incomplete. Since individual edges are only observed once in a citation network's lifetime, we will never discover the existence of edges that occurred before time t_0.

However, the same is generally not true for discovering vertices that existed before time t_0. When a new vertex joins the network and links to a vertex that existed before t_0, the older vertex is 're-discovered' and can incorrectly be presumed to be a new vertex. We call this phenomenon *vertex re-dicovery*, and it can sometimes be controlled with additional

metadata. For example, Leskovec *et al.* (Leskovec et al., 2007) use a patent citation dataset where the time that each node was created is explicitly recorded, eliminating any vertices for which they do not have a creation time. However, when this information is not available, each *apparently* new vertex can affect a measure M by a significant amount, increasing the measurement error between the true and observed networks. In some cases, as we will show, this error can progressively alter trends in some measures over time, suggesting change in the underlying network in a manner that is not actually happening. This is the first source of error in dynamic network measurements that we analyze.

In addition to vertex re-discovery affecting graph measurements, a second source of error is simply the structure of the missing past graph. Even assuming that the measurement process is perfect and accurately records all vertex and edge additions in the observation stream, the unobserved temporal history of the network contains a missing past graph of unknown structure. The true change in the underlying system is determined by the structure of this missing past graph in relation to the actual observations. Unfortunately, this missing data graph looks like is difficult to determine, in general, for real datasets.

2.3.1 Assessing sensitivity

The central question of this section is whether the errors introduced by the processes just mentioned are significant enough to be of concern. Given a dataset and a measure M, the difficulty of assessing the sensitivity of a temporal trend in M is that we would require the structure of the missing past network in order to do so. However, a number of stochastic models of growing networks have already been developed that allow us to generate

presumably realistic, continuously growing, citation networks (Chakrabarti et al., 2010). This allows us to conduct a systematic study of the effects of various amounts of missing past on network measurements by using simulations of these models to generate data. The diversity of network growth models in the literature means that we can systematically study a reasonably large class of dynamical systems.

We use a simulation setup where a network growth model is used to generate a synthetic dynamic network dataset with known properties. We designate this as the ground truth network. By censoring the initial portion of the synthetic network timeline, we can simulate both growth and vertex re-discovery without modifying the network growth model. The independent parameter of the simulation (beyond the parameters of individual network growth models) is the amount of data to censor, which has a well-defined meaning, and is the primary phenomenon we are interested in. Ideally, the missing past effect would be small and the graph properties we study would converge to their true values quickly, or at least yield qualitatively similar trends over time.

There have been two prior attempts, to the best of our knowledge, to investigate the impact of missing past data on network trends over time when using the growing network aggregation method. This is in contrast to the many studies described in Section 2.2 that simply use the growing network method under the same conditions. In 2002, Barabási *et al.* described a supplementary experiment assessing the effect of missing past on the average shortest path length over time, in their seminal paper on co-authorship networks (Barabási et al., 2002). Our simulation setup in this section is very similar to theirs, but we system-

atically study several citation network generating processes, graph measures, and amounts of missing past data.

The second experiment was performed in 2007 by Leskovec *et al.*, who use a subtly different definition than Barabási *et al.* of what constitutes the missing past. Namely, they only consider a limited form of the missing past in citation networks: "citations to [vertices] that predate our earliest recorded time" (Leskovec et al., 2007, pg.17). Where Barabási *et al.* analyzed the effect of missing past using synthetic simulation data, Leskovec *et al.* use a real dataset, known to have a missing past with unknown structure, in order to determine what the impact of the missing past would be. They used the following experiment to validate that the trend they observed in the *effective diameter* graph measure was not an artifact of the observation process (note that $t = 0$ corresponds to the start of observations in the quotation below, not to the start of the process as in our notation):

> We pick some positive time $t_0 > 0$ and determine what the diameter would look like as a function of time if this were the beginning of our data. We then put back in the nodes and edges from before time t_0 and study how much the diameters change. If this change is small – or at least if it does not affect the qualitative conclusions – then it provides evidence that the missing past is not influencing the overall result (Leskovec et al., 2007).

We can now illustrate how simulations with synthetic data can shed light on the efficacy of the test described above. A well-known network generation model is the *preferential attachment*, which we describe in more detail subsequently in Section 2.3.2, and which is

known to generate graphs with a slowly growing diameter (see Section 2.1.3 for definition). Starting with an initial seed graph of a single vertex, we use the preferential attachment model to generate a random 'ground truth' growing citation network. After an arbitrary number of timesteps t_0, we simulate the start of the observation process. Specifically, the entire structure of the network prior to t_0 is considered to be the missing past and censored, but additions to its structure are observed and aggregated into a growing graph. We can then compare the value of some measure M, here the effective diameter, on the full ground truth network and the truncated, mising past network. This allows us to measure the impact of different amounts of missing past.

Figure 5 shows the ground truth for the d_{90} (effective diameter) measure over time for the true dataset and the measured, missing past network. In this instance, the effect of the missing past is extremely significant – where the ground truth network shows a slowly increasing diameter, the missing past network would suggest that the diameter is rapidly decreasing over time. The inference we make about the change in the structure of the underlying process is therefore an artifact of the structure of the missing past network and vertex re-discovery. This essentially reproduces the results of the experiment conducted by Barabási *et al.* , using effective diameter as the measure of interest instead of average shortest path length (Barabási et al., 2002).

We then conducted the experiment described by Leskovec *et al.* above on the truncated missing past network, treating it as the observed dataset, and knowing the ground truth in advance. We chose an arbitrary time greater than t_0 and essentially repeated the pro-

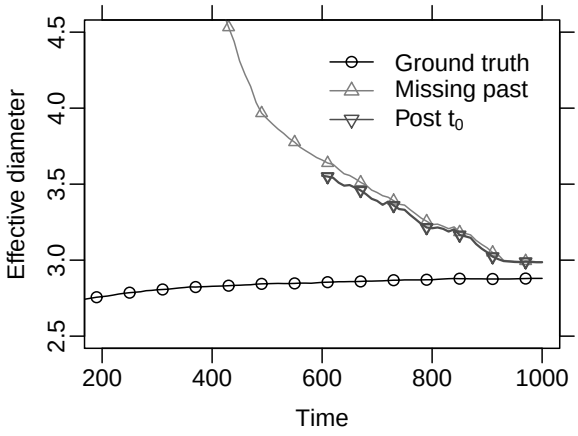


Figure 5: The effective diameter over time in a synthetic preferential attachment network, the observed trend after an initial portion is censored to simulate the missing past, and the trend produced by a 'Post-t_0' validation experiment

cedure of truncating the network before that time. The 'Post-t_0' validation experiment used by Leskovec *et al.* intuitively states that if the trend in the missing past network and the 'Post-t_0' are comparable, then the trend in the missing past network and the ground truth are also comparable. However, we find that the trend resulting from the validation experiment is essentially identical to the trend in the missing past network, in agreement with similar observations on real datasets in (Leskovec et al., 2007). However, both trends are nonetheless artifacts here of the missing past, given that the ground truth has a slowly increasing d_{90}. The validation experiment described in (Leskovec et al., 2007) therefore appears to be limited in its ability to discern a spurious trend from a representative one, as shown in Figure 5.

We use the same methodology in combination with other network growth models and graph measures to estimate the impact of various amounts of missing past data (and thus vertex re-discovery). Figure 5 shows the output of a single random trial for clarity, but in general, we are interested in the *expected* trend of a measure M in the presence of missing past data, over many random trials. In the next subsection, we describe the stochastic network growth models we will use in our empirical analysis in Section 2.3.3.

2.3.2 Network growth models

A network growth model can be viewed as a probabilistic algorithm that adds nodes and edges to a graph, driven by random noise. A full review of network growth models is beyond the scope of this thesis, but we describe a few key models that will be used here. For a comprehensive survey of network growth models, the reader is referred to (Chakrabarti and Faloutsos, 2006; Chakrabarti et al., 2010).

Definition 2.3.1. *Network growth model.* A network growth model accepts a graph $G = (V, E)$ as input and a vector of parameters Θ, and stochastically produces an output graph $G' = (V', E')$, where $V \subseteq V'$ and $E \subseteq E'$.

Growth models are generally applied recursively, *i.e.,* an initial seed graph is given to the growth model, and the output graph becomes the input graph for the next timestep, for a fixed number of timesteps. Figure 6 shows graph layouts of three network growth models for networks of increasing size. The following are three network growth models that we will consider in this chapter:

1. (*Dynamic Random Attachment*) This is possibly the simplest network growth model, originally proposed by Callaway *et al.* (Callaway et al., 2001) to study the properties of randomly growing graphs relative to classical Erdös-Rényi graphs. The growth process is as follows: at every timestep, a new vertex is added to the graph, and with constant probability p, two unconnected vertices are connected with an edge uniformly at random. The observation stream therefore consists of an isolated vertex at each timestep with probability $(1-p)$, or an edge and two or three vertices with probability p. In a variant, we choose an arbitrary pair of vertices for the observation stream instead of an unconnected pair. In the former version, the missing past simulates vertex re-discovery; in the latter, both vertex and edge re-discovery. The first version simulates a citation network, and the latter an interaction network. Finally, since one new vertex is added at each timestep, and the expected number of edges at time t is pt, the expected average degree over time is constant at $2p$.

2. (*Preferential Attachment*) The Barabási-Albert *preferential attachment* (PA) model (Barabási and Albert, 1999; Barabási et al., 2002) was one of the first random graph models intended to generate 'realistic' graphs. It is based on the basic principle that a vertex that has just joined the network will randomly connect to an existing vertex with a probability directly proportional to the degree of the vertex being connected to. While most vertices will end up having a low degree, vertices that initially have a high degree will continue to rapidly increase in degree, as a mathematical embodiment of the 'rich-get-richer' adage. Remarkably, this simple game generates networks with many

of the graph-theoretic properties observed in real networks; for example, a skewed degree distribution of vertices, and a small average shortest path length (Bollobás et al., 2001; Newman, 2003; Boccaletti et al., 2006). Furthermore, the PA model is expected to generate graphs with a slowly growing diameter, *i.e.,* either as $\Theta(\log(V))$ or $O(\log(\log(V)))$ depending on the parameters (Bollobás and Riordan, 2004). The version of the PA model we consider here was presented as a model of bibliographic co-authorship networks by Barabási *et al.* (Barabási et al., 2002). Starting with an initial seed graph G_0 and two integer parameters $a > 0$ and $b > 0$, the following describes the simplified PA growth model applied to the graph at each timestep t:

(a) A new node u is added to the graph and connected to a existing vertices, where the probability of linking to vertex v is defined as

$$P(u, v) = \frac{N(v)}{\sum_{i \in V} N(i)}$$

where $N(v)$ is the degree of vertex v.

(b) b links are created between existing vertices in the graph, where the probability of a link between vertices i and j is defined as

$$P(i, j) = \frac{N(i)N(j)}{\sum_{s,m \in V, s \neq m} N(s)N(m)}$$

As a variant, in the second step of the algorithm above, if we do not distinguish between already unconnected and connected vertex pairs, the missing past effectively simulates edge as well as vertex re-discovery. Otherwise, only vertex re-discovery is simulated. Note that the version of the algorithm given above would have an expected diameter that grows as $\Theta(\log(V))$ where V is the number of vertices. At each timestep, one new vertex and $(a+b)$ new edges are added to the graph, so the expected average degree over time is constant at $2(a+b)$. If resampling of edges is allowed in the second step, then at most $(a+b)$ new edges are added at each timestep, and the average degree converges from below to $2(a+b)$. Since the average degree converges to a constant, the Densification Power Law is not applicable.

3. (*Forest Fire*) The Forest Fire model was proposed by Leskovec *et al.* (Leskovec et al., 2005) as an alternative to network growth models like Preferential Attachment, which generate graphs with a slowly increasing diameter and constant average degree. Instead, Leskovec *et al.* used the growing network methodology and found that the networks they analyzed showed a rapidly decreasing diameter over time, and super-linearly increasing average degree, which would invalidate Preferential Attachment as a dynamical model for growing networks. They proposed the Forest Fire model to generate graphs with a decreasing diameter and superlinearly increasing average degree.

   Although determining the expected properties of the Forest Fire model appears to be analytically intractable, it is able to generate graphs with the properties mentioned

above for specific parameter values described in (Leskovec et al., 2007). Algorithmically, it can be seen as a form of *copying* model (Kumar et al., 2000), where new nodes pick a set of *ambassadors* and then probabilistically link to their neighborhoods. Given two probability parameters p and r, the graph growth algorithm at each timestep adds a new vertex u as follows (Leskovec et al., 2007):

(a) u chooses an *ambassador node* v and links to it.

(b) Let x and y be two geometrically distributed random numbers with parameters $(1-p)$ and $(1-rp)$. u randomly links to $(x-1)$ in-link vertices of v and $(y-1)$ out-link vertices of v, excluding any that have already been visited in the current iteration.[1]

(c) The second step repeats at all nodes that have just been linked to, until the process dies out.

Consider the output of iterating any of the network growth models above. Recall that we start with an initial seed graph, possibly empty, which is designated $G_0 = (V_0, E_0)$. At each timestep, the network growth model generates an *observation graph* $G' = (V', E')$, which is added to the input graph to produce the output. Note that V' and E' can have elements in common with the input graph, but are not necessarily strict supersets or subsets of it.

---

[1]The -1 constants on x and y are not mentioned in the description of the algorithm, but are critical for reproducing the results in (Leskovec et al., 2005; Leskovec et al., 2007).

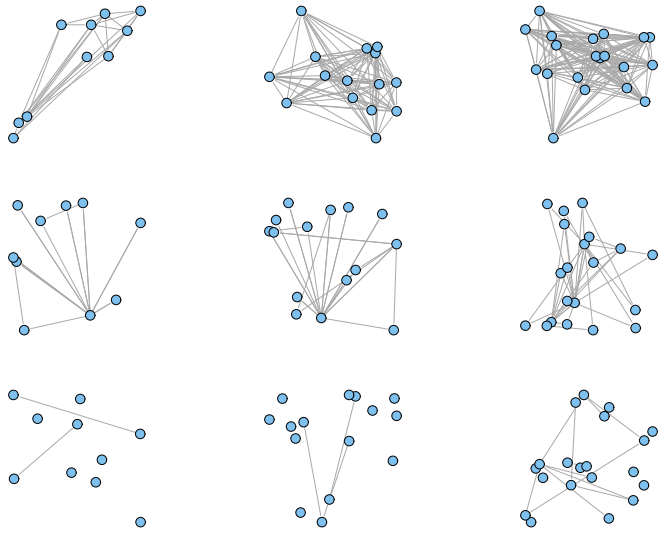


Figure 6: Examples of the output of network growth models for graphs of 10, 15, and 20 nodes: forest-fire (top row), preferential attachment (middle), random attachment (bottom).

Let Grow be such a network growth model:

$$\textsc{Grow} : G \times \Theta \rightarrow G'$$

where G is the input graph, Θ is set of model parameters, and G' is the output observation graph describing changes made to the input graph. Let C be the number of initial timesteps to censor to simulate the effect of re-discovery. Starting with timestep 0 and a seed graph

G, we iterate the network growth model in the following manner to generate the ground truth network:

$$O_1 = \textsc{Grow}(G, \Theta) \qquad G_1 = G \cup O_1$$

$$O_2 = \textsc{Grow}(G_1, \Theta) \qquad G_2 = G_1 \cup O_2$$

$$...$$

This yields the underlying (ground-truth) network:

$$\langle \mathbf{G} \rangle = \langle G, G_1, ... \rangle$$

We then censor the first C timesteps to generate the observed, aggregated network $\langle G^+ \rangle$ (see Definition 2.1.2).

$$\langle G^+ \rangle = \langle O_C, O_C \cup O_{C+1}, O_C \cup O_{C+1} \cup O_{C+2}, ... \rangle$$

Depending on the model, the occurrence of some vertices in $O_t, t \geq C$ will be caused by re-discovering a pre-existing vertex. This could happen, for example, when a new edge connects to a vertex that exists in the censored part of the network. In some network models, existing edges can be re-activated, so there is also the possibility than an edge in some O_t for $t \geq C$ is a re-activation of an edge in the censored part of the network. This describes a

very common 'missing-past' scenario in network data collection, and it is important to keep in mind that the properties of interest are those of the underlying network $\langle \mathbf{G} \rangle$.

2.3.3 Empirical results

The experimental methodology for simulating the missing past is simple: iterate a network growth model for C timesteps to generate a ground truth network, and then 'fork' its evolution into a secondary, "missing past" network that only receives updates to the ground truth network from timestep C onwards.[1] For example, if the ground truth network receives edge (u, v) at timestep $C + 1$, the missing past network at the same timestep contains *only* the edge (u, v). The larger the value of C, the larger the missing past network, and depending on the graph model, the higher the prevalence of the re-discovery process in uncovering censored vertices. We might intuitively expect that for small values of C, any difference between the properties of the ground truth and missing past network would converge to the same value very quickly. By varying C, we can test this assumption with any network growth model.

For each of the three network growth models described earlier, we started with the simplest seed graph permissible by the model, generally a single isolated vertex. The growth model was iterated for a total of 1,000 timesteps, with the missing past size ranging from 50 to 250 timesteps in increments of 50 timesteps. Note that this represents a very small

---

[1] In Unix-like operating systems, the $fork()$ system call creates a second concurrent copy of a process. The difference is that unlike a Unix process, our 'missing past' network does not inherit any data from the original ground truth network.

amount of missing past data. In all three models, this equates to a missing past graph of just 50 vertices. Various graph theoretic properties were measured on both the missing past and ground truth networks every 10 timesteps. All properties were averaged over 500 random trials. Shortest paths were computed exhaustively between all reachable vertex pairs using the `igraph` network library. We present results in the following subsections for the following model configurations:

1. Random attachment with edge creation probability $p = 0.8$ (sparse network[1]).
2. Preferential attachment with $a = 2$ and $b = 2$ (similar to (Barabási et al., 2002)).
3. Forest fire with $p = 0.35$ and $r = 0.57$ ('sparse graph' instance in (Leskovec et al., 2007)).

We report trends in the average shortest path length, effective diameter, clustering coefficient, largest eigenvalue of the adjacency matrix, and the densification exponent of the DPL.

2.3.3.1 Random Attachment

Figure 7 shows various properties of the ground truth network relative to 5 missing past networks. We focus first on the average shortest path length and the effective diameter, which are both statistics of the pairwise shortest path length distribution. Even with a relatively small amount of missing data of 50 timesteps, the trend in the observed dataset

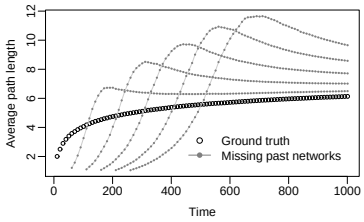
---

[1]Note that the edge creation probability is distinct from the edge probability in classical Erdös-Rényi graphs. In the random attachment growth model, $\lim_{p \to 1} E(t) = V(t)$, which is the classical definition of a sparse graph.

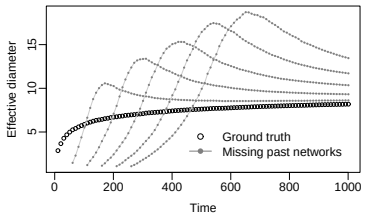
is always the opposite of the trend in the underlying network. This is not only a qualitative difference (increasing vs. decreasing), but also numerically quite a large difference at short times.

Predictably, as the amount of missing data increases, the observed trend takes longer to converge to the true trend; with 100 censored timesteps (the second gray trend from the left), the ground truth and missing past trends appear to be converging in about 1,000 timesteps – which represents more than 10 times the number of vertices in the missing past. This convergence takes considerably longer with more missing data. For example, with 250 missing past timesteps (and thus, 250 missing past vertices and $250 * 0.8 = 200$ edges in expectation), the missing past trend in the average shortest path length is a little less than double its ground truth value at the end of 1000 timesteps. The effective diameter is almost three times its true value. Perhaps what is of most concern is that qualitatively, the missing past trends appear to be decreasing, whereas the ground truth trend is increasing in both cases. The clustering coefficient and principal eigenvalue appear to be good qualitative approximations even with missing past data. However, the former is a biased measure that is intrinsically correlated with the average degree, and the latter is known to correlate with the maximum degree, decreasing their value as dynamic graph characterizations.

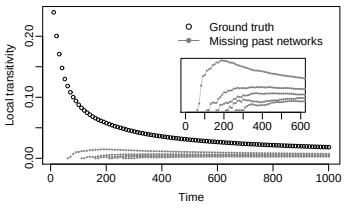
Finally, the most surprising finding here is that of the DPL densification exponent α (see Definition 2.1.6). Recall that α lies strictly between 1 and 2, and that a value greater than 1 indicates that the number of edges is growing superlinearly relative to the number of nodes. In the random attachment model, the number of edges is a constant function of the



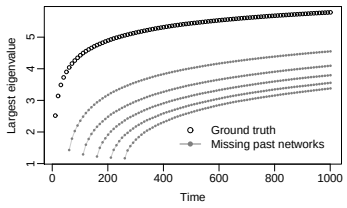
(a) Average shortest path length



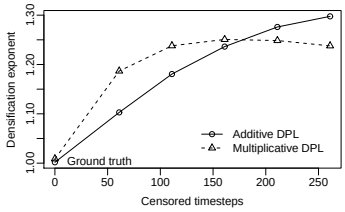
(b) Effective diameter



(c) Clustering coefficient (inset shows early stages of missing past network)



(d) Principal eigenvalue



(e) Densification exponent

Figure 7: Random attachment network model with $p = 0.8$: the effect of 5 different amounts of missing data on the ground truth (empty circles) and missing past (filled circles) networks.

number of nodes, so the DPL is known not to hold in this case. However, the re-discovery process has the following effect in the early stages of observed network growth: when a new edge is created between two vertices in the censored graph, the observations show the appearance of two new vertices and one new edge, instead of one new edge and no new vertices. The number of edges therefore appear to grow slowly initially, as censored vertices are re-discovered. This manifests in an apparently super-linear growth of the number of edges compared to the number of nodes over the later portion of the observation period, whereas the bias is really towards *sub*-linear growth in the early stages. Unfortunately, both phenomena can manifest as super-linear fits with regression.

For each missing past network, we used non-linear least-squares regression to fit the Additive DPL equation and linear least-squares on log-transformed data to fit the Multiplicative DPL equation. Figure 7e shows the densification exponent α for each missing past dataset, with $C = 0$ corresponding to the ground truth dataset. In all cases, α is greater than 1, apparently suggesting that densification is taking place in the underlying system when we know that this is not the case. Larger amounts of missing past lead to higher densification exponents using the Additive DPL equation; the Multiplicative DPL appears to level off and then decrease. This can be explained by the re-discovery process being more likely to uncover nodes that already existed in the censored portion of the network due to its increased size.

2.3.3.2 Preferential Attachment

We now consider a network that is growing according to the Preferential Attachment model, which is a more realistic than the random attachment model analyzed in the previous section. We conduct a similar set of experiments as with the random attachment model by censoring a portion of the initial output of the network growth model.

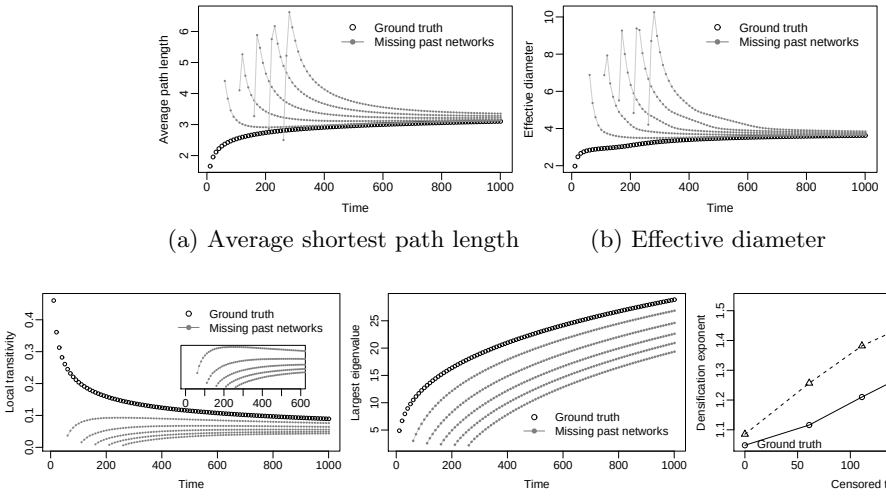
The results are quite similar to those of the random attachment model. Figure 8 shows measurements for the ground truth and missing past networks of various sizes. As with the random attachment model, the missing past networks exhibit false trends in both the average shortest path length and effective diameter over time. However, they are significantly more pronounced than in the random attachment model, both qualitatively as well as quantitatively. Although the trend in the underlying networks is growing slowly, as predicted by

theoretical results (Bollobás and Riordan, 2004), the observed network exhibits an initially sharply decreasing effective diameter, followed by slow convergence to the true trend. Thus, we can conclude that the spurious trends in the random attachment model were not solely the result of the random network structure. The clustering coefficient measure also exhibits a false trend at short times, but converges to the true trend quickly.

The DPL densification exponents α also show strong variability with the amount of missing past data. Particularly, the Additive DPL equation appears to exhibit a linear dependence on the size of the missing past network. However, it is unlikely that this property, even if it holds generally, can be used to determine the size of the missing past from the densification exponent; that would only be applicable if the underlying network was truly growing according to preferential attachment. Once again, two sources of concern are the strong dependence of α on the missing past size, and the very fact that α indicates densification with a very small amount of missing past when the ground truth network does not possess that property.

2.3.3.3 Forest fire

Finally, we analyze a configuration suggested by the authors of the Forest Fire model. Since the model appears to be analytically intractable (Leskovec et al., 2007), we use a parameter set that has been empirically shown to generate a growing sparse network with a slowly increasing effective diameter. We also adjusted the timescale used for the experiments to match that in the original paper to allow for an easy comparison of behavior; it is about 10 times longer than the timescale used for the other experiments. Since the Forest Fire



(a) Average shortest path length (b) Effective diameter

(c) Clustering coefficient (inset shows early stages of missing past network) (d) Principal eigenvalue (e) Densification exponent

Figure 8: Preferential attachment network model: the effect of 5 different amounts of missing data on the ground truth (empty circles) and missing past (filled circles) networks.

model is a type of 'copying' model (Kumar et al., 2000), we might expect different results than the random and preferential attachment models. Figure 9 shows results for the 'sparse' configuration of the Forest Fire model.

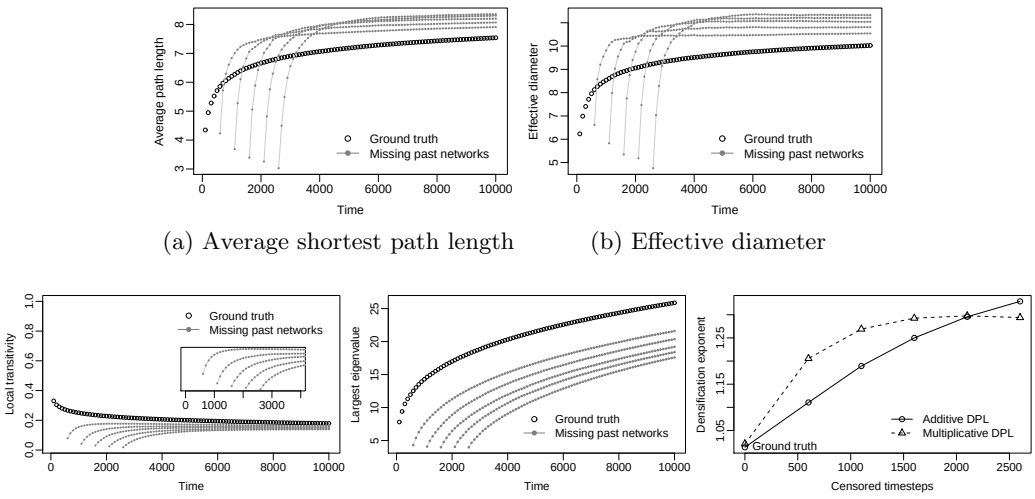
The trends in path length statistics are somewhat unusual: the ground truth network has a slowly increasing diameter, much like the preferential and random attachment models, the effective diameter in the missing past networks in Figure 9b are essentially constant after a brief initial period. This can be explained as a facet of the forest fire graph generation process. When a vertex is revealed in the missing past network, it burns edges and thus re-discovers old vertices in the ground truth network. In practice, this 'forest fire' process

(described in Section 2.3.2) burns out very quickly, and thus the re-discovered vertices are short distances away from the new vertex, keeping the effective diameter low. When the network is large enough, this effect becomes insignificant, and the trend manifests as an almost constant effective diameter. This is therefore a case when the data suggests a steady state where there is none.

Other properties not based on shortest paths display errors or consistency similar to the random and preferential attachment models: the clustering coefficient at short times shows a trend contrary to the ground truth, and the principal eigenvalue tracks its ground truth value qualitatively, if not quantitatively. The densification exponent, however, shows the same dependence upon the amount of missing past as the other models, ranging between approximately 1 and 1.3 for a ground truth value near 1. We note that the dependence of α resembles the trend in the random attachment model (Figure 7), suggesting that the dependence of the bias in certain cases might be related to the relative sizes of the missing past and observed networks. Whether this is a universal feature, and if the size of the missing past network might be computable from it, is a question for future research.

2.4 Sensitivity of measured trends in interaction networks

In the previous section, we asked how sensitive certain measures are on growing *citation* networks, given the presence of various amounts of missing past data. In this section, we ask if the same trends in the measures of growing *interaction* networks are meaningful. Recall that in an interaction network there is an underlying dynamic process that causes entities to interact with each other over time, and that we discover the structure of the



(a) Average shortest path length

(b) Effective diameter

(c) Clustering coefficient (inset shows early stages of missing past network)

(d) Principal eigenvalue

(e) Densification exponent

Figure 9: Forest fire model with $p = 0.35$ and $r = 0.57$: the effect of 5 different amounts of missing data on the ground truth (empty circles) and missing past (filled circles) networks.

network only through observing these interactions. This is an appropriate model for many communications and information networks, such as phone call, e-mail, physical proximity, and instant message logs. For example, in the case of e-mail networks, if a user never sends a single e-mail, their existence will generally not be noted on transmission logs, and conversely a user's first *observed* e-mail transmission would suggest true network growth to the observer, regardless of whether the observation contains the first instance of that edge.

One of the assumptions in measuring the change in a growing interaction network is, naturally, that the underlying network is truly changing with respect to some measure M. If this were not the case, then any trend in M over time would simply be a reflection of

the convergence of a sampling process to the limiting value of M. It would be a reflection of the sampling process, a product of interaction network dynamics and missing past data. In fact, at least one study uses the method described here to study the convergence of various network measures to limiting values (Latapy and Magnien, 2008). In this section, we expand upon an elegant conceptual framework proposed by (Pedarsani et al., 2008) called *edge sampling* to ask if a given dynamic network dataset is growing at all. In other words, is there a plausible dynamic sampling process operating on an *unchanging* network that can statistically explain a trend in measure M when M is truly constant?

Definition 2.4.1. *(Edge Sampling)* The basic assumptions of edge sampling are as follows:

1. There is a fixed or slowly changing (relative to the size of observations) underlying graph.
2. Edges in this underlying graph randomly 'activate' according to a sampling distribution at each timestep, and are thus discovered by the observer.

Since we are assuming a steady state with respect to measure M, under a consistent sampling process and enough data, we can treat the final aggregate static network obtained from a sequence of observations as a good approximation of the underlying system. We then specify an *edge sampling* model to randomly activate edges from the aggregate graph at each timestep of observations, simulating interactions (and thus observations) along an unchanging graph. By starting with a real dataset and specifying an edge sampling model, we can perform many Monte Carlo edge sampling simulations and determine the expected

trend over time in any graph theoretic property, as well as its sampling distribution, as a form of statistical permutation test (Good and Wang, 2005). Although extremely computer intensive when operating on large networks, permutation tests have a long history and are extremely powerful analytical tools.

In the next subsection, we describe the basic edge sampling model proposed in (Pedarsani et al., 2008). In Section 2.4.2, we describe four additional edge sampling models to satisfy the second part of Definition 2.4.1. Since permutation methods are very computationally intensive, we report results on a number of smaller datasets:

1. *Enron (internal).* All e-mail traffic between `@enron.com` e-mail addresses, quantized by month.
2. *IMDB Photos.* Celebrities photographer together, quantized by month.
3. *DBLP-FOCS.* A small sample of the DBLP digital bibliographic database, focusing on co-authorship in the Foundations of Computer Science conference. Note that the earlier years have significant missing data issues. The quantization timestep is one year.
4. *HEP-Th.* Citations in high-energy physics publications, quantized by month.

2.4.1 Uniform Edge Sampling

The basic edge sampling model assumes an underlying graph $G = (V, E)$, from which edges are 'activated' at each timestep uniformly and independently at random with a fixed probability p_e (Pedarsani et al., 2008). Vertices are only discovered as a consequence of an

edge activating; thus, isolated vertices of degree 0 are never discovered. We present a minor modification of the edge sampling model to a temporal setting here.

Let $v(t)$ and $e(t)$ be the number of nodes and edges that have been discovered from the underlying graph at time t, with $v(0) = 0$ and $e(0) = 0$. At each timestep $t \geq 1$, each edge in the underlying graph activates with fixed probability p_e independent of all other edges, and is therefore discovered if it has not previously activated. The number of activations of an edge over t timesteps is therefore binomially distributed as $Bin(t, p_e)$, and the probability that the edge does not activate at all after t timesteps (and thus remains undiscovered) is q_e:

$$q_e = (1 - p_e)^t$$

At time t, the expected number of discovered edges $e(t)$ is the mean of a second binomial distribution with probability $(1 - q_e)$ for E trials.

$$\begin{aligned} \mathbf{E}[e(t)] &= E \cdot (1 - q_e) \\ &= (1 - (1 - p_e)^t) \\ Var[e(t)] &= E \cdot (1 - (1 - p_e)^t)(1 - p_e^t) \end{aligned}$$

A node is discovered if one of its adjacent edges fires, and remains undiscovered if all of its edges do not fire. The probability of a node of degree d remaining undiscovered at each timestep is therefore q_e^d. After t timesteps, this probability is $(q_e^d)^t$. Conditioning on node

degree, where $f_D(d)$ is the proportion of nodes in the graph with degree d, the probability of a node not being discovered after t timesteps is given by:

$$q_v(t) = \sum_{d=1}^{V-1} q_e^{d \cdot t} \cdot f_D(d)$$

Thus, the expected number of nodes $v(t)$ at time t is:

$$\begin{aligned} \mathbf{E}[v(t)] &= V \cdot (1 - q_v(t)) \\ &= V \cdot (1 - \sum_{d=1}^{V-1} q_e^{d \cdot t} \cdot f_D(d)) \end{aligned}$$

Given the size (number of vertices and edges) and empirical degree distribution of an 'underlying' network, and a fixed sampling probability p_e, the edge sampling model lets us analytically determine the number of vertices and edges we can expect to see at each timestep in a dynamic discovery process. This, in turn, allows us to theoretically determine if we can expect to see edge densification for a given value of the sampling parameter p_e. It was shown that this model leads to densification in graphs with a power-law-like distribution (Pedarsani et al., 2008). Analytically determining more complex properties, such as the average shortest path length, quickly becomes difficult[1], but can be investigated using Monte Carlo analysis.

---

[1]As an example, the proof techniques used to analytically determine the diameter of a scale-free graph are far from trivial (Bollobás and Riordan, 2004).

2.4.2 Other edge sampling models

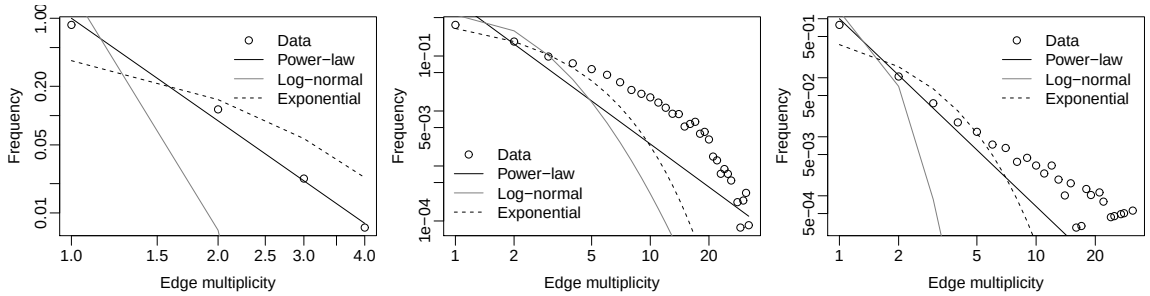
We describe four additional edge sampling models to be used for hypothesis testing. Each model attempts to preserve some statistic of the observed dataset.

2.4.2.1 Size-preserving edge sampling

The *size-preserving* (SP) edge sampling model samples $|E_t|$ edges from the static graph at each timestep, independently and uniformly at random without replacement, where $|E_t|$ is the number of edges actually observed in the dataset at time t. This model effectively stipulates that there is an unknown process governing the number of edges observed at each timestep (an observation or sampling process), but that within each timestep, edges are activated uniformly at random from the underlying network (the process governing interactions in the underlying system). Note that edges are chosen without replacement for generating the observation for a particular timestep, but with replcement across timesteps.

Figure 10 shows the distribution of edge multiplicity values (*i.e.,* , the distribution of the number of times each edge was observed in the dataset) for three datasets that exhibit edge multiplicity. A notable feature of all three distributions is their heavy skew: most edges are observed only once within the observation period, but a significant number are observed multiple times. In the same figure, we also show fitted distributions to the data: a power-law fit using the methods described by (Clauset et al., 2009), and exponential and log-normal distributions fitted using maximum likelihood estimators.

In the SP edge sampling model, each edge has the same probability of being chosen at a particular timestep, although with different probabilities across timesteps, so the distri-



(a) DBLP-FOCS (b) Enron internal (c) IMDB Photos

Figure 10: Distribution of edge multiplicity (also known as *support*) in various datasets, along with several common fitted distributions, shown on a doubly logarithmic scale. Although the data is heavily skewed, none of the distributions appear to be a good fit.

bution of edge multiplicities is Poisson-binomial (Wang, 1993), which is a general case of the binomial distribution. If E is the number of edges in the aggregate static network, and E_t the number of edges observed in the dataset at time t, then the probability of picking an edge at any timestep is E_t/E. Since this is generally small relative to the number of timesteps, the Poisson-binomial distribution is well-approximated by a Poisson distribution (Chen and Liu, 1997), which can appear as a skewed distribution qualitatively similar to the empirical distributions in Figure 10. Although a Poisson distribution does not yield a good fit to the data in Figure 10, it can serve as a useful first approximation.

2.4.2.2 Degree-preferential edge sampling

Similar to the preferential attachment growth model (Newman, 2001a), we propose the *degree-preferential* (DP) discovery model as a refinement of the SP edge sampling model. In the DP model, E_t edges are still picked at each timestep, but not uniformly at random.

Instead, the probability of an edge being sampled is directly proportional to the product of the degrees of its vertices, *i.e.,* edges between nodes prolific in connections are more likely to be re-activated. Specifically, the probability of sampling an edge $e = (u, v)$ at any timestep is given by:

$$p(e_{uv}) = \frac{d(u) \cdot d(v)}{\sum_{i \neq j} d(i) \cdot d(j)} \tag{2.3}$$

where $d(i)$ is the degree of vertex i. Algorithmically, sampling E_t edges without replacement from a graph according to Equation Equation 2.3 can be carried out efficiently using the weighted random reservoir sampling (WRS) algorithm described in (Efraimidis and Spirakis, 2006).

Intuitively, the DP model still stipulates some unknown process dictating the number of edges observed at each timestep, but a slightly different underlying interacting process that governs edge activations. In the DP model, being connected to a high degree vertex increases a vertex's chance of generating an activation at any given timestep. Furthermore, interactions between two connected high-degree vertices are more likely than between a pair of low-degree vertices.

2.4.2.3 Rate-preserving edge sampling

The *rate-preserving* (RP) edge sampling model samples each edge independently at each timestep with probability equal to its estimated probability from data, *i.e.,* the edge multiplicity divided by the total number of timesteps. This is perhaps the most realistic of the

edge sampling models presented here. Although it makes a strong independence assumption, it reproduces a number of statistics of the original dataset, such as the distribution of edge multiplicities. For an edge with multiplicity c in the dataset, its multiplicity under the RP model is a binomially distributed random variable with success probability c/t.

2.4.2.4 Size and Count-preserving edge sampling

Similar to the size-preserving sampling model, the *size and count-preserving* (SCP) sampling model preserves not just the number of edges observed at each timestep in the original dataset, but also the total number of activations of each edge, *i.e.,* edge multiplicity. Unlike the RP model, however, the multiplicity of each edge is a constant c instead of a random variable binomially distributed with mean c.

Preserving both the number of edges at each timestep as well as the total count of each edge is a non-trivial problem. We use a common Markov Chain Monte Carlo (MCMC) approach to perform this randomization, which has been previously used to randomize single graphs while preserving properties like the average path length (Hanhijärvi et al., 2009), and originally developed to generate random bipartite perfect matchings (Broder, 1986).

The approach involves representing each possible satisfying dynamic network as a node in a Markov chain with equiprobable transitions to all nodes that can be reached by performing a single *swap* operation. In our context, this operation swaps two distinct edges in different timesteps such that the total count of edges in each timestep remains the same before and after the swap. By iterating this Markov chain, we eventually reach the limiting

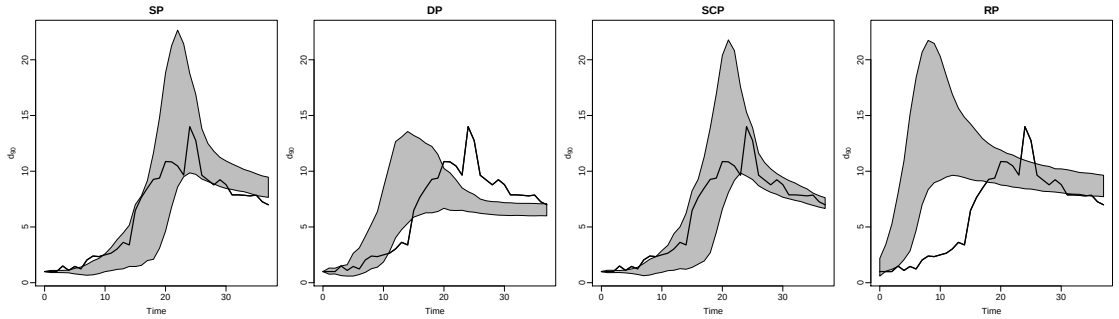
uniform distribution and have successfully randomized the network subject to the size- and count-preserving constraint.

In practice, the Markov chain is iterated until it mixes before being used, but it is difficult in general to tell when the limiting distribution has been reached; in (Hanhijärvi et al., 2009), a constant number of iterations was used. We run the chain forward for a number of iterations equal to the total number of edge occurrences before using the randomized network, and then use each randomized network as the starting point for subsequent randomizations.

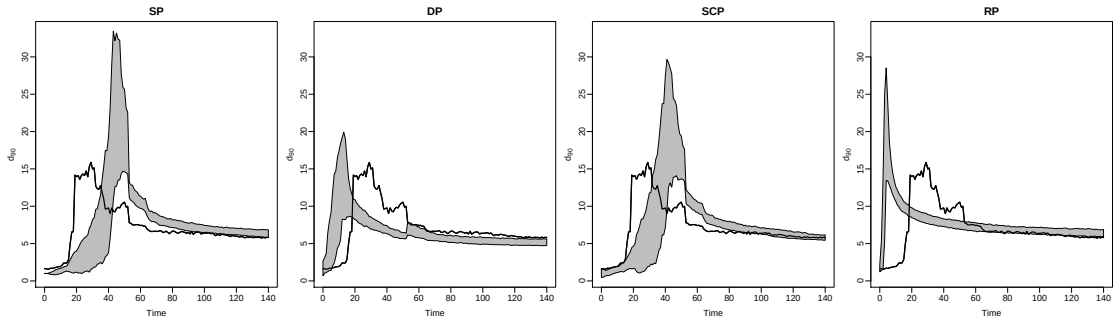
2.4.3 Empirical results

Recall the that the four discovery models we proposed were: rate-preserving (RP), size-preserving (SP), degree-preferential (DP), and size- and count-preserving (SCP). Given the original dataset, none of these edge sampling models require any parameters. We ran each model between hundreds and thousands of times (depending on the size of the dataset) on each of the five datasets described earlier. For each run, we sampled edges from the final aggregate static network for the same number of timesteps as the original dataset, and recorded the values of various graph properties at each timestep. From these values, we can estimate the mean property value at each timestep under a given edge sampling model, and thus the mean trend, as well as the variance and other statistics. Intuitively, if the edge sampling models are bad fits, the mean trend will be far from the actual observed trend.

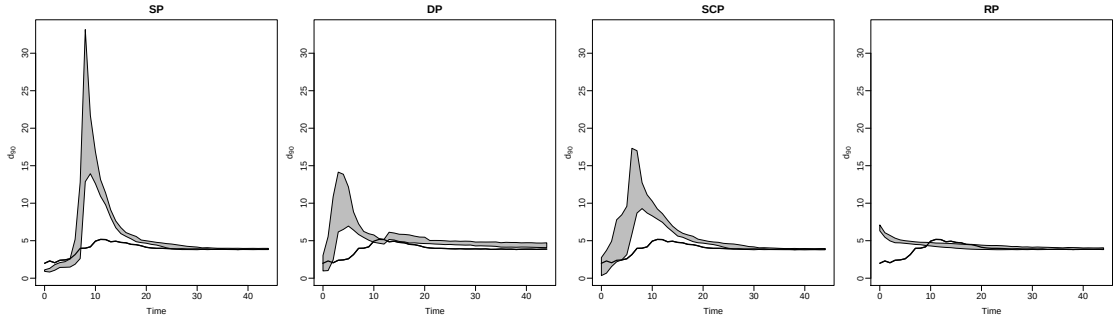
Figure 11 shows the mean effective diameter d_{90} of each edge sampling model, along with a band showing two standard deviations. Closed circles represent the original value of



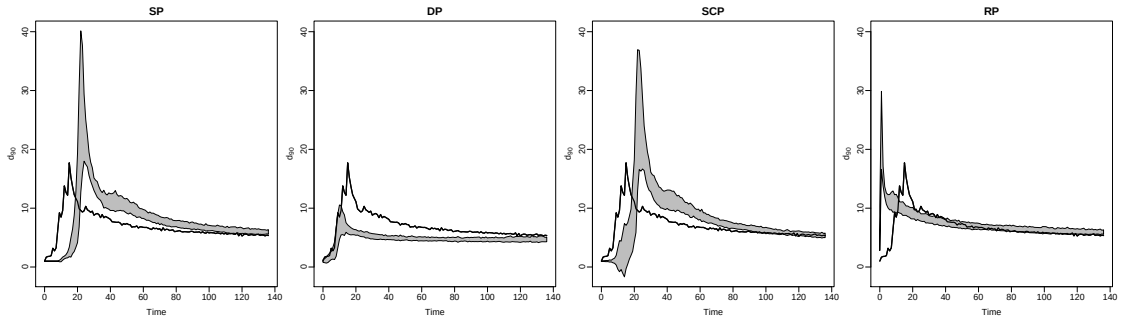
(a) DBLP-FOCS



(b) IMDB Photos



(c) Enron



(d) HEP-Th

Figure 11: The effective diameter over time of various datasets (solid line), and a band of two standard deviations around the expected trend under each edge sampling model.

the effective diameter measured in the dataset at each timestep. The DBLP-FOCS dataset, shown in Figure 11a, is a good example of a reasonable fit to an edge sampling model. The SP and SCP appear to be very good fits to the observed data. In both cases, the qualitative trend from the edge sampling model follows the trend observed in the data quite closely. Furthermore, points in original dataset are quite often within two standard deviations of the edge sampling trend. The DP model yields a slightly worse fit, with the d_{90} value ending higher than the observed data. Fits for the other datasets are not as close as the DBLP-FOCS dataset, but all show the same qualitative trend of a decreasing effective diameter.

Recall that one interpretation of the edge sampling model is that the underlying network is not changing at all; edges are merely being activated at each timestep and discovered. This is exactly the algorithm used for the sampling process – edges are sampled from a fixed, unchanging, underlying network. In spite of this, the trend suggested by the measurements is that the effective diameter in the underlying network is decreasing. The spurious trends are not insignificant either, and all follow the same decreasing trend, which might suggest a universal phenomenon when it is demonstrably an artifact.

2.5 Summary and suggestions

In this chapter, we described two common methods for measuring the properties of dynamic networks over time, and summarized the trends in common properties reported on multiple datasets. We also classified dynamic network data into two broad classes: *citation* networks and *interaction* networks, depending on whether the change in the structure of the

underlying process is either directly observed or measured through a proxy of interactions occurring on the network, with possibly independent dynamics. Most importantly, we systematically analyzed whether trends in common measures can be attributed to true change in the underlying system, or explained as either missing past artifacts (citation networks) or artifacts introduced by the dynamics of interactions (interaction networks).

There is an inherent bias in the growing network methodology for citation network measurement at short times, particularly in networks with a missing past. This bias can be understood as being caused by two sampling processes at play in the collection of network data: *re-discovery* of pre-existing vertices, and true *growth* in the underlying network. Prior studies assumed that all observations could be attributed to growth of the underlying network. In most cases, however, the sampling process is doubly stochastic as explained, with both discovery and growth playing a role. Very small amounts of missing past data can manifest as significantly erroneous trends in observations. Using synthetic data generated by well-known random graph models, we were able to show that: *densification (super-linear growth of edges relative to nodes) can manifest in networks that are not densifying, effective diameter can manifest as sharply decreasing in networks with a slowly increasing effective diameter, and these biases cannot be detected by withholding a portion of the observations.*

Similarly, trends in common measures in a number of real interaction network datasets can be explained using simple random models of interactions occurring along an unseen graph structure that we want to measure. The change in the graph is not directly observable, but we assume that the observation of an interaction indicates the existence of an

edge between the adjacent vertices. New interactions occurring along freshly created paths eventually reveal those paths to the observer, but conversely, there is no way to tell if a newly discovered path is actually new. Thus, we hypothesize that there might be little or no growth in the underlying network, and the observations are purely a product of discovering pre-existing vertices and edges. Using randomization tests, the expected trend under this hypothesis matched real data qualitatively, and in many cases, quantitively as well, within acceptable statistical bounds. Using the growing network method on an interaction network therefore reveals a trend that can also be interpreted as the convergence of measure M to its limiting value under a steady state hypothesis for M.

In the rest of this section, we focus on suggestions for future research.

2.5.1 Choosing an appropriate network representation

Recall that we had previously been able to classify network datasets into two categories: *interaction* networks, where the observations at any timestep represent instantaneous associations between vertices that can re-occur in the future, and *citation* networks, where observations represent the one-time formation of permanent links. We deal with each separately:

- *(Interaction networks)* A fully dynamic or dynamic interaction network with some form of smoothing or vertex and edge decay might be the most appropriate representation (see Section 2.1). If vertex or edge removal is a characteristic of the underlying system, then one of the following two methods could be used to reduce noise in the time series of observations:

1. *(Weighting)* Use a decay function to weight edges, and then either use weighted graph measures (Newman, 2001; Barrat et al., 2004; Newman, 2004; Zhang and Horvath, 2005; Barthelemy et al., 2005), or remove vertices and edges that have fallen below a threshold weight at any time[1] (Kossinets and Watts, 2006; De Choudhury et al., 2010).

2. *(Smoothing)* An alternative to weighting graphs is to expand the timescale to consist of larger timesteps in order to reduce noise. For example, if the observations of a network are at the resolution of a day, one could consider a dynamic network that aggregates a month of observations into a single timestep. Furthermore, one could consider a sliding window over this observation stream instead of a shifting-window quantization. A number of studies have looked at the effect of different timescales on network properties (Delvenne et al., 2010; De Choudhury et al., 2010; Eagle and Pentland, 2006), and there have been at least two algorithms proposed to find meaningful timescales for a time series of graphs (Sun et al., 2007; Sulo et al., 2010).

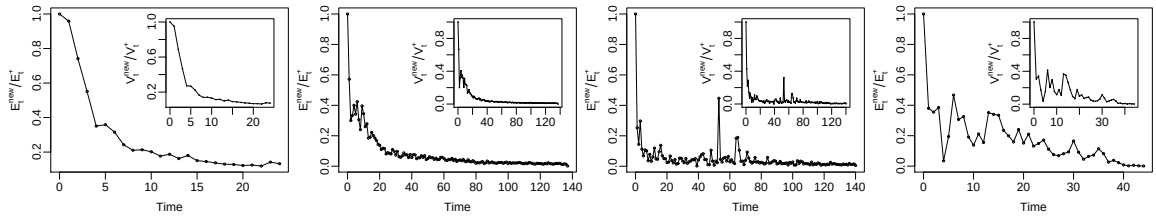
- *(Citation networks)* Although growing networks can be an appropriate representation for citation networks, the missing past issue introduces a vertex discovery process. In this case, we can only suggest that the network is allowed to stabilize to the point where new observations comprise a smaller portion of the aggregated network. This

[1] A recent arXiv working paper suggests that this method might be too noisy in general (Thomas and Blitzstein, 2011).

eliminates the noisy initial portion of the network when the discovery process is most likely to have a significant impact. Note that taking such a precaution is a heuristic; it does not solve the missing past problem.

Figure 12 shows the proportion of new edges and vertices introduced at each timestep as a function of the aggregate growing graph size at that timestep. As expected, new vertices and edges comprise a smaller portion of the aggregate graph as time progresses, so a simple heuristic might be to wait until new observations comprise no more than a small portion of the aggregate graph before considering measurements. This is similar in principle to the initial 'burn-in' period of many iterative statistical algorithms, where the output of an algorithm is discarded until the underlying model has reached stationarity (Hastie et al., 2001, p.280). Setting this threshold arbitrarily at 5% would entail dropping approximately the first 20 measured data points of the patent citations dataset (Leskovec et al., 2007), the first 40 timesteps of the HEP-Th co-authorship dataset (Leskovec et al., 2007), and the first 20 timesteps of the IMDB photos dataset (Lahiri and Berger-Wolf, 2008).

An entirely different solution would be to abandon the graph-theoretic representation of networks for one that is perhaps less sensitive to noise and missing past effects. As we mentioned earlier, spectral density plots have been proposed as a concise and effective way to look into the structure of networks (Banerjee and Jost, 2009), but doing so over time poses both a mathematical and visualization challenge. Alternatively, latent space models embed the nodes of a social network into a vector space, using edges to define some form of



(a) Patent citations (b) HEP-Th co-authorship (c) IMDB Photos (d) Enron internal

Figure 12: The number of previously unobserved edges and vertices (insets) at each timestep relative to the size of the aggregated network at that timestep for four real datasets.

distance function (Hoff et al., 2002). These models have been extended to dynamic networks as well (Sarkar and Moore, 2005), but represent a fundamentally different direction than classical graph theory.

2.5.2 Equilibrium assumption

Recall that a fundamental assumption in static network analysis is that the physical system is in equilibrium with respect to graph theoretic properties, so taking a large enough sample of the network yields representative approximations to the properties of the underlying system. On the other hand, one of the tenets of growing network analysis is that the properties of the underlying system are changing over time, as evidenced by the trends in graph theoretic properties over time.

As a telling example of this difference, consider two lines of research that use the same methodology, report essentially the same observations, but start with different assumptions and come to very different conclusions. Latapy and Magnien (Latapy and Magnien, 2006) ask how large a network sample must grow until the properties of the aggregated network

reach a steady state. Their methodology samples some networks over time, only because this is a natural order for those datasets, and shows graph-theoretic measurements that eventually converge to a fixed value. For measurements that converge, the authors conclude that the sampling process was effective and the steady-state values of the underlying network have been discovered. On the other hand, classic studies of growing networks use the same methodology to infer how the properties of the underlying network are *changing* over time (Barabási et al., 2002; Leskovec et al., 2005). In both cases, we see qualitatively identical plots of the same property (*e.g.,* average shortest path length over time in an aggregated network), but diametrically opposite inferences about the underlying network.

So are our observations successively converging to the underlying network, which is in a steady state, or are each of our observations representative enough to allow us to infer that the underlying network is changing? There is unlikely to be a general answer to this question, since one can imagine different situations in which each is plausible. The methods we have presented earlier in this chapter allow us to explicitly construct some of these situations. Thus, the equilibrium assumption, or lack thereof, needs to be justified. Consider the numerical analogy to growing network analysis shown in Figure 13. In each figure, a process is emitting uniformly distributed random numbers, either with a constant expectation or a trend as shown by the dotted line. The solid line shows the running average of the observations, similar to how growing networks accumulate observations into a single graph. The solid line at each time point represents the best estimate for the expectation of the process. In an alternative interpretation of the solid line, the expectation of the

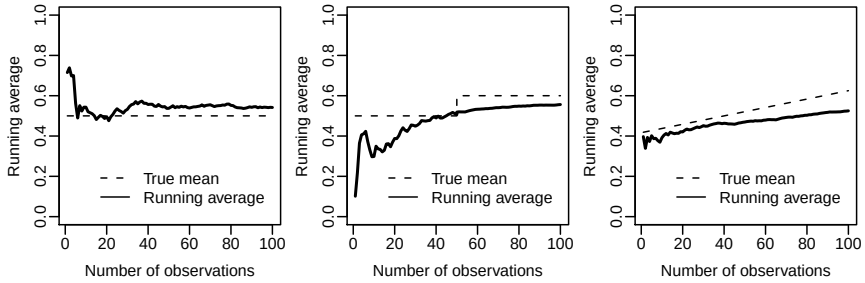


Figure 13: An illustration of the running average of a growing pool of random variates drawn from various uniform random distributions, as an analogy to the difficulty of detecting nonstationarity with the growing network methodology at short times. The task is akin to inferring the behavior, not the value, of the dotted line (ground truth) from a single growing pool of observations (solid line).

underlying process is either roughly constant (leftmost figure), or slowly growing (middle and right figures). However, the ambiguity of the setup and the actually measured solid line makes it difficult to infer the behavior of the dotted line. Neither is correct in all cases, so the observed measurements themselves cannot be used as a justification for assuming equilibrium or non-equilibrium at short times.

2.5.3 Dynamic measures

Finally, we note that interaction networks appear to be more common than citation networks. Since interaction dynamics can play a big part in our view of the underlying physical system, it might be advantageous to develop measures that directly extract information about the dynamics of the process, rather than measuring properties of a static structure at fixed time intervals. This is the motivation for the techniques developed in the remaining chapters of this thesis.

CHAPTER 3

MEASURING AND MINING PERIODICITY

In this chapter, we deal with the detection of a type of predictable behavior in such systems, namely periodically recurring interaction patterns in networks that change over time. Our goal is to detect periodic behavior even if it persists only for a short period of time, since such *locally periodic* behavior often holds a special meaning in real-world systems. As the simplest form of predictable behavior, periodic interaction patterns can indicate interesting relationships between the individuals involved in the interactions. Furthermore, with the right formal definition of what constitutes periodic behavior, the aggregate periodicities of an entire set of mined interaction patterns can yield insight about the global dynamics of the system being observed. We define the periodic pattern mining problem for dynamic networks as a step towards this goal, and describe an efficient algorithm to mine all such patterns from a stream of dynamic interaction data.

Part of the motivation for our focus on periodicities is the fact that streams of dense, time-varying interaction data are being collected in very diverse settings, and the first step in numeric signal processing is generally to take the Fourier transform of a signal to decompose it into a function of sinusoidal components, and subsequently to be able to analyze the spectrum of periodicities the signal contains. In this chapter, we aim to develop a similar tool for dynamic networks. As a motivating (but not unusual) example, ecologists often tag wild animals with GPS or proximity sensors to study behavioral and social associa-

tion patterns of the animals (Fischhoff et al., 2007; Sundaresan et al., 2007; Juang et al., 2002). This results in a continuous stream of interaction data, where periodically recurring patterns might correspond to seasonal or other recurrent association patterns. The same methodology has been used in human behavior experiments, with location-aware cellphones naturally replacing tracking collars (Eagle and Pentland, 2006), or human interactions being approximated by mobile phone or e-mail logs (Nanavati et al., 2006; Diesner and Carley, 2005; Chapanond et al., 2005) . Analyzing the local periodicities in such datasets presents opportunities for social science research, as well as commercial applications like recommender systems, traffic analysis and user modeling. The method presented in this chapter helps to answer two questions: what are the typical periodicities present in a dataset, and what are the specific interaction patterns that occur at these periodicities?

Our definition of the periodic pattern mining problem is specifically tailored for the analysis of dynamic networks, and is generic enough to handle all the situations just mentioned. It differs from earlier work in periodic pattern mining primarily in the use of two related concepts: (a) the concept of closed subgraphs, and (b) the principle of parsimony. Closed subgraph mining has been extensively explored in the context of a related problem of *frequent pattern mining* (Han et al., 2007). It draws from the areas of formal concept analysis and lattice theory to reduce redundancy in the definition of a frequent pattern, and thus reduces the potentially exponential (in the size of the input) number of output patterns that must be computed (Pasquier et al., 1999; Carpineto and Romano, 2004). The principle of parsimony is commonly known as Occam's Razor, and is a widely practiced guideline that

suggests favoring the simplest hypothesis that is consistent with a phenomenon. Combining these two concepts allows us to define periodic patterns in a way that avoids any redundant information, is more amenable to analysis, and allows the development of a provably efficient online mining algorithm. Furthermore, all the information contained in earlier definitions of periodic pattern mining is contained in ours in a more compact form, *i.e.,* the output of earlier algorithms can be deterministically generated from the output of our algorithm, but such a process would only add redundant information to the output.

We describe the *periodic subgraph mining* problem for dynamic networks, or more generally, for an arbitrary time series of structured interaction data. We draw on the concept of *closed subgraphs* from the related area of frequent pattern mining in order to mine coherent interaction patterns without redundancy. The theoretical framework of closed subgraphs allows us to derive an exact upper bound on the maximum number of patterns possible in any dynamic network, which subsequently allows us to develop a conceptually simple but powerful mining algorithm with polynomial worst-case time and space guarantees. The last point also underscores the fact that periodic subgraph mining in dynamic networks has inherently lower computational complexity than frequent pattern mining, which is also proved in this chapter. Another important aspect of our algorithm is the fact that it only requires a single scan of the data and heuristically accommodates patterns that are not perfectly periodic, which is what might be expected in real-world domains.

We demonstrate the usefulness of mining periodic patterns on four diverse real-world datasets. Mirroring the increasing diversity of network analysis domains, we examine

datasets of wild zebra association patterns, geographical movement patterns of university students, and the sightings of celebrities associated with the entertainment industry, among others. In addition to demonstrating the practical efficiency of our algorithm, we find that analyzing the collective periodicities of all mined patterns is indeed informative about the dynamics of the system being studied, yielding highly intuitive results about the specific systems we analyzed. We also found a number of interesting patterns which are intriguing because of a combination of their structure and periodicity. Some of these patterns occur relatively infrequently and might not have stood out had only their frequency of occurrence been considered, as is the case in frequent pattern mining. The fact that we can recover these structural patterns is an advantage over methods that deal with dynamic networks as tensors (Sun et al., 2006).[1]

This chapter is organized as follows. In the next section, we present some preliminary definitions related to dynamic networks, as well as some graph theoretic properties that are key to the inherent complexity of the problem. In Section 3.2, we formally define the mining problem, which incorporates the concepts of closed subgraphs and parsimony. This is followed by a discussion of related literature in Section 3.3. In Section 3.4, we analyze the inherent complexity of the problem and derive an exact upper bound on the maximum number of possible periodic subgraphs in any dynamic network. We show that the mining

---

[1]An alternative approach to detecting periodicities is to use the PARAFAC/CANDECOMP decomposition on a dynamic network and then use a conventional Fourier transform along the time dimension. This is an interesting topic for future research.

problem is in the computational complexity class P (polynomial), in contrast to the closely related frequent pattern mining problem (Boros et al., 2002; Yang, 2004). The complexity analysis of the problem is then used in Section 3.5 to build an efficient, online mining algorithm. The results of our experimental evaluation are presented in Section 3.6, followed by some concluding remarks and possible future research directions.

3.1 Preliminaries

Dynamic networks are a representation for a time series of interactions between a set of unique entities. Let $\mathcal{V} \in \mathbb{N}$ represent this set of entities. Interactions between entities can be either directed or undirected, and are assumed to have been recorded over a period of T discrete timesteps. The question of how much real time should constitute a timestep is beyond the scope of this thesis; we use natural quantizations specific to each of our datasets, such as one day per timestep. The only requirement is that a timestep should correspond to a meaningful amount of real time, as the periodicities of mined subgraphs will be in multiples of the chosen timestep.

Definition 3.1.1. (Dynamic network) A *dynamic network* $\mathcal{G} = \langle G_1, ..., G_T \rangle$ is a time-series of graphs, where $G_t = (V_t, E_t)$ is a simple graph of interactions E_t observed at timestep t among the subset of entities $V_t \subseteq \mathcal{V}$ at timestep t.

Figure 14 is an example of a dynamic network with five timesteps. Definition 3.1.1 implies a convenient graph theoretic property that reduces the high computational complexity of many algorithmic tasks on graphs: since a vertex represents a unique entity, each vertex v in a particular timestep's graph G_t has a *unique vertex label*. This constitutes a class

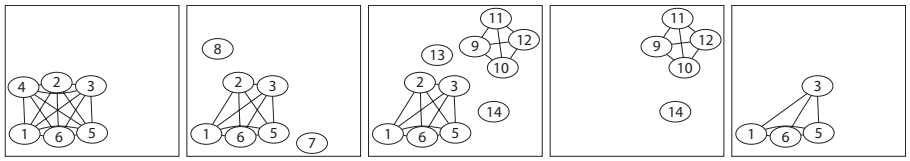


Figure 14: An example of a dynamic network with five timesteps.

of graphs that can be represented as sets of integers, resulting in a reduction to quadratic computational complexity (in the number of vertices) for certain hard graph problems, such as maximal common subgraph and subgraph isomorphism (Dickinson et al., 2003; Lahiri and Berger-Wolf, 2007; Lahiri and Berger-Wolf, 2008; Lahiri and Berger-Wolf, 2010).

Property 3.1.1. (Set Representation) *For a graph $G = (V, E)$ with unique vertex labels, the* set representation *R for G is formed by mapping each vertex and edge to a unique element in R, where $R \subset \mathbb{N}$.*

Since each vertex is uniquely identifiable by its label, it follows that each edge is also uniquely identifiable by its endpoints. This allows each vertex and edge to be coded as a unique integer, even across different graphs over the same vertex set. It can trivially be shown that two graphs (or timesteps) will result in the same set R if and only if they have identical vertex and edge sets. Although connectivity information is lost in the set representation, it is a useful transformation for the following algorithmic tasks, which are key to the development of our algorithm.

Property 3.1.2. (Subgraph Testing) *For two graphs G_1 and G_2 with unique vertex labels, testing whether G_1 is a subgraph of G_2 or vice versa is equivalent to checking whether the corresponding set representations R_1 and R_2 are subsets of each other. For this reason, we use the subset operator $\subseteq$ to denote a subgraph relationship between G_1 and G_2.*

Property 3.1.3. (Maximal Common Subgraph) *For a set f graphs with vertex unique labels, finding the* maximal common subgraph *(MCS) is equivalent to the maximal intersection of their set representations. For a set of graphs $G_1, ..., G_T$, a vertex or an edge is part of the MCS if it is part of every G_t. As a result, the maximal common subgraph always exists, is unique and well-defined, but could possibly be the empty graph with no vertices or edges. We use the intersection operator $\cap$ to denote the maximal common subgraph of two or more graphs.*

Property 3.1.4. (Hashing) *A hashing function exists for graphs since the set representation R has a global ordering by virtue of $R \subset \mathbb{N}$.*

Figure 15 demonstrates the use of Property 3.1.1 to calculate the maximal common subgraph of two graphs using set representation. A further implication of the set representation is that a dynamic network can be represented as a transaction database (also known as 'market-basket' data (Agrawal and Srikant, 1994)) for certain data mining tasks like frequent subgraph mining[1] (Inokuchi et al., 2000; Kuramochi and Karypis, 2001). Al-

[1]Since connectivity information is lost in the set representation, frequent *connected* subgraphs and subgraphs with other specific graph-theoretic properties cannot be extracted from the set representation.

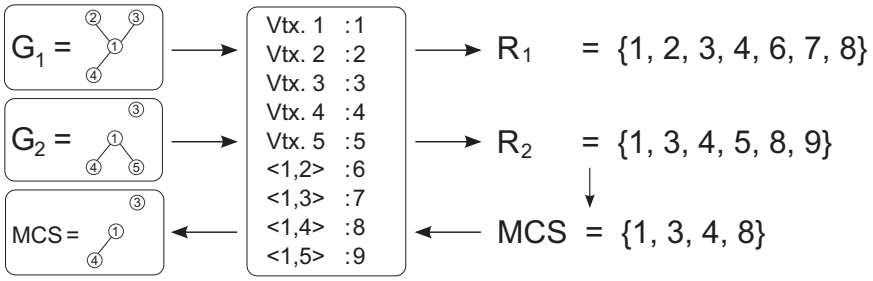


Figure 15: The correspondence between graph and set representations for graphs with unique vertex labels. The example demonstrates the computation of the maximal common subgraph of two graphs using set representation.

though mining for periodic patterns in time-ordered transaction databases has been studied in different contexts (Özden et al., 1998; Han et al., 2007; Han et al., 1999; Yang et al., 2001; Huang and Chang, 2005), one of the main advantages of our framework is the ability to handle structured data like dynamic networks (with connectivity information) while also being applicable to unstructured data like transaction databases.

We now introduce some terminology from the frequent pattern mining problem to be used in our problem definition and analysis.

Definition 3.1.2. (SUPPORT) Given a dynamic network $\mathcal{G}$ of T timesteps and an arbitrary graph $F = (V, E)$, the *support set* $S(F)$ of F in $\mathcal{G}$ is the set of all timesteps t in $\mathcal{G}$ where F is a subgraph of G_t, which we denote $F \subseteq G_t$. The *support* of F is the cardinality of its support set, $|S(F)|$:

$$S(F) = \{t_i, ..., t_j\} \quad \text{such that} \quad \forall t \quad (t \in S(F) \leftrightarrow F \subseteq G_t).$$

Definition 3.1.3. (FREQUENT SUBGRAPH) Given a dynamic network $\mathcal{G}$ of T timesteps, an arbitrary graph $F = (V, E)$ is *frequent* if its support exceeds a user-defined *minimum support threshold* $\sigma \leq T$.

Definition 3.1.3 is the basis of the well known frequent pattern mining problem, which deals with the extraction of all subgraphs F where $|S(F)| \geq \sigma$. An implication of the naïve definition of a frequent subgraph is the *downward closure* property, which states that every subgraph of a frequent subgraph F is itself frequent. This serves as the underpinning of Agrawal and Srikant's classic *Apriori* algorithm, which searches for large frequent patterns by iteratively concatenating the smaller, frequent sub-patterns implied by the downward closure, relying on the sparsity of larger frequent patterns (Agrawal and Srikant, 1994). The downward closure is what makes a principled, incremental search through pattern space tractable, but is also a double-edged sword. Although many improvements have been made to the classic *Apriori* algorithm (Han et al., 2007; Cheng et al., 2008), any mining algorithm required to explicitly enumerate every frequent pattern in a dataset would, in doing so, have to enumerate the exponential number of subgraphs of every frequent subgraph which is a redundant and resource expensive process. The cornerstone of a solution to this problem is the use of closed subgraphs (Pasquier et al., 1999; Carpineto and Romano, 2004; Han et al., 2007).

Definition 3.1.4. (CLOSED SUBGRAPH) Given a dynamic network $\mathcal{G}$ of T timesteps and an arbitrary graph $F = (V, E)$, F is *closed* if it is maximal for its support set: no vertex or edge can be added to F while maintaining its support.

Mining frequent closed subgraphs is an elegant solution to the redundancy of the general frequent pattern mining problem. It captures all the information of the more general formulation, but can result in output that is exponentially smaller in size without any loss of information. We therefore adopt it as an integral part of our problem definition, which is described in the next section.

3.2 Problem Definition

We formally define the periodic subgraph mining for dynamic networks as a special case of frequent closed pattern mining with important additional computational properties. These properties allow the development of efficient mining algorithms and justify an independent treatment of the problem, rather than an approach that would, for example, push constraints into a conventional frequent pattern mining algorithm (Pei and Han, 2000; Garofalakis et al., 1999; Pei et al., 2002; Zhu et al., 2007). The relation to frequent pattern mining also highlights the fact that we are searching for locally periodic patterns, *i.e.*, those that exhibit periodic behavior in a contiguous subsequence of the entire data stream. These are also known as *partially periodic* patterns (Han et al., 1999; Ma and Hellerstein, 2001; Huang and Chang, 2005). We begin with a basic formulation of the problem and then develop it into a parsimonious formulation. We end this section by describing mechanisms to rank periodic patterns and handle imperfect periodicity in real-world datasets.

3.2.1 Basic Formulation

Definition 3.2.1. (Periodic support set) Given a dynamic network $\mathcal{G}$ and an arbitrary subgraph $F = (V, E)$, a *periodic support set* of F in $\mathcal{G}$, denoted $S_P = (i, p, s)$, is a maximal,

ordered set of s timesteps starting at t_i with every two consecutive timesteps being p steps apart.

$$S_P = (i, p, s) = \langle t_i, t_{i+p}, ..., t_{i+p(s-1)} \rangle$$

subject to the following constraints:

1. *Existence in $\mathcal{G}$:* F must exist at all timesteps in S_P, *i.e.,* $\forall t$ $(t \in S_P \rightarrow F \subseteq G_t)$. Note that the implication in the constraint is only in the forward direction, unlike Definition 3.1.3.
2. *Minimum size:* A periodic support set has to have at least two elements, *i.e.,* $|S_P| = s \geq 2$.
3. *Temporal maximality:* The support set cannot be extended in time to contain F and still be periodic, *i.e.,* $F \not\subseteq G_{t_{(i-p)}}$ and $F \not\subseteq G_{t_{(i+p \cdot s)}}$.

The *phase offset* of a periodic support set is defined as $m = (t_i - 1)$ **mod** p, since indices start from 1. Thus, $0 \leq m < p$.

A key difference in the definitions of a support set for frequent pattern mining and periodic pattern mining is that a single graph F can have multiple periodic support sets to allow for multiple, disjoint, or overlapping periodic behavior. Thus, we require the extraction of all periodic subgraph *embeddings*, rather than just the periodic subgraphs themselves. This is encompassed in the following definition.

Definition 3.2.2. (Periodic subgraph embedding) Given a dynamic network $\mathcal{G}$, a *periodic subgraph embedding (PSE)* is a pair $\langle F, S_P \rangle$, where F is an arbitrary graph that

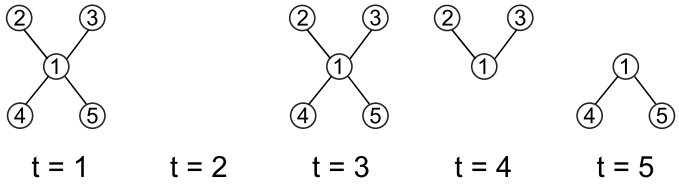


Figure 16: An example of a dynamic network with 2 PSEs at $\sigma = 3$.

is closed over a periodic support set S_P with $|S_P| \geq \sigma$. The following list summarizes the properties of a PSE:

1. *Minimum support:* $|S_P| \geq \sigma \geq 2$, from Definition 3.2.1.
2. *Structural maximality:* F is maximal over S_P, *i.e.* F is the maximal common subgraph of S_P, from Definition 3.1.4.
3. *Temporal maximality:* S_P is temporally maximal for F, from Definition 3.2.1.

Figure 16 shows an example of a dynamic network with two PSEs at $\sigma = 3$. The first is the subgraph $\{(1,4),(1,5)\}$ with a period of 2 and support set of $\langle 1,3,5\rangle$, and the second is the singleton vertex $\{1\}$ with a period of 1 and a support set of $\langle 3,4,5\rangle$. Note that the subgraph $\{(1,2),(1,3)\}$ is frequent but not periodic at $\sigma = 3$.

3.2.2 Parsimonious Formulation

We now address the issue of redundant information in the output. If we think of a PSE from Definition 3.2.2 as communicating a set of timesteps at which a particular subgraph exhibits periodic behavior, a PSE which communicates information that is already contained in another PSE is redundant. For example, a subgraph F of period 2 with adequate support

will also be output as a subgraph of period 4, and so on. This will continue for a fixed number of multiples of the base period, depending on the support of the pattern and the minimum support, in spite of the fact that the higher multiples communicate no new information about the subgraph in question. Furthermore, when analyzing periodic behavior in terms of the periodicities of mined patterns, there is no justifiable reason *prima facie* (or in keeping with Occam's Razor) to count multiples of a base pattern's period, unless those multiples extend beyond the support of the base pattern.

Although the use of closed subgraphs reduces much of the redundancy associated with the output of an *Apriori* style algorithm, the basic definition of a PSE still retains some of it. To eliminate all such redundancy, we pose our problem as that of mining a *minimal set of patterns* to cover all periodic occurrences of all periodic subgraphs. Keeping in line with the principle of parsimony, this eliminates patterns with periods that are multiples of a base period, unless they convey some new information about a periodic occurrence. In order to describe this concept formally, we first define the notion of *subsumption* of PSEs.

Definition 3.2.3. (SUBSUMPTION) For two periodic subgraphs F_1 and F_2 with respective periodic support sets $S_{P,1} = (i_1, p_1, s_1)$ and $S_{P,2} = (i_2, p_2, s_2)$, $\langle F_1, S_{P,1} \rangle$ completely contains or *subsumes* $\langle F_2, S_{P,2} \rangle$ if all of the following conditions hold:

1. $F_2 \subseteq F_1$
2. $t_{i_2} \geq t_{i_1}$
3. $t_{i_2 + p_2 \cdot (s_2 - 1)} \leq t_{i_1 + p_1 \cdot (s_1 - 1)}$

4. $p_2 = k \cdot p_1$ for some integer $k > 0$

5. $t_{i,2} = t_{i,1} + l \cdot p_1$ for some integer $l \geq 0$

We prove that all conditions listed above are necessary for subsumption. Condition 1 is trivially required to ensure that no information is lost. Let $f_1(l) = t_{i,1} + l \cdot p_1$ and $f_2(l) = t_{i,2} + l \cdot p_2$ be the l^{th} occurrence of F_1 and F_2 respectively, for some integer l. For subsumption, we require that the support set $S_{P,2}$ is completely contained within the support set $S_{P,1}$. Conditions 2 and 3 require that the support set of of F_2 is contained within the bounds of the support set of F_1, although they could be of different phase offsets and not overlapping at all, or partially overlapping but of different periods. Condition 4 requires that the period of F' is an integer multiple of F, and condition 5 requires that F_1 and F_2 have compatible phase offsets, which ensures that they overlap. This is handled by requiring that the first occurrence of F_2 overlap with any occurrence of F_1. Thus, $t_{i,2} = f_1(l)$, which yields the final condition $t_{i,2} = t_{i,1} + l \cdot p_1$.

Definition 3.2.4. (Parsimonious PSE) A PSE that is not subsumed by any another PSE is a *parsimonious periodic subgraph embedding* (PPSE).

As an example to motivate the mining of PPSEs, consider a system in which all the nodes only interact periodically with either period 2 or 4, starting at arbitrary times and continuing for an arbitrary number of repetitions. Suppose that we want to discover these unknown periodicities by observing the system for a period of time. With non-parsimonious PSEs, duplicates of each true periodic pattern would be reported for a fixed number of

multiples of either 2 or 4, depending the specific pattern. If we were to plot a histogram of the periodicities of all mined patterns, we would see various artifacts from the higher order periodicities, which could obscure the true periodicities. On the other hand, with parsimonious PSEs and enough data, the true periodicities of 2 and 4 would, with high probability, be the most prominent peaks.

Definition 3.2.5 (*Periodic Subgraph Mining Problem)*. Given a dynamic network $\mathcal{G}$ and a minimum support threshold $\sigma \geq 2$, the Periodic Subgraph Mining problem is to list all parsimonious periodic subgraphs embeddings in $\mathcal{G}$ that satisfy the minimum support.

3.2.3 Practical Considerations

3.2.3.1 Handling noise by smoothing

Since real-world networks are unlikely to always contain perfectly periodic patterns, we use smoothing as a mechanism for accommodating imperfect periodicity. Given a user-defined smoothing parameter $S \geq 1$, we transform the dynamic network by considering a sliding window over its timesteps. In other words, we transform the dynamic network $\mathcal{G}$ in the following manner[1], where $G_i \in \mathcal{G}$:

$$\mathcal{G}' = \langle G_1 \cup ... \cup G_S, G_2 \cup ... \cup G_{S+1}, ... \rangle$$

---

[1]Blank timesteps are appended to the beginning and end of the dynamic network as necessary to handle boundary conditions.

In addition, the following two conditions handle the removal of artifacts introduced by the smoothing process.

1. The minimum period $P_{\min}$ is set to S.
2. PSEs of the same subgraph that share the same period and differ in their starting positions by at most $S-1$ timesteps are merged. In other words, the PSE with the highest support is retained. This can be done as a post-processing step or incorporated into the mining algorithm itself.

By introducing this smoothing mechanism, we allow a window of timesteps within which the order of events does not matter. No smoothing is performed at $S = 1$.

3.2.3.2 Purity: a measure for ranking periodic subgraphs

A periodically recurring subgraph is not necessarily representative of an interaction pattern that occurs *only* periodically, as shown in Figure 17. The *purity* measure expresses how likely it is that a periodic subgraph embedding occurs only periodically over its support set.

Definition 3.2.6. (PURITY) Given a periodic subgraph embedding $\langle F, S_P \rangle$ with period p, starting at timestep t_i and with support $s = |\langle t_i, ..., t_j \rangle|$, the purity of F is the ratio of its periodic support to its total support in the timestep range $[t_i, t_j]$.

$$purity(F) = \frac{s}{|\{t : F \subseteq G_t, t_i \leq t \leq t_j\}|}$$

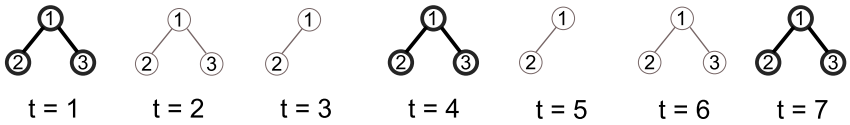


Figure 17: A periodic subgraph embedding (bold) with non-periodic occurrences. The purity of this periodic subgraph is 3/5, whereas its average purity is $\frac{1}{2}(\frac{3}{5} + \frac{3}{7}) \sim 0.51$.

It is sometimes advantageous to define the purity of a subgraph as the average purity of its edges. Doing so is more representative of the temporal characteristics of the entire subgraph. We use the term 'purity' to refer to average purity for the remainder of this chapter. Figure 17 shows an example of the purity measure.

Definition 3.2.7. (AVERAGE PURITY) The *average purity* of a subgraph $F = (V, E)$ is the average purity of all of its edges.

$$avgPurity(F) = \frac{1}{|E|} \sum_{e \in E} purity(e)$$

3.3 Related Work

Searching for periodicity and periodic patterns have appeared in different contexts in data mining. In this section, we review relevant literature concerning periodic pattern mining, as well as the closely related problem of frequent pattern mining. We omit certain earlier antecedents to this line of research, such as mining cyclic association rules (Özden et al., 1998) and frequent sequential patterns (Agrawal and Srikant, 1995), as they are not directly relevant. Also omitted for the same reason are periodic pattern mining approaches

that require or assume that the entire input is at least approximately periodic, including techniques that use Fast Fourier Transforms (Elfeky et al., 2005a; Elfeky et al., 2005b).

Most algorithms for mining periodic patterns deal with unstructured data such as a sequence or multiple, aligned sequences. In the most general formulation of the problem, the input consists of a sequence of symbols sets $S = \langle a_1, ..., a_T \rangle$, where each symbol set a_i is drawn from a finite universal set $\mathcal{L}$. A pattern is a sequence $P = \langle b_1, ..., b_p \rangle$ of length p, where p is the period of the pattern and each $b_i \subseteq \mathcal{L} \cup \{*\}$. The '*' character is a wild card that matches any symbol. Less general versions consider only a single sequence as the input, so each $a_i \in \mathcal{L}$ and $b_i \in \mathcal{L} \cup \{*\}$. The pattern mining problem is to extract all such patterns from the input sequence, subject to constraints such as a minimum support. Algorithms for this task are generally variants of the classic *Apriori* algorithm of Agrawal and Srikant (Agrawal and Srikant, 1994), in which larger patterns are iteratively built from smaller ones. Note that the definition of a periodic pattern in this line of research is essentially a sequence with wildcards, whereas our definition is closer to concepts from frequent pattern mining.

Han et al. introduced one of the first algorithms to mine partial periodic patterns in multidimensional sequences (Han et al., 1999). They adopt an Apriori-inspired search through pattern space using a novel prefix-based data structure called a *max-subpattern tree*. Ma and Hellerstein (Ma and Hellerstein, 2001) propose a similar, Apriori-inspired approach consisting of two level-wise algorithms for mining periodic patterns in the presence

of both partial periodicity as well as imperfect periodicity. They also propose an interesting statistical (as opposed to combinatorial) foundation for defining periodicity.

Yang et al. (Yang et al., 2001; Yang et al., 2002) proposed another level-wise mining algorithm for detecting 'surprising' periodic patterns, i.e. those judged to be interesting based on deviation from their expected frequency. This is intended to overcome limitations of using the support of a pattern as the sole measure of its worth. They devise two variants of information gain as measures of interest: *bounded information gain* (Yang et al., 2001) and *generalized information gain* (Yang et al., 2002), the second of which obeys the triangle inequality. However, a number of independence assumptions are made, such as the probability of occurrence of an event being the same at any point in time, and these might not hold in dynamic networks.

Yang et al. (Yang et al., 2003) propose a level-wise mining algorithm that allows imperfect (or 'asynchronous') periodic patterns to be discovered. They do this by introducing two user-defined parameters into the mining process to specify the minimum number of repetitions of a pattern and the maximum amount of disruption allowed. Huang and Chang (Huang and Chang, 2005) build on this in their description of SMCA, a suite of four algorithms for mining periodic patterns (Huang and Chang, 2005). The fundamental idea is still to conduct a level-wise search through pattern space, but augmented with more efficient data structures and algorithms than earlier approaches. Each algorithm enumerates more complex patterns from the output of an earlier stage.

Finally, our work is inspired by frequent pattern mining, which is concerned with the discovery of patterns that occur more frequently than a user-defined threshold. A relatively young offshoot of this line of research is frequent subgraph mining (Inokuchi et al., 2000; Kuramochi and Karypis, 2001), which was originally devised to search for common structures in databases of chemical compounds represented as graphs. A detailed overview of this field is beyond the scope of this thesis, but may be found in (Han et al., 2007) and (Cheng et al., 2008). There are, however, a number of recent complexity results for frequent pattern mining that are relevant. Specifically, given a set of maximal frequent itemsets, Boros et al. (Boros et al., 2002) show that it is NP-complete to decide if there is a further maximal frequent itemset. Yang (Yang, 2004) shows that different variants of maximal frequent pattern mining, including itemsets and subgraphs with unique vertex labels, are either #P-hard or #P-complete in terms of counting the number of satisfying solutions. Thus, many variants of frequent pattern mining are computationally intractable in the worst case.

3.4 Complexity Analysis of the Mining Problem

We now analyze the computational complexity of the periodic subgraph mining problem as defined in Section 3.2. In order to do this, we derive an exact upper bound on the number of PSEs that can exist in any dynamic network of T timesteps. We prove that this upper bound is a polynomial function of the number of timesteps and the minimum support value.

We show that the upper bound is sharp by constructing a 'worst-case' dynamic network.[1] The proof leads to the conclusion that mining all closed PSEs can be done in polynomial time in the size of the input, proving that the mining (enumeration) problem is in the complexity class P, when the graphs have unique vertex labels. This is in contrast to the more general frequent subgraph mining problem, which is NP-hard for enumeration and #P-complete for counting, even with unique vertex labels (Yang, 2004; Boros et al., 2002). We take advantage of the intrinsic polynomial complexity of the problem to design an efficient single-pass mining algorithm in Section 3.5. We do not include smoothing in the following analysis, and purely algebraic manipulations are omitted for brevity.

Theorem 3.4.1. Periodic Subgraph Mining *in dynamic networks is in P.*

To prove Theorem 3.4.1, we first construct a class of worst-case dynamic networks and show that any member of this class has the maximum possible number of PSEs. We utilize the concept of a *projection* of a discrete time sequence to count the maximum number of PSEs in this class of dynamic networks (Elfeky et al., 2005a).[2]

Definition 3.4.1. Given a dynamic network $\mathcal{G}$, a *projection* $\pi_{m,p}$ of $\mathcal{G}$ is a subsequence of graphs

$$\pi_{m,p} = \langle G_{1+m}, G_{1+m+p}, G_{1+m+2p}, ... \rangle,$$

---

[1] An alternate version of this proof in terms of maximal subgraphs, but with the same outcome, can be found in (Lahiri and Berger-Wolf, 2008).

[2] In principle, any combinatorial technique can be used to count the number of PSEs. Projections are convenient for Definition 3.2.2 and some extensions to it.

where p is the *period* of the projection and $0 \leq m < p$ is the phase offset.

It should be clear from the definitions of periodicity and projection that any periodic support set at minimum support σ is embedded in at least σ consecutive positions of some projection $\pi_{m,p}$.

Proposition 3.4.1. *Let F be the maximal common subgraph of any $s \geq \sigma$ consecutive positions of any projection $\pi_{m,p}$. If F is not empty, then it is a periodic subgraph and the s consecutive timesteps from $\pi_{m,p}$ are part of a PSE for F.*

Proof. A non-empty maximal common subgraph F of any $s \geq \sigma$ consecutive positions implies that F is maximal over a support set of at least σ periodic timesteps, which in turn might or might not be temporally maximal for F. However, in either case, the s timesteps are part of some valid periodic support set of size at least σ. This is a sufficient condition to satisfy Definition 3.2.2, and thus F is a periodic subgraph. □

Corollary 3.4.1. *In the worst computational complexity case for mining periodic subgraph embeddings in a dynamic network, the maximal common subgraph of every $s \geq \sigma$ consecutive positions of every projection is not empty and contains a unique PSE.*

Proof. Clearly, if every periodic subset of $s \geq \sigma$ timesteps of the dynamic network contains a unique maximal common subgraph, then they all need to be enumerated by any mining algorithm and it is indeed the worst case input for a periodic subgraph mining problem. We now show that it is attainable using an explicit construction. We place a different edge in each $s \geq \sigma$ consecutive positions of every projection to ensure that each edge is part of

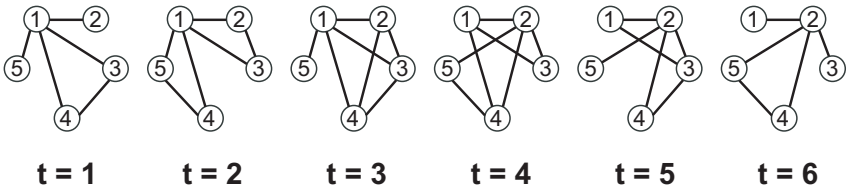


Figure 18: An example of a worst case dynamic network for mining PSEs at $\sigma = 3$.

a unique periodic subgraph embedding. Let edge e be created in this way with support set S_P in some $\pi_{m,p}$. Considering only S_P, we know that it is temporally maximal for the edge e because e does not exist in any other timesteps. Furthermore, the maximal common subgraph of S_P is non-empty because it contains at least the edge e. Thus, each edge is part of a unique PSE whose support set is S_P. Since a different edge was placed in every $s \geq \sigma$ consecutive positions of every projection, the number of PSEs is equal to the number of edges created. No additional PSEs can be created since every permissible support set, *i.e.* with support greater than σ, is already part of a unique PSE. Therefore, the described structure is a worst case instance for its size. □

Figure 18 shows an example construction of such a worst-case dynamic network with 12 PSEs at $\sigma = 3$. The next step is to explicitly calculate the upper bound on the number of PSEs in the worst-case network instances. Following from Corollary 3.4.1, we only need to count the number of $s \geq \sigma$ consecutive positions of every projection to derive this bound. In order to do this, we first state the bounds on several other parameters.

Proposition 3.4.2. *In a dynamic network with T timesteps, the* maximum period *of any periodic subgraph with support at least σ is $P = \lfloor (T-1)/(\sigma-1) \rfloor$.*

Proposition 3.4.3. *In a dynamic network with T timesteps, the* length of any projection *is $|\pi_{m,p}| = \lceil (T-m)/p \rceil$.*

The proofs of the Propositions 3.4.2 and 3.4.3 are straightforward and similar to those in (Elfeky et al., 2005a). Given the above expressions, we now derive an exact bound by construction.

Theorem 3.4.2. *In a dynamic network with T timesteps, there are at most $O(T^2 \ln \frac{T}{\sigma})$ closed PSEs at minimum support σ.*

Proof. From Corollary 3.4.1, the maximum number of PSEs possible in a dynamic network at minimum support σ is equal to the number of $s \geq \sigma$ length windows over all possible projections of the network. For a given projection $\pi_{m,p}$ and value of s, it is clear that the number of length-s windows over the projection is $|\pi_{m,p}| - s + 1$, where $|\pi_{m,p}|$ is the length of the projection defined in Proposition 3.4.3. Thus, for a given value of s, the number of length-s windows over all projections can be obtained by substituting the expressions from Propositions 3.4.2 and 3.4.3:

$$\sum_{p=1}^{\lfloor \frac{T-1}{s-1} \rfloor} \sum_{m=0}^{p-1} \left(\left\lceil \frac{T-m}{p} \right\rceil - s + 1 \right)$$

We have replaced σ with s in the expression for the maximum period of a pattern from Proposition 3.4.2, since we only want projections which contain at least one length-s

window for any s. This constitutes the outer summation; the inner summation is over all possible phase offset values m for a given period p. Finally, the term inside the summation is the number of length-s windows in any projection, where $|\pi_{m,p}|$ has been substituted from Proposition 3.4.3. We now sum this expression over all possible values of s, which run from σ to T, and relax the floor and ceiling expressions for an asymptotic closed form approximation.

$$\sum_{s=\sigma}^{T} \sum_{p=1}^{\lfloor \frac{T-1}{s-1} \rfloor} \sum_{m=0}^{p-1} \left(\left\lceil \frac{T-m}{p} \right\rceil - s + 1 \right) \tag{3.1}$$

$$\sim \sum_{s=\sigma}^{T} \sum_{p=1}^{\frac{T-1}{s-1}} \sum_{m=0}^{p-1} \left(\frac{T-m+p}{p} - s + 1 \right) \tag{3.2}$$

Expression Equation 3.2 algebraically simplifies to an expression that is $O(T^2 \cdot H(\frac{T-1}{\sigma-1}))$, where $H(n) = \sum_{k=1}^{n} \frac{1}{k}$ is the nth harmonic number, asymptotically approximated by $\ln n$. Thus, the number of PSEs at minimum support σ is bounded asymptotically by $O(T^2 \ \ln \frac{T}{\sigma})$ (and exactly by Equation Equation 3.1). □

Proof of Theorem 1. To finally prove Theorem 3.4.1, consider an algorithm that outputs the maximal common subgraph of every σ length window of every projection. Since the maximal common subgraph of a set of graphs with unique vertex labels can be found in time $O(V + E)$ (Dickinson et al., 2003), in the worst case, this results in $O(T^2 \ln \frac{T}{\sigma})$ periodic 'fragments' computed in $\Theta((V+E)T^2 \ln \frac{T}{\sigma})$ time. Every pair of periodic fragments is then compared and merged if they represent overlapping embeddings of the same periodic subgraph, in time $O((V + E)(T^2 \ln \frac{T}{\sigma})^2)$, resulting in all PSEs. Another run over pairs of

PSEs can eliminate all non parsimonious PSEs, resulting in an overall time complexity of $O((V+E)T^4(\ln \frac{T}{\sigma})^2)$. Thus, the mining problem is in P, and the exact bound on the number of closed PSEs is given in summation form in Theorem 3.4.2. □

3.5 The Algorithm

We now present PSEMiner[1], our algorithm for mining all parsimonious periodic subgraph embeddings (PPSEs) in a dynamic network. We start by describing the most basic form of the algorithm, which mines closed (not just parsimonious) periodic subgraph embeddings, and proving its correctness and complexity. We then describe some simple optimizations to the basic algorithm that allow it to output only PPSEs and also improve its efficiency in practice.

PSEMiner is based on the following idea: as each timestep of the dynamic network is read, we maintain a list of all periodic subgraph embeddings seen up to timestep t. This list is maintained in a simple data structure called a *pattern tree*, which also tracks subgraphs that might become periodic at some point in the future. Once PSEs cease to be periodic, they are flushed from the tree and written to the output stream if they satisfy certain conditions like the minimum support. As each timestep G_t is read from the data stream, the pattern tree is updated with the new information, which could involve modifying, adding and deleting tree nodes. The complexity analysis in Section 3.4 allows us to prove worst-case computational time and space bounds that are polynomial in the size of the input.

[1]Periodic Subgraph Embedding Miner

We describe the algorithm, its parameters, data structures and a proof of correctness in the following five sections. In Section 3.5.6, we describe optimizations that complete the description of the algorithm.

3.5.1 Parameters

Our algorithm is a single-pass, polynomial time and space algorithm for mining all closed periodic subgraph embeddings in a dynamic network. It does not require any parameters, but *optionally* accepts the following:

1. Minimum support threshold $\sigma \geq 2$ (default: 2).
2. Minimum period $P_{\min}$ (default: 1).
3. Maximum period $P_{\max}$ (default: unrestricted).
4. Smoothing timesteps $S \geq 1$ (default: 1).

When the $P_{\max}$ parameter is restricted, our algorithm functions as an online algorithm, retaining only the parts of the dataset in memory that it requires to calculate periodicities. There is a natural bound on the maximum period of mined patterns if the number of timesteps T is finite and known (see Proposition 3.4.2). However, in many situations this information is not available or relevant, such as in streaming sensor data. In such cases, an unrestricted maximum period value places a large computational burden on the algorithm, and requires that the entire dataset be retained in memory. This is because at any timestep t, any previously observed timestep $t' < t$ could contain the initial occurrence of a periodic subgraph whose second occurrence is at timestep t. Testing for this situation requires all

previously seen timesteps to be retained in memory, either explicitly or in some compressed form. The optional $P_{\max}$ parameter limits the maximum period of mined patterns, and thus eliminates the need to retain previously seen timesteps beyond a certain history.

The default parameters mine a complete set of periodic subgraphs without any smoothing, although in practice, only σ values of 3 or more are meaningful. The output of the algorithm is a set of closed parsimonious periodic subgraphs embeddings that satisfy the minimum support. Each embedding is written to the output stream as soon as the last possible occurrence of the subgraph has been encountered, or when the input stream has been exhausted.

3.5.2 Data Structures

As the algorithm scans the input stream, it maintains three primary data structures to track PSEs: a *pattern tree*, a *subgraph hash map*, and an optional *timeline list* to increase efficiency. An auxiliary data structure, called a *descriptor*, is used as a compact representation of a periodic support set. We refer to nodes in the pattern tree as *treenodes* to distinguish them from nodes (vertices) in the dynamic network or in a periodic subgraph. Each treenode N is associated with a single periodic subgraph F and a set of descriptors that represent PSEs of F. We use the notation *'treenode N/F'* to refer to a treenode N that represents subgraph F.

3.5.2.1 Pattern Tree, Subgraph Hash Map and Timeline List

The tree structure represents a subgraph relationship between periodic subgraphs. The structure of the pattern tree is subject to a single constraint: with the exception of the

special root node, all descendants of a treenode N/F are associated with proper subgraphs of F, but not all subgraphs of F are necessarily its descendants in the tree. This property allows efficient traversal of the tree by the mining algorithm, and also allows the tree to be built and manipulated quickly and represented using very little space.[1] It also allows efficient traversal by virtue of the fact that if F is not observed at a given timestep for treenode N/F, then neither are the subgraphs represented by N's descendants (except for the root node). Direct access to treenodes is also required, which is achieved by using a hash map to associate periodic subgraphs with their corresponding treenode. This can be done efficiently, as described in Property 3.1.4 of the set representation of dynamic networks. The timeline list is an optional component that links treenodes to the future timesteps at which they are expected to appear. Its use is discussed in Section 3.5.6.

3.5.2.2 Treenodes

Each treenode N/F contains a list of descriptors $\{D_1, ..., D_n\}$, one for each observed PSE of F. In addition, each treenode maintains a list of periods and phases of all live descriptors (see below), which is used by the tree update algorithm. Querying, adding to, and removing descriptors from this list are the primary operations on a treenode.

---

[1] An alternative to the tree representation would be to construct a full subgraph lattice (Carpineto and Romano, 2004), with a corresponding increase in time and space complexity. Whether lattices are more efficient given the typical sparsity of dynamic networks is a question for future research.

3.5.2.3 Descriptors

A descriptor D is the abbreviated representation of a periodic support set. It is associated with a treenode N/F and defines a unique PSE for F. It is formally described as a triple, since it represents a periodic support set $S_P = (i, p, s)$. The last element in the support set is defined as $t_j = t_i + p \cdot (s - 1)$ and the next expected timestep as $t_n = t_j + p$. Since descriptors are created, updated, and deleted as the input stream is read, the following definition describes the different states in which a descriptor could be at any given time.

Definition 3.5.1. (DESCRIPTOR STATES) At timestep t, a descriptor D for a subgraph F is *live* if $t_n > t$ or if $t_n = t$ and F is present at G_t. A descriptor that is not live is not currently exhibiting periodic behavior; it cannot change state again once it is not live. A descriptor where $t_i = t_j$ is a special case called an *anchor descriptor*, as it does not represent a periodic support set but could potentially become one if the associated subgraph F is observed at a future timestep. An anchor descriptor is defined to have a period of 0. An anchor descriptor is always live, unless $P_{\max}$ is defined and $t - t_i > P_{\max}$, in which case the anchor can never lead to a valid PSE with period at most $P_{\max}$, and is no longer needed.

3.5.3 Tree Update Algorithm

We now describe the update algorithm for the pattern tree, which is the core of the mining process. It is called once for each timestep that is read from the input. Starting with an initial pattern tree with an empty root treenode, at timestep t the algorithm traverses the pattern tree in a breadth-first search (BFS) to update treenodes with the new information contained in G_t. For each G_t, we are only interested in treenodes which might

be affected by the new information. This excludes any subgraph F which has an empty maximal common subgraph with G_t. In most cases, this process eliminates some branches of the pattern tree from the BFS traversal. At each treenode N/F where F has some part in common with G_t, we update descriptors at N in a manner described below. We end each tree update by ensuring that a treenode for G_t in its entirety exists in the tree with an anchor descriptor for timestep t. This accounts for the possibility that G_t in its entirety is the first occurrence of a (future) periodic subgraph. If such a treenode does not exist, it is created at a location which does not violate the subgraph property of the tree, such as the root.

During the breadth-first traversal of the tree, one of the following three conditions holds at each treenode N/F. Let $C = F \cap G_t$ be the maximal common subgraph of G_t and F.

1. *Update descriptors:* If $F \subseteq G_t$, *i.e.* if $F = C$, then F has appeared in its entirety at timestep t. Let D be any descriptor in N and $t_n = t_j + p$ be the next expected timestep for D.

   (a) If $t_n = t$, then D has appeared where it was expected. Timestep t is added to D's support to ensure temporal maximality.

   (b) If $t_n < t$, then D has not appeared when expected and is thus no longer live. It is written to the output stream if its support is greater than or equal to σ, and removed from the tree.

   (c) If $t_n > t$, then nothing is done.

(d) If $p = 0$, then D is an anchor descriptor. Given that timestep t is the second occurrence of F, a new descriptor D' is spawned with period $p' = t - t_i$ and phase offset $m' = (t_i - 1)$ **mod** p'. If N does not contain a live descriptor with the same period and phase offset, D' is added to the list of descriptors at N.

2. *Propagate descriptors:* If $C \neq \emptyset$ and the condition above does not hold, then a subgraph C of F is present at timestep t, instead of F in its entirety. This happens, for example, when a formerly periodic subgraph F fractures into a smaller subgraph C that continues F's periodic behavior. If a treenode for C does not already exist in the tree, determined using the subgraph hash map, it is created as a child of N (to satisfy the subgraph relationship). Let D be any descriptor at N. If $t_n = t$, then D represents a PSE which subgraph C must inherit and continue. The treenode for C receives a copy of D, if a live descriptor of the same period and phase offset does not already exist. The pattern $< F, D >$ is written to the output stream if the support of D is greater than or equal to σ, and then D is removed from treenode N.

3. *Dead subtree:* If $C = \emptyset$, then G_t and F have no common subgraph, and no descriptors at N are directly affected by the observation of G_t. Furthermore, no treenode that is a descendant of N will have any common subgraph with G_t either, since they are all subgraphs of F. The subtree rooted at N is therefore eliminated from the rest of the tree traversal.

Algorithm 1 UpdateTree(G_t)

Require: G_t is the graph of timestep t

```
1: Q ← new queue
2: push(Q, root.children)
3: while N ← pop_front(Q) do
4:    C ← G_t ∩ N
5:    if C is not empty then
6:       if N ⊆ G_t then
7:          UpdateDescriptors(N)
8:       else
9:          W ← FindNode(N) or NewNode(N, C)
10:         PropagateDescriptors(N, W)
11:      end if
12:      push(Q, children(N))
13:   end if
14: end while
15: W ← FindNode(G_t) or NewNode(root, G_t)
16: Add anchor descriptor for G_t to W.
```

Figure 19 shows the pattern tree at each timestep during the execution of the algorithm on the network from Figure 16. For clarity, we have described a very basic version of the algorithm. Two notable aspects of this algorithm are (1) that it outputs all PSEs, which are a superset of all PPSEs, and (2) it can dynamically calculate the purity measure. Non-parsimonious PSEs can be post-processed out of the output, but in Section 3.5.6, we show how this can be accomplished dynamically.

3.5.4 Correctness

The pattern tree is intended to hold all PSEs seen up to timestep t. We prove by induction that this consistent state holds at any point during the execution of the algorithm. We define a consistent state for the pattern tree as the following four conditions.

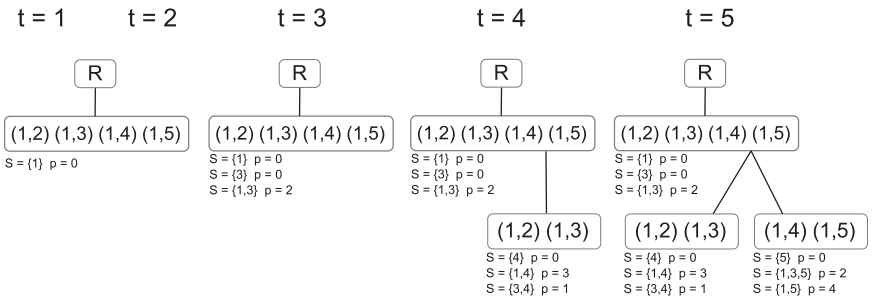


Figure 19: The pattern tree at each timestep for the dynamic network shown in Figure 16, considering only edges for brevity.

Definition 3.5.2. (Pattern Tree Consistency Conditions) The pattern tree is in a *consistent state* if the following four conditions are met:

1. The subgraph property of the pattern tree holds, *i.e.* all descendants of a treenode N/subgraph F contain subgraphs that are proper subgraphs of F.

2. All descriptors in the pattern tree are unique, *i.e.* no two descriptors D_1 and D_2 anywhere in the tree share the same subgraph and the same support set.

3. All PSEs with support $S_P \geq 2$ encountered in the data stream so far have a descriptor (and thus a treenode) in the tree.

4. All non-anchor descriptors represent PSEs that are closed up to timestep t, *i.e.* for a descriptor D in a treenode N/subgraph F, F is the maximal common subgraph of the support set described by D, and the support set is temporally maximal at timestep t as per Definition 3.2.1.

If the tree is in a consistent state at timestep t, then the remaining output up to timestep t can be obtained by traversing the tree once and writing every subgraph/descriptor pair where the support of the PSE is $|S_P| \geq \sigma$. The tree is initially empty except for a dummy root node. It is therefore consistent because the four consistency conditions are vacuously true. For the inductive hypothesis, assume that the pattern tree is consistent after processing timestep G_{t-1}. Then after processing G_t, we show below that the tree is still in a consistent state, thus proving that the tree is in a consistent state during and at the end of the execution of the mining algorithm. The following is the statement and proof of the inductive step.

Theorem 3.5.1. *If the pattern tree is in a consistent state after processing G_{t-1}, then the pattern tree is also in a consistent state after using Algorithm 1 to process G_t.*

Proof. On reading G_t from the input stream, the first two consistency conditions are not violated because no new subgraphs or descriptors have been added to the tree. Conditions 3 and 4, on the other hand, might be violated because G_t could potentially contain a previously unseen PSE, violating condition 3, or require that an existing one have its support set extended to include t, violating condition 4. Therefore, we start by focusing on events that would violate the latter two consistency conditions, while showing that the first two remain satisfied during processing. We describe each event in turn and how the consistency of the tree is violated, as well as the correctness of the actions taken to restore consistency. The following is an exhaustive list of such events, along with the action that the algorithm takes:

Case 1: G_t contains the first occurrence of a new PSE, violating condition 3; an anchor descriptor starting at timestep t is added to a treenode for G_t in its entirety.

Case 2: G_t contains the n^{th} occurrence of a new PSE, where $n > 1$ and prior occurrences were contained within some other PSE, violating condition 3; the PROPAGATEDESCRIPTORS function is called. When $n = 1$, we have case 1 above.

Case 3: G_t contains the n^{th} occurrence of an already existing PSE, where $n > 1$, violating condition 4; the UPDATEDESCRIPTORS function is called. Timestep t cannot be the first occurrence for an *existing* PSE, by definition.

Case 1:

The first possibility is that G_t could contain the *first occurrence of a new PSE*. Since we have no way of knowing the future, we always assume that the entire graph G_t is going to become a periodic subgraph in the future with timestep t as its first timestep.[1] In Algorithm 1, a treenode W is added for G_t at the root if one does not already exist in the tree, and an anchor descriptor starting at t is added to W. Adding W at the root is a simple way to ensure that condition 1 is never violated. The descriptor is guaranteed to be unique, because no other PSE of G_t will have started at timestep t prior to G_t having been observed, and therefore condition 2 is not violated. If we are correct about the assumption that timestep t is the first occurrence of a new PSE for G_t, then we have 'presciently' added a descriptor and treenode for it at the correct time, and ensured that condition 3 is not

[1]Incidentally, there is at least one dynamic network where each timestep contains the first occurrence of a new PSE – the worst-case construction from Section 3.4.

violated. On the other hand, if G_t never occurs again, then its treenode will only contain an anchor descriptor, which is exempt from condition 4. Therefore, case 1 no longer causes the tree to be inconsistent.

Case 2:

Suppose that G_t is the n^{th} *occurrence of a new PSE, for* $n > 1$. This happens when a subgraph stops exhibiting periodic behavior, but a smaller portion of it continues to do so. The treenode for the smaller subgraph might therefore need to 'inherit' some descriptors from the treenode of the larger subgraph. For each treenode N/F, case 2 arises when $F \cap G_t \neq \emptyset$ except when $F \subseteq G_t$ (this exception is handled in the next case). Let $C = F \cap G_t$, the maximal common subgraph of F and G_t. Let W be the treenode for C, which is created in the tree (at a position that does not violate condition 1) if it does not already exist.

We now need to copy descriptors where $t_j + p = t$ from N to W, since these descriptors would have been updated if F had been observed in its entirety. Let D be one such descriptor. D is now no longer live for N/F because it has failed to appear in its entirety at timestep t. The propagation process transfers D to W if W does not already have a live descriptor of the same period and phase offset D and an earlier starting position. Since treenodes N and W represent different subgraphs, copying D from N to W does not violate condition 2. Furthermore, since D was temporally maximal before, it is again temporally maximal with the addition of t to its support set. This handles conditions 4 and 3, and

case 2 no longer causes the tree to be inconsistent.

Case 3:

Finally, we handle the case that G_t is the n^{th} *occurrence of an existing PSE, for* $n > 1$. This happens when a treenode N/F has $F \subseteq G_t$, which means that F has appeared in its entirety and its descriptors need to be updated. The update process scans each descriptor D in treenode N. If D is next expected at timestep t, then t is added to its support set by setting $t_j = t$. This satisfied consistency condition 4. If D is no longer exhibiting periodic behavior, *i.e.* if $t_j + p < t$, then D is flushed to the output stream if appropriate and then deleted. The other conditions are not violated. The final case is therefore handled correctly, and the pattern tree is again in a consistent state. □

We have inductively shown that Algorithm 1 results in a consistent tree after processing each timestep G_t in increasing order of t. This proves the correctness of the algorithm.

3.5.5 Time and Space Complexity

Given that the tree consistency conditions hold, the number of descriptors (and therefore nodes) in the tree at timestep T is bounded by Theorem 3.4.2 at $\sigma = 2$. As each timestep is read, the tree is traversed once. When descriptors are created or propagated, we ensure that at most one live descriptor exists at each treenode for a given period and phase offset. If the list of periods and phase offsets of live descriptors in the treenode are represented as sparse two-dimensional arrays, then lookup can be performed efficiently in constant time with $O(P_{\max}^2)$ or $O(T^2)$ space complexity to hold the arrays. Thus, the worst-case time com-

plexity of the algorithm involves traversing each descriptor in the tree once for each timestep and calculating the maximal common subgraph at each treenode. From Property 3.1.1, the maximal common subgraph of two graphs can be calculated in time $O(V + E)$. This yields a total time complexity of $O((V + E)T^3 \ln T)$ when $P_{\max}$ is not specified. When $P_{\max}$ is specified, the range of allowable periods is bounded in Theorem 3.4.2 and the maximum number of patterns can drop very significantly. The worst-case space complexity of our algorithm is $O((V + E + P_{\max}^2)T^2 \ln T)$ when $P_{\max}$ is specified. In practice, however, the tree size is usually several orders of magnitude smaller than the worst-case bound, as we will demonstrate.

3.5.6 Extensions to the Basic Algorithm

We have described a basic version of the mining algorithm in Section 3.5.3. A number of algorithmic refinements are possible to increase efficiency, but at the cost of conceptual simplicity. We briefly describe some of these refinements below.

3.5.6.1 Mining Parsimonious PSEs

The most important enhancement is to make the algorithm dynamically output only parsimonious PSEs. Recall the subsumption conditions from Definition 3.2.3. A simple way to modify Algorithm 1 to only output parsimonious PSEs is by adding an indicator bit to each descriptor to indicate subsumption. This bit is initially cleared when the descriptor is created. When any descriptor D from treenode N/F is flushed, its subsumed bit is first checked. If it is cleared, then D is compared to all other live descriptors at N. If D is subsumed by another descriptor, it is not written to the output. On the other hand, if D

subsumes (as of timestep t) some other descriptor D', the subsumed bit for D' is set. If the support of D' increases in the future, its subsumed bit is cleared since Condition 3 from Definition 3.2.3 is no longer true. However, if its support does not increase, then all the conditions from Definition 3.2.3 hold and D' is not parsimonious. It will not be flushed when the cessation of its periodic behavior is finally confirmed.

3.5.6.2 Sorted Descriptor List

The list of descriptors at each node can be stored sorted by the next expected timestep of each descriptor. At timestep t, only descriptors which are expected at or before t will be examined, in addition to at most one descriptor that is expected after timestep t. This cuts down on the number of descriptors that need to be examined during each tree update, at the computational cost of having to sort the list of descriptors after each update. Since the number of descriptors per treenode is generally not very large, the computational overhead is minimal in practice.

3.5.6.3 Lazy Tree Updates

In practice, the algorithm spends most of its running time calculating intersections of integer sets (line 7 in Algorithm 1). Although the maximum common subgraph of two graphs is calculated in time linear in the number of vertices and edges, the size of the graphs results in a relatively expensive intersection computation. The sparsity of the network generally results in a relatively small number of treenodes, which means that many such intersections between large sets must be performed. Thus, to improve the practical efficiency of the algorithm, we can delay calculating intersections until it is absolutely necessary. This results

in the lazy-intersection tree update algorithm shown in Algorithm 2. The tradeoff is that the total support of patterns, and therefore the purity measure, cannot be dynamically calculated.

3.5.6.4 Using a Timeline to Trim the Tree

The timeline associates each future timestep with a list of treenodes that have at least one descriptor expected at that timestep. It can be dynamically updated at an insignificant cost (constant or logarithmic) once per treenode update, and stored in space linear in the number of treenodes. After the tree update for timestep t, all treenodes that are still associated with timestep t are guaranteed not to have been visited during the tree update, and have at least one descriptor which is no longer periodic. These treenodes can then be visited and the invalid descriptors removed, in time proportional to the number of descriptors to be removed. Thus, at the end of each tree update operation, the treenode only contains descriptors that are live at the next timestep. This ensures that the pattern tree contains a minimal number of descriptors and treenodes at any given timestep.

3.6 Experimental Evaluation

We use four real-world dynamic social networks to evaluate our algorithm as well as some characteristics and applications of periodic subgraph mining. We also use artificial data to compare the performance of our algorithm with that of SMCA (Huang and Chang, 2005), a periodic pattern mining algorithm that generates periodic patterns in a level-wise search similar to *Apriori* and without closed or parsimonious considerations. SMCA is a four-phase algorithm and we only use the first two phases (SPMiner and MPMiner), since

Algorithm 2 LazyUpdateTree(G_t)

Require: G_t is the graph of timestep t

```
1: Q ← new queue
2: push(Q, root.children)
3: while N ← pop_front(Q) do
4:    lazy ← true
5:    while lazy = true do
6:       D ← next descriptor at N
7:       next ← last(D) + period(D)
8:       if D is an anchor or next = T then
9:          lazy ← false
10:      else
11:         if next < T then
12:            flush D to output and delete
13:         else
14:            break
15:         end if
16:      end if
17:   end while
18:   if lazy = false then
19:      C ← G_t ∩ N
20:      if C is not empty then
21:         if N ⊆ G_t then
22:            UpdateDescriptors(N)
23:         else
24:            W ← FindNode(N) or NewNode(N, C)
25:            PropagateDescriptors(N, W)
26:         end if
27:         push(Q, children(N))
28:      end if
29:   else
30:      push(Q, children(N))
31:   end if
32: end while
33: W ← FindNode(G_t) or NewNode(root, G_t)
34: Add anchor descriptor for G_t to W.
```

their combined functionality is comparable to our algorithm.[1] We first report results on the comparison with SMCA on synthetic data, before moving on to evaluating our algorithm on real dynamic networks.

We implemented our algorithm in C++, incorporating all the optimizations described in Section 3.5.6. The subgraph hash map was implemented using the Google `dense_hash_map` library[2], optimized for speed over memory usage. The experiments with synthetic data were run on a dual-core Intel Pentium D system running at 3.2 GHz with 3 GB of RAM and Linux kernel 2.6.28. The experiments with real data were run on a quad-core Intel Xeon server running at 2.6 GHz with 24 GB of RAM and Linux kernel 2.6.22. In all cases, computation time is reported as the sum of the user (computation) and kernel (I/O, etc.) CPU time reported by the Linux `getrusage()` system call. Memory usage is the maximum resident set size reported by the Linux `proc` filesystem. The SPMiner and MPMiner components of the SMCA algorithm were implemented in C++ according to the pseudocode in (Huang and Chang, 2005), and use the same input, timing and output mechanisms as our algorithm.[3]

---

[1]The functionality is comparable in terms of the stated goal of the algorithm only, which is to mine periodic 'multiple event 1-patterns'. SMCA suffers from the fact that it does not generate closed or parsimonious output, thus increasing its computation time and output size relative to our algorithm, without adding any extra information.

[2]`http://code.google.com/p/google-sparsehash/`, version 1.4.

[3]A misprint in the pseudocode for SPMiner in (Huang and Chang, 2005, (Fig 3, line 12)) was corrected. For MPMiner, we used the Time-Based Enumeration (TBE) scheme, since the Segment-Based Enumeration (SBE) scheme exhausted all available system memory for the datasets we tried.

Dataset	Vertices	Timesteps	Avg. density	S	$P_{\max}$
Enron	82,614	2,588	0.028 ± 0.064	3	40
IMDB Photos (full)	29,257	13,987	0.097 ± 0.21	3	400
Plains Zebra	313	1,276	0.31 ± 0.27	6	400
Reality Mining	100	2,940	0.23 ± 0.17	2	60
Server Log 1 (days)	111,108	783	0.024 ± 0.019	2	40
Server Log 2 (hours)	111,108	18,807	0.24 ± 0.3	2	960

TABLE V: DATASET CHARACTERISTICS, SMOOTHING (S), AND MAXIMUM PERIOD ($P_{\max}$) VALUES USED FOR EXPERIMENTAL EVALUATION.

3.6.1 Datasets

We used dynamic networks collected from a variety of sources and covering a range of interaction dynamics. These networks are described below.

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.[1] Timestamps, senders and lists of recipients were extracted from message headers for each e-mail on file. We chose a day as the quantization timestep, with a directed (unweighted) interaction present if at least one e-mail was sent between two individuals on a particular day.

Plains Zebra. Ecologists are interested in studying the association patterns of wild Plains zebras (*Equus burchelli*) in their natural habitat. For this dataset, social interactions between animals were recorded in a nature reserve in Kenya by behavioral ecologists from Princeton University, based on direct visual observations (Fischhoff et al.,

[1] Available at `http://www.cs.cmu.edu/~enron/`

2007; Sundaresan et al., 2007; Juang et al., 2002). Zebras are uniquely identifiable by the pattern of stripes on various parts of their bodies. The data was collected by ecologists making visual scans of the herds, typically once a day over periods of several months. Each entity in the dynamic network is a unique Plains zebra and an interaction represents social association, as determined by spatial proximity and the domain knowledge of ecologists.

Reality Mining. Cellphones with proximity tracking technology were distributed to 100 students at the Massachusetts Institute of Technology over the course of an academic year (Eagle and Pentland, 2006). The timestep quantization was chosen as 4 hours (Clauset and Eagle, 2007).

IMDB Celebrities. The Internet Movie Database (IMDB)[1] maintains a large archive of tagged, disambiguated and dated photographs of individuals associated with the production of commercial entertainment, including actors, directors and musicians. One might reasonably assert that a degree of social (or at least professional) association exists between people photographed together by the popular press. Thus, similar to the methodology of the Plains zebra sightings, we collected metadata on 193,707 photos[2], which collectively represent a partial structure of the social network of people

---

[1] `http://www.imdb.com`

[2] In (Lahiri and Berger-Wolf, 2008), we only used photos with two or more people, which is the reason for the dataset size discrepancy between versions. For this dataset, it is also informative to represent singleton (disconnected) vertices, which we have done here.

associated with the entertainment industry. The quantization period was one day. Although the time span of the dataset is just under forty years, most of the interactions occur in the later portion of the dataset.

Server Logs. We used the HTTP access logs from an Apache web server hosting organization and personal pages for the Laboratory of Computational Population Biology at the University of Illinois at Chicago.[1] Each vertex is either an IP address on the Internet or a file hosted on the web server. A directed edge from an IP address to a file indicates that the file was successfully accessed by a host at the IP address, creating a bipartite graph at each timestep. The log data runs from April 2007 to May 2009. We used two different quantizations of one day and one hour per timestep.

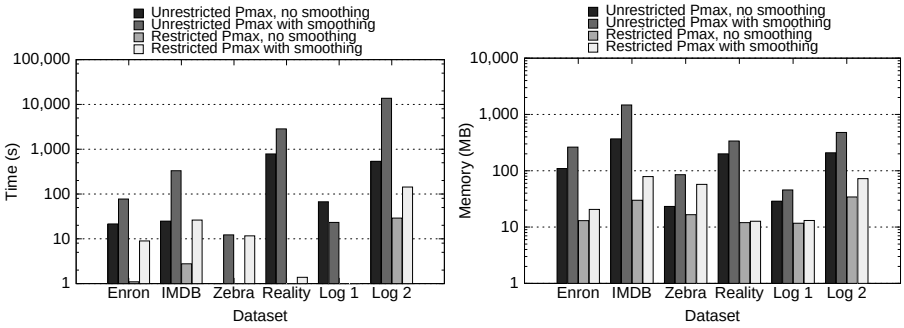
3.6.2 Results on Natural Data

3.6.2.1 Algorithm Performance

We first ran a series of experiments on our algorithm with $\sigma = 3$ and no smoothing, *i.e.* mining only perfectly periodic patterns. We then ran a second set of experiments with $P_{\max}$ set to restricted values, and a third set of experiments with $\sigma = 3$ and variable amounts of smoothing per dataset. Table Table V summarizes the $P_{\max}$ and smoothing values used for each dataset, based intuitively on typical periodicities and how much noise we would expect in each dataset. The second and third set of experiments demonstrate the performance of the algorithm in online and noisy situations, respectively.

---

[1] `http://compbio.cs.uic.edu/`

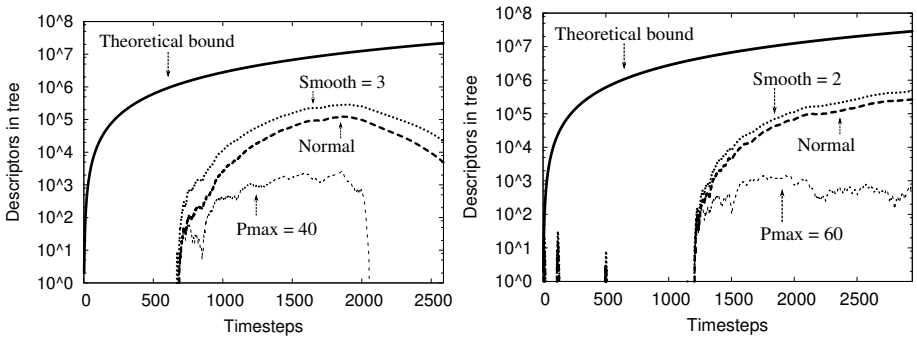


(a) Mining time (b) Memory usage

Figure 20: Performance of the periodic subgraph mining algorithm at $\sigma = 3$, shown with an exponential y-axis.

Figure 20 shows the running time and memory usage of our algorithm under different circumstances. The black column shows the case when no smoothing is used and the maximum period is unrestricted. This might be considered a typical 'offline' analysis scenario. An interesting point to note is that Reality Mining takes much more time to complete mining than the much larger Enron dataset, most likely due to the density of periodic patterns in it. In the typical online analysis scenario with a restricted $P_{\max}$, the algorithm took less than 30 seconds to execute and used less than 40 MB of memory in all cases. As expected, restricting the maximum period has a very significant effect on the performance of the algorithm.

Figure 21 shows the size of the pattern tree at each timestep for the Enron and Reality Mining datasets. It can be seen that the actual tree size is a small fraction of the theoretical upper bound. Furthermore, limiting the maximum period of mined patterns has a large



(a) Enron (b) Reality Mining

Figure 21: Number of pattern tree descriptors with no smoothing or restrictions on period ('normal'), and for various smoothing and $P_{\max}$ values, compared to the theoretical bound.

impact on reducing the tree size, as expected. The Enron plot dips dramatically after about timestep 2,000 because most timesteps after that are empty. A large number of descriptors are flushed from the pattern tree when the empty timesteps are encountered. No such dip occurs in the Reality Mining dataset, which is densely periodic and continues to exhibit periodic behavior right up to the very end of the observation period.

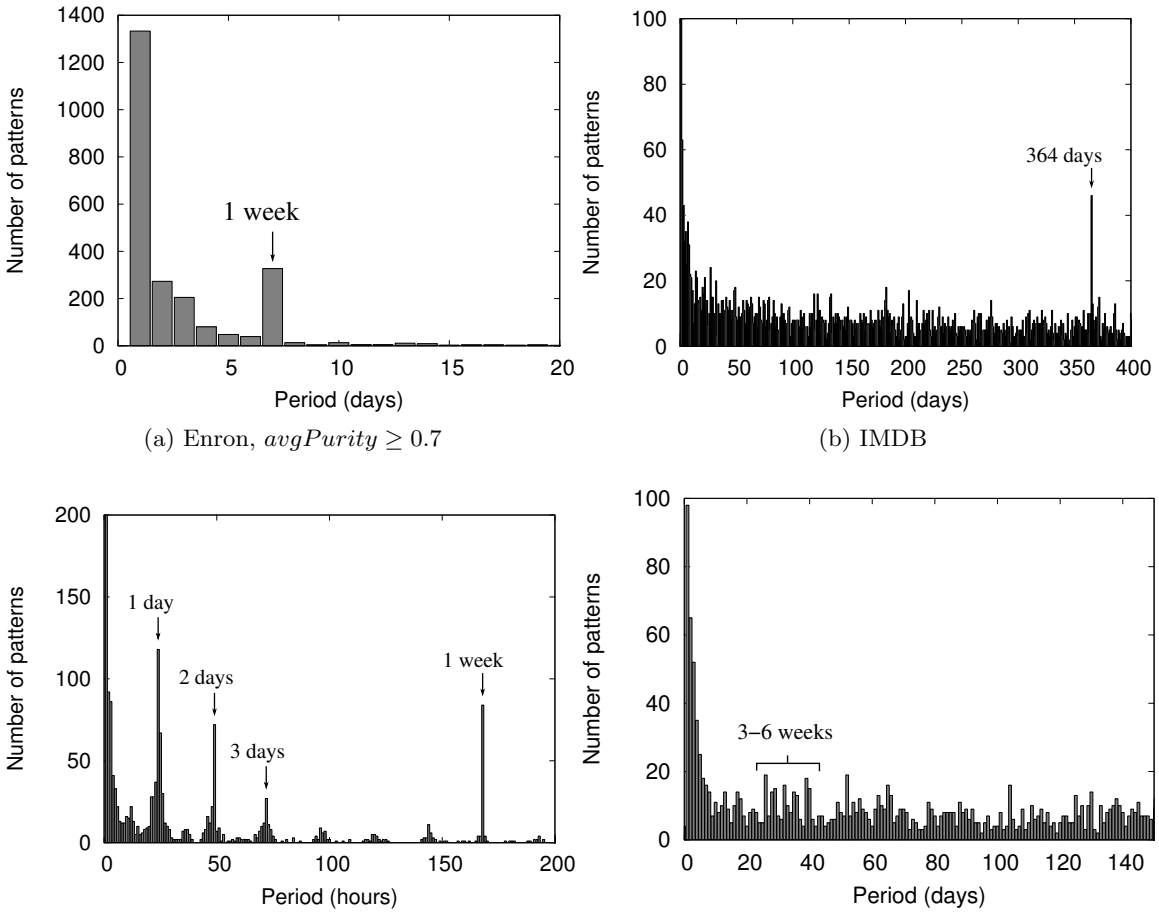
3.6.2.2 Characterizing Inherent Periodicity

In addition to investigating specific periodic interaction patterns, a second goal for mining parsimonious PSEs is to analyze global periodicities in the system. In the context of dynamic networks, the goal would be to characterize the gross dynamics of the individuals in the system. Figure 22 shows histograms of the periods of patterns mined from the Enron, IMDB, Server Log and Plains Zebra datasets. For Enron, we restrict our attention to patterns with a high average purity, i.e. patterns which are likely to capture truly periodic

behavior. Daily interaction patterns are the most prevalent periodic patterns[1], followed by weekly patterns, as manifested by the clear peak at $p = 7$. For the IMDB dataset, we notice a similar peak at about $p = 364$. This can be explained by celebrity sightings at annual events – awards shows, for example. Thus, we are able to capture and characterize plausible natural periodicities in human interactions with no prior knowledge about the datasets. The hour-quantized Server Log dataset shows a number of interesting peaks at about 24, 48 and 168 hours (the last one corresponding to a periodicity of one week). Note that there is also relatively little variance around the peak at one week, suggesting that these accesses were performed automatically. Inspecting patterns at these periods revealed the activity of various search engine crawlers, confirmed by checking ownership of IP netblocks and User-Agent strings in the HTTP requests. The Plains Zebra dataset showed a wide range of periodicities, as one might expect of animal behavior, with no strongly discernible peaks.

Figures 22a and 22c are histograms of the periods of patterns that are above a minimum purity threshold. Clearly, changing this threshold could result in a different picture, as patterns of lower purity get included. Figure 23 shows a two-dimensional view of the histograms as a density plot. Each row represents a histogram as in Figure 22, but thresholded by the value of the y-axis. Darker cells represent a higher concentration of patterns at that period (relative to the most concentrated cell in the row), and correspond to the peaks in

[1]Too much importance should not be attached to patterns of period 1 in plots thresholded by purity, since all patterns of period 1 necessarily have purity 1.



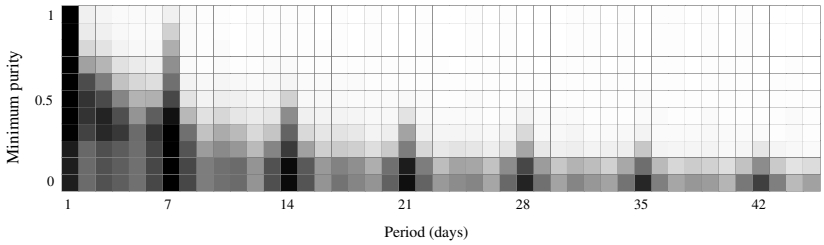
(a) Enron, $avgPurity \geq 0.7$

(b) IMDB

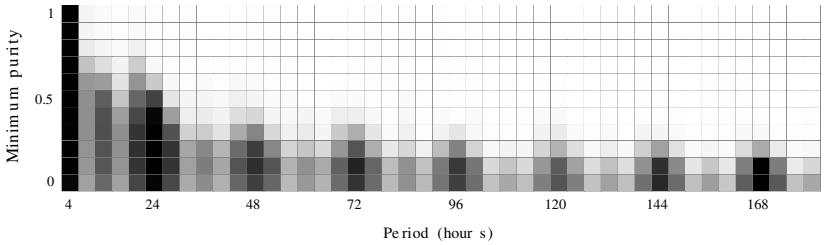
(c) Server Log 2, $avgPurity \geq 0.5$

(d) Plains Zebra

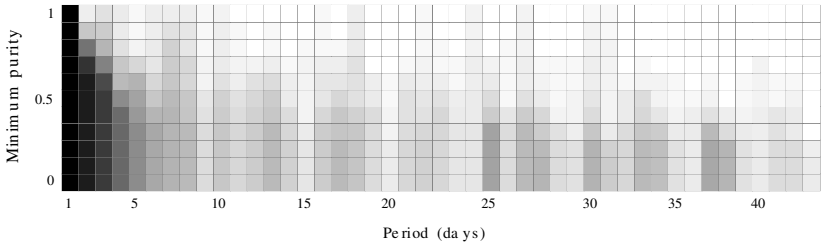
Figure 22: Number of patterns at each period.



(a) Enron



(b) Reality Mining



(c) Plains Zebra

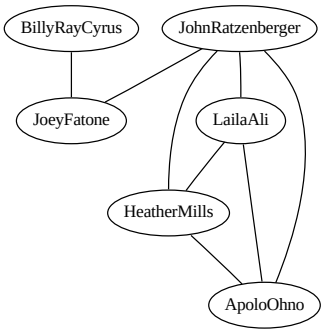
Figure 23: Pattern density at each *minimum purity* threshold. Each row shows the distribution of pattern periods for patterns with purity at or greater than the y-axis value. Darker cells indicate more patterns.

Figure 22. The top-most row is the distribution of the periods of patterns that only occur periodically, *i.e.,* never in-between periodic occurrences, whereas the lowest row places no constraints and shows the period distribution of all mined patterns. In Figure 23a, for example, the row corresponding to a y-value of 0.7 represents the histogram in Figure 22a.

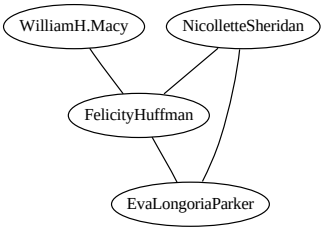
The Enron and Reality Mining datasets show strong daily and weekly periodicities, as might be expected from human interactions. This commonality is interesting because the interactions occur by different mechanisms in each dataset – by e-mail in the Enron dataset, and by physical proximity in the Reality Mining dataset. The Plains Zebra dataset, while not showing periodicities as strong as the human datasets, seem to contain relatively dense region at periods between 25 and 38. It is currently unclear whether this region indicates behavior that is ecologically meaningful, or is an artifact of the data.

3.6.2.3 Qualitative Analysis

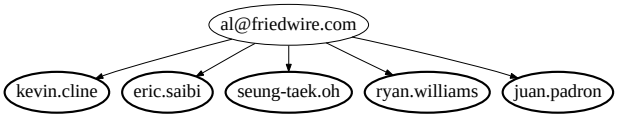
We now turn our attention to some qualitatively interesting periodic subgraphs discovered by our algorithm illustrating a range of periodic behavior. Figure 24a illustrates a somewhat complex pattern from the IMDB photo database that repeated approximately every week. Although the support is relatively low, what is interesting about this subgraph is the repeated non-trivial grouping of people, all of whom turned out to be contestants on a weekly 'reality television' show. Figure 24b is also from the IMDB database and is an approximately annually repeating pattern. The three individuals in the clique are actresses in a popular (circa 2004) television show, while the fourth vertex is the spouse (as of 2009) of one of the actresses. Given this context, the low average purity of the pattern is to be expected as the three actresses are very likely to have appeared together in between what are likely to be award shows. The nontrivial links in such patterns are particularly interesting and are indicative of the show's progression or relationships other than co-starring.



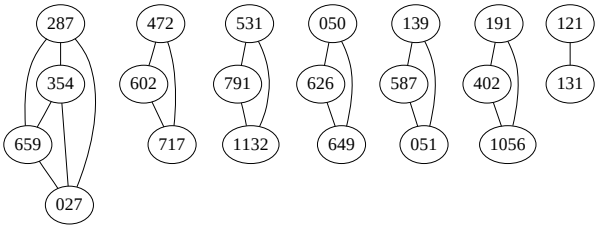
(a) IMDB: period 7 ± 2, support 3, avg. purity 1



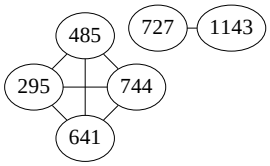
(b) IMDB: period 364, support 3, avg. purity 0.4



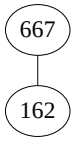
(c) Enron: period 1, support 84, avg. purity 1. Bold circles represent @*enron.com* e-mail addresses.



(d) Plains: period 7, support 4, avg. purity 0.94.



(e) Plains: period 61±6, support 3, avg. purity 0.71



(f) Plains: period 81±6, support 4, avg. purity 1

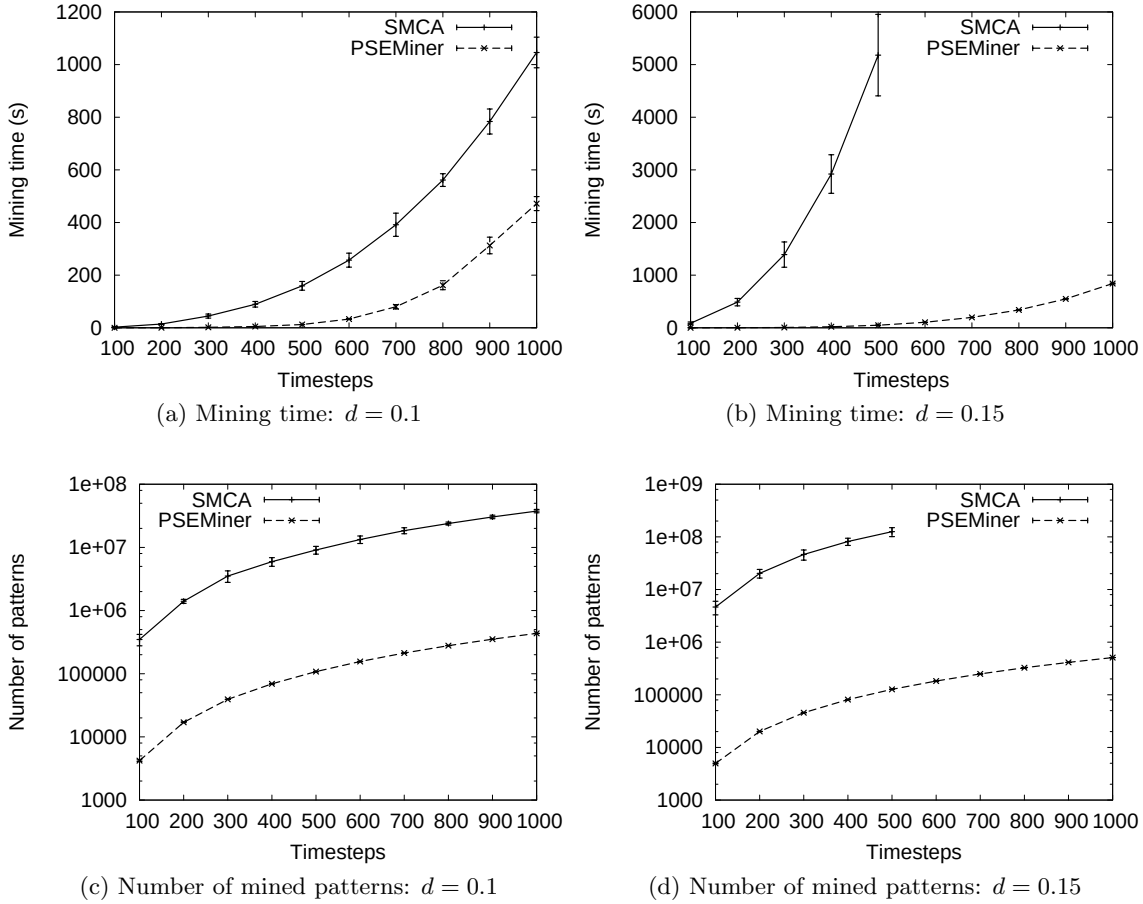
Figure 24: Examples of some interesting periodic subgraphs.

The subgraph shown in Figure 24c has the highest periodic support in the Enron dataset, repeating every day for 84 consecutive days, including weekends. This is representative of a large number of similar periodic patterns in Enron, in which one person emails a group of people with periods ranging from 1 to 14 days. As shown earlier in Figure 22, weekly emails seem to be particularly popular in a corporate setting such as this, and could be used to infer functional communities within the corporation.

Finally, we turn to the Plains Zebra dataset and to one of the most intriguing patterns shown in Figure 24d. Although it is quite likely that the period of 7 days is an artifact of the manner in which the population was sampled, the high purity of the pattern indicates that this is a relatively stable grouping. It is also by far the largest and most repetitive such pattern, parts of which are periodic at other times as well. In contrast, the subgraphs that repeat over multiple months are shown in Figures 24e and 24f. Although the support of the latter two patterns is relatively low, the high purity of Figure 24f stands out and is representative of a large number of small but highly periodic patterns. Moreover, all the patterns are of interactions of stallion male zebras and correspond to their harems grouping for a period of time. Such groupings are indeed considered more stable for short periods of time than bachelor associations (Fischhoff et al., 2007).

3.6.3 Comparison to SMCA

We generated relatively small synthetic datasets with different characteristics to compare the performance of our algorithm with the SMCA algorithm on simple interaction data. Starting with a population of 30 individuals, we generated a single graph of density d. The



(a) Mining time: $d = 0.1$

(b) Mining time: $d = 0.15$

(c) Number of mined patterns: $d = 0.1$

(d) Number of mined patterns: $d = 0.15$

Figure 25: The performance of SMCA compared to our algorithm, for synthetic networks of different densities d. Error bars for PSEMiner are too small to see.

edges of this graph were then sampled independently at random for each of the T timesteps. Although this is not intended to be a realistic model of a social network, it allows us to control two parameters crucial to the mining process – the overall density of the dynamic network, and the number of timesteps. Since real social networks are generally sparse, we used two values for d: 0.1 and 0.15. For each of these values, T was varied from 100 to 1000 in increments of 100.

Ten random dynamic networks were generated for each combination of T and d, and converted to their set representations. Both algorithms were run on the same set of synthetic networks with a minimum support value of $\sigma = 3$ and the maximum period unrestricted, calculated using Proposition 3.4.2 for each value of T. All algorithms were limited to 8 GB of disk space for storing their output, which can be considered reasonable given that this value is several orders of magnitude larger than the size of the input networks.

Figure 25 shows the performance of SMCA compared to our algorithm. The computation time used by both algorithms is comparable for density $d = 0.1$, although SMCA does not scale as well as our algorithm. For a slightly higher density of $d = 0.15$, the number of periodic patterns is expected to increase as well. The computation times are no longer comparable between algorithms, as shown in Figure 25b. In Figures 25b and 25d, there are no data points for SMCA beyond $T = 500$ since the algorithm reached the maximum output size of 8 GB prior to completion. This is partly caused by the fact that SMCA does not output closed or parsimonious patterns, which is evident from the number of patterns generated by SMCA, shown in Figures 25c and 25d on a logarithmic scale.

Thus, our algorithm scales much better than SMCA. The number of patterns generated by SMCA is generally about three orders of magnitude larger than the number of parsimonious patterns output by our algorithm. The intractability of non-parsimonious periodic pattern mining is one of the main reasons we could not use SMCA on the larger natural datasets, where the number of vertices, timesteps, and the average timestep density are much higher than the values used here.

3.7 Summary

We have proposed and formalized the *periodic subgraph mining* problem for dynamic networks and analyzed the computational complexity of enumerating all periodic subgraphs. We have shown that there are at most $O(T^2 \ln \frac{T}{\sigma})$ closed periodic subgraphs at minimum support σ in a dynamic network of T timesteps. Furthermore, we have described a polynomial time, online algorithm to mine all periodic subgraphs, including a smoothing mechanism for mining subgraphs that are not perfectly periodic. We have also proposed a new measure, *purity*, for ranking mined subgraphs according to how perfectly periodic a subgraph is. We have demonstrated our algorithm on four real-world dynamic social networks, spanning interactions between corporate executives, college students, wild zebra, and Hollywood celebrities. Our algorithm efficiently mines all periodic patterns, is provably tractable, and is a meaningful alternative to using frequent subgraph mining to look for interesting patterns in dynamic networks. We have also shown that periodic subgraphs can be used as an effective characterization of the dynamics of various systems. Our technique was able

to discover plausible natural periodicities in many of the systems we examined, and shows promise as a tool for exploratory analysis of interaction dynamics.

There are a number of interesting avenues for future research. One such direction is to incorporate probabilistic models of periodicity instead of strictly combinatorial ones. Yang et al. (Yang et al., 2002) and Ma and Hellerstein (Ma and Hellerstein, 2001) are two examples of such attempts; it would be interesting to see how well they perform in dynamic networks. Along the lines of various studies on assessing the interestingness of frequent patterns (Bringmann and Zimmermann, 2009; Tatti, 2008; Yan et al., 2008), a method for assessing the statistical significance of mined patterns under different statistical models would be valuable in dynamic networks, especially in the context of inter-disciplinary research. A number of extensions can also be made to the algorithm we have presented in this chapter. These include an extension to mine complex periodic patterns, similar to the types of patterns mined in (Han et al., 1999; Huang and Chang, 2005; Yang et al., 2003; Ma and Hellerstein, 2001), and different algorithms or heuristics for manipulating the structure of the pattern tree to increase efficiency. The concept of noise could also be extended to discover noisy subgraphs instead of just noisy periodicities. Finally, we believe that the capabilities of the method, especially in an inter-disciplinary context, can only be fully explored if the results of the mining process are presented or visualized in a succinct but insightful manner. This is a challenging and open question.

CHAPTER 4

MINING COUPLED EDGES

In this chapter, we deal with the problem of mining strong temporal correlations between interactions in a dynamic network, which we call *coupled edges*. Typical dynamic network datasets can contain thousands or millions of edges occurring and re-occurring on a continous streaming basis. It is likely that a majority of these edges are unpredictable with any tractable model, and yet there are likely to be interactions that are quite regular and predictable. In a phone call network, for example, a user's outgoing and incoming phone calls may seem entirely random in aggregate, but there might be a regular phone call that is always made on the first Friday of each month, or between business hours each Tuesday. Similarly, if an e-mail sent from one person to another is frequently followed by a reply within a few hours, or a forward to a third person within five minutes, that is evidence for a specific type of relationship between them. The crux of this chapter is to tease out a few of these *predictive*, tightly coupled relationships solely from the dynamics of the network, *i.e.,* to extract structural correlations solely from network dynamics.

In contrast to other data mining methods, our definition of a strong temporal correlation requires a degree of predictive power on unseen data, rather than being based on descriptive statistics. In the examples above, the temporal correlations have to be in both directions – if an e-mail from A to B is correlated with a reply within a few hours, it should also be that a reply from B to A is preceded by an initial e-mail from A. In other words, an e-mail from

A to *B predicts* a reply within a few hours. By using predictive power on unseen data as the property of interest for data mining, we overcome a number of issues: model selection between competing models is trivial (choose whichever model predicts unseen data better), data mining results come with a degree of statistical generality (as opposed to being based on descriptive statistics like frequency), and the significance of a particular mined pattern is easy to assess based on the degree of predictability.

In many cases, most interactions present in a network might be unpredictable without external information, but the ones that are may be of scientific and commercial value. Some of these practical uses include, for example, (i) *community inference*, where regular interactions between two people can signify a special relationship between them, (ii) *marketing*, where advertising opportunities arise from knowing when a particular interaction between two people is going to occur, (iii) *unusual activity detection*, such as churn prediction (Wei and Chiu, 2002), which can be triggered by many of an individual's regular interactions ceasing to occur within a short period of time, (iv) *detecting plausible hidden relationships*, where two otherwise unconnected interactions between different sets of individuals are found to be temporally predictive, and (v) *spam detection* based on mechanistic and predictable interaction patterns, even when they are perturbed by random noise.

Specific types of predictable behavior have been studied independently in the literature, and one of our goals here is to automatically and transparently detect many such restricted subsets:

1. **Periodic patterns**, such as a weekly e-mail exchange between two people, which have been found at different timescales in many types of dynamic networks (Lahiri and Berger-Wolf, 2008).

2. **Bursty behavior**, such as periods of inactivity followed by intense periods of activity, has been observed in e-mail networks (Malmgren et al., 2009) and web server connections (Frias-Martinez and Karamcheti, 2003). Although the authors of (Malmgren et al., 2009) analyzed the aggregate behavior of users, and not specific interactions, there are compelling intuitive reasons (such as the human diurnal cycle) to believe that 'bursts' should be observed at the level of individual interactions as well.

3. **Temporal association rules**, where the occurrence of a particular event a leads to event b occurring after a fixed time interval (Oates et al., 1997; Tung et al., 1999; Frias-Martinez and Karamcheti, 2002; Lahiri and Berger-Wolf, 2007). A variant of temporal association rules is the *frequent episode* formulation (Mannila et al., 1997a; Laxman et al., 2005).

The methods we describe in this chapter model the temporal relationships above, and also have a number of other attractive features. A typical pre-processing step for dynamic network data is to quantize the time stream of interactions into coarser timesteps. Where the original dataset might have a timestamp for each edge at the resolution of about a second, typical pre-processing groups all edges into shifting windows of hours, days, or even months (see Chapter 2 for a description of various quantization levels used in the literature). Our techniques operate in continuous time and do not require any time quantization; no

information is lost, and the user is spared a parameter and an extra preprocessing step. Furthermore, our techniques operate on differenced time series, *i.e.,* by considering the time *delay* between edges rather than their occurrence times, and can thus handle short- and long-range interactions seamlessly within the same framework. Finally, our method is designed specifically for networks and takes advantage of network structure, rather than retrofitting techniques for more limited types of data by assuming, *e.g.,* independence of edges.

4.1 Problem Definition

We begin with interaction data in its unprocessed form: a continuous stream of interactions between uniquely identifiable entities data at the finest possible timescale. In cases like e-mail and phone call networks, this means that each interaction (edge) carries a timestamp on the granularity of about a second. In almost all other types of network analysis, timestamped interactions are quantized into a coarser timescale where timesteps comprise of hours, days, months, or even years. Multiple interactions within the same quantized timestep are either considered a single interaction, or weighted by the count of interactions within that timestamp. For our approach, however, we work in continuous time and do *not* require the user to quantize interaction data. We see this as an advantage of our method.

The focus of our method is to mine *coupled edges*. Figure 26 illustrates a pair of edges in a dynamic network stream that are coupled. In this case, occurences of the edge $(1, 3)$ reliably predict occurrences of edge $(1, 2)$ after a fixed time period, even though the opposite is not necessarily true. The mining problem is to extract these edges under two additional

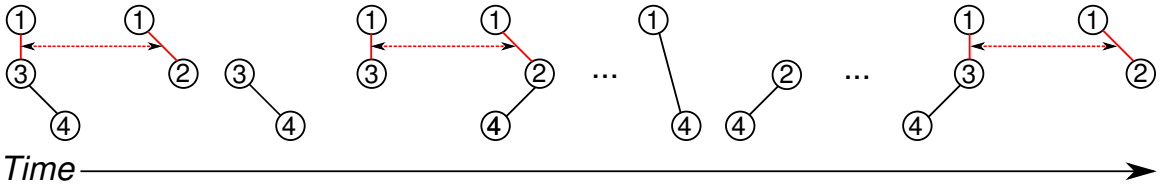


Figure 26: Streaming interaction data in continuous time with a coupled pair of edges shown in red. Note that the coupling is directional.

constraints: there is generally more noise than signal in the dataset (*i.e.,* most edges are likely to be uncoupled), and the temporal relationship between coupled edges might not be a simple fixed time delay, as shown in Figure 26. We review other approaches that make a fixed time delay assumption in Section 4.2.

Definition 4.1.1. *(Coupled edges)* An ordered pair of edges $\langle E_1, E_2 \rangle$ is *coupled* if occurences of E_2 can be predicted solely from prior occurrences of E_1, where possibly $E_1 = E_2$.

In the context of a data mining problem, Definition 4.1.1 is the property of interest, and we aim to develop tractable techniques that will extract pairs of edges from large quantities of data where the property holds. Definition 4.1.1 can be broken down into three components: how an edge is predicted, how the prediction of a particular algorithm is evaluated, and how that translates into a mining problem. We deal with each part in the subsequent subsections.

4.1.1 Prediction

Given a pair of candidate edges E_1 and E_2, we want to find the degree to which they are coupled by measuring how well E_1 predicts E_2. The method of determining which edge

pairs to test is part of our solution to the problem, and is described in Section 4.3. For now, we assume that we are given an edge pair, where E_1 is called the *trigger* and E_2 the *response*. Let the *timelist* of an edge (u, v) be the ordered sequence of timestamps at which it occurs, denoted $\mathcal{T}(u, v)$. We also assume that there is significant overlap in the timelists of the trigger and response. Without this assumption, the case of time-shifted coupling is very hard to detect. If, for example, five instances of a trigger within an hour are coupled to five instances of a response several hours later, it is hard to distinguish that relationship from noise, or an autocorrelation in the response.

The timelist of the response edge is segmented into training, validation, and testing segments. The training segment is used to learn the parameters of a model that will predict occurrences of the response based on historical observations. The validation segment, in our formulation, is used for model selection and to avoid overfitting, although other regularization or heuristic methods may be used as well. The prediction problem is then as follows.

Definition 4.1.2. *(Prediction)* Given a set of candidate models $M_i(\theta_i)$ each parameterized by θ_i, and training timelists for the trigger and response edges, find the optimal parameters of the model. Choose the model M' that maximizes an *evaluation score* E on the validation timelists $\mathcal{T}_V$.

$$M' = \arg\max_{M_i} E(M_i(\theta_i), \mathcal{T}_V)$$

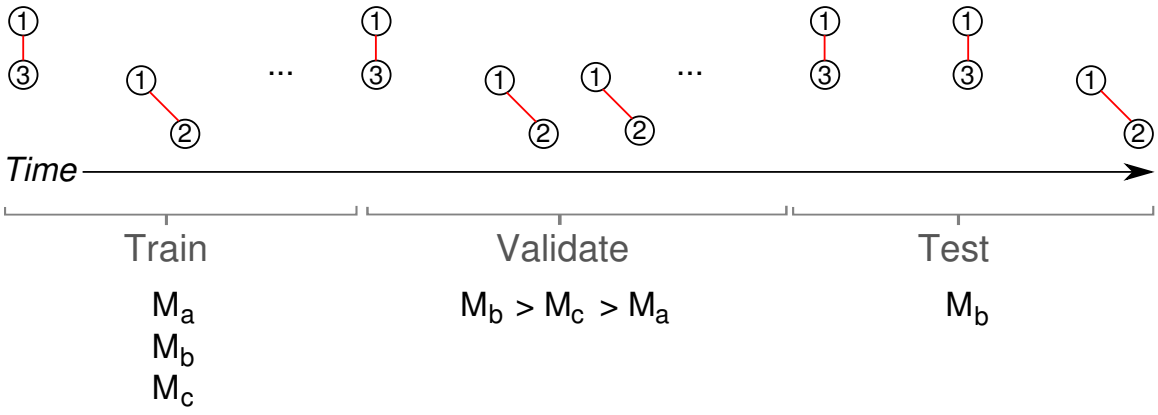


Figure 27: Setup for testing the predictability of a pair of edges.

Note that this describes a fairly generic machine learning setup. The final segment of the timelist will be used later for testing and mining. We use the validation segment to test the performance of multiple prediction models on unseen data, while still withholding a final test segment to evaluate the chosen model. A number of other statistical mechanisms can be used for model selection, such as the Bayesian Information Criterion (Raftery, 1999), but since our final goal is to test predictable edges, using predictive performance on unseen data is a logical choice.

4.1.2 Evaluation

A key factor of our problem formulation is that the timestamp for each observed interaction is a positive real value, and consequently, predictions of the next occurrence of

each interaction are also real values.[1] As a result, matching an edge's predicted occurrences to its observed occurrences is non-trivial, and traditional measures of prediction efficacy based on a confusion matrix, such as the F_1-score or ROC curves, cannot be used directly. Furthermore, since data is received on a streaming basis, predictions are made continuously and incorporate previously seen observations, which further complicates performance evaluation. We therefore propose a novel evaluation framework to handle both these facets of our problem.

To handle both these facets of our problem, we propose an evaluation framework that takes into account the fact that both observations and predictions of interactions occur in continuous time, and that data is received on a continuous, streaming basis.

For a particular edge, a prediction algorithm generates a *predicted* timelist $\mathcal{T}_P$, which is to approximate a *true* (observed) timelist $\mathcal{T}_T$. Since the prediction algorithm also has a partial view of $\mathcal{T}_T$ as it streams by, we are also given a *predicted from* timelist $\mathcal{T}_F$, which contains the timestamp at which each prediction $t_i \in \mathcal{T}_P$ was made, *i.e.,* if x_i is the i^{th} element of $\mathcal{T}_F$ and y_i the i^{th} element of $\mathcal{T}_P$, then the prediction algorithm generated a prediction of time y_i at time x_i, where naturally $y_i > x_i$.

We first define what qualifies as a *correct* prediction, and subsequently the relative numbers of true and false positives and negatives, and then *how close* (in continuous time)

---

[1] In practice, discrete-valued epoch seconds are used as timestamps for communications networks, but our argument applies generally to any fine-grained timescale.

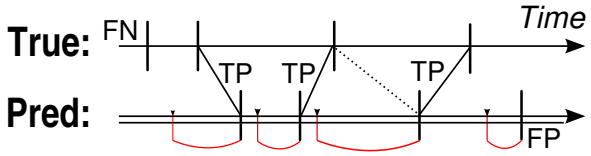


Figure 28: Example of matching a true and predicted timelist in continuous time in an online setting. Solid connecting lines are matched pairs, dotted lines are potential matches, and red lines show when each prediction was made.

correct predictions are to their corresponding true occurrences. This can be formulated as a matching problem, based on the following constraints:

1. A predicted occurrence should only be matched to *at most* one of the true occurrences that are adjacent to it in time, or not at all.

2. A predicted occurrence at t_i cannot be matched to a true occurrence at t_j if the algorithm predicted t_i *at or after* time t_j. This is an intuitive constraint due to the online setting of the problem, and also prevents a trivial algorithm that predicts an occurrence at $t_j + 1$ immediately after observing an occurrence at t_j from achieving near-perfect performance.

3. The benefit of matching a true occurrence $t_i \in \mathcal{T}_T$ and predicted occurrence $t_j \in \mathcal{T}_P$ should be inversely related to the time difference between them, *e.g.,* as $(|t_i - t_j| + \epsilon)^{-1}$, subject to the first constraint, and where ϵ is a small constant to ensure that the quantity is well defined for a perfect algorithm.

Specifically, the constraints above can be formalized as a weighted bipartite maximum matching problem (West, 2001). Each element of $\mathcal{T}_P$ and $\mathcal{T}_T$ is represented by a node in the two corresponding partitions of a bipartite graph. An edge connects $t_i \in \mathcal{T}_P$ and $t_j \in \mathcal{T}_T$ if they are sequentially adjacent in time (with no other elements of $\mathcal{T}_T$ or $\mathcal{T}_P$ in between) and satisfy constraints 1 and 2. The edge weight of (t_i, t_j) is defined as $(|t_i - t_j| + \epsilon)^{-1}$ for some small constant ϵ. Efficient algorithms exist for finding a maximum-weight matching in such a graph (Galil, 1986). Figure 28 shows an example of the result of matching a predicted and a true timelist, where the red loops represent the *predicted from* timelist.

If M is the set of matched edges, then some traditional measures are conveniently expressed in terms of the cardinality of M, *i.e.,* the number of matched pairs:

$$\textit{Precision} = |M|/|\mathcal{T}_P|$$

$$\textit{Recall} = |M|/|\mathcal{T}_T|$$

$$\textit{Mean Absolute Error (MAE)} = \frac{1}{|M|} \sum_{(t_i,t_j)\in M} |t_i - t_j|$$

We summarize the Precision and Recall measures into their geometric mean, the F_1-score, and characterize the performance of a prediction algorithm on a single timelist in terms of the F_1-score and the MAE. The former gives a sense of the *completeness* of the predictions, *i.e.,* how well the number and temporal spacing of predictions approximate $\mathcal{T}_t$, and the latter of their *temporal accuracy.* A perfect prediction algorithm would have an F_1-score of 1 and an MAE of 0. Note that it is trivial to construct cases where one quantity

is completely maximized or minimized at the cost of the other, and hence both measures need to be taken into account. We refer to the space of (F_1, MAE) pairs as the *evaluation plane*.

It is also often necessary to determine whether a given (F_1, MAE) pair of values is in some sense 'better' than another pair. This is needed, for example, in comparing the performance of two different algorithms or parameter sets, and for the final mining task described in the next subsection. The case of trivial dominance in a ranking is straightforward: one pair will have a higher F_1 score *and* a lower MAE than the other pair. However, for non-trivial cases, we introduce a parameter η that specifies the relative cost of improving one measure over the other. It can be interpreted as the *maximum increase in MAE for a unit gain in F_1-score* for one pair of values to dominate another. As an example, consider the evaluation plane shown in Figure 29, where the point labeled **A** represents a perfect prediction algorithm. The slope of the line connecting **C** and **D** is equal to η, and thus both points are considered equivalent in a ranking. Thus, an overall ranking for the points in Figure 29 is: $A, B, E, \{C, D\}$.

4.1.3 Mining

Finally, we return to the mining problem of finding the most tightly coupled edges. After model selection is done on the validation segment, the final withheld segment of testing data is used to assess the predictive accuracy of each model. The evaluation scores on this final test segment are used to determine the order of data mining results. Although it would seem that there are a number of tractability issues with the mining framework described in

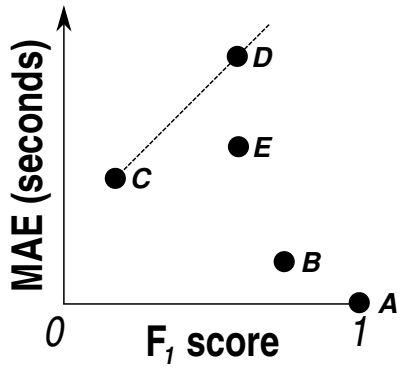


Figure 29: Some points in the evaluation plane.

this section, we show how to overcome them by assuming a smaller (but still meaningful) dependency space in Section 4.3.

4.2 Related Work

In this section, we review literature directly related to mining coupled edges in dynamic networks. There are two broad categories of related work: a similar problem in network analysis called *link prediction*, and a number of approaches for other types of data that could, in principle, be applied. We outline the similarities, differences and shortcomings of these approaches in this section.

4.2.1 Link Prediction

There are two variants of the *link prediction* problem for networks: one deals with a single graph in which missing links are to be predicted based on some model of link formation, and another formulation that deals with predicting links over time in dynamic networks. An important difference between static and dynamic link prediction is that the former does

not make predictions on a fine temporal scale, but rather just about which new links are likely to be formed 'in the future'. Both versions, however, are generally classification or ranking problems, and as such, focus less on the interpretability of the model than predictive performance. Although we describe both variants below, our problem has more in common with dynamic link prediction.

4.2.1.1 Static networks

The link prediction problem for static networks is defined as the question of ranking *unseen* edges by their likelihood of appearance in the future, without specifying *when* the edges will form. It therefore deals with the prediction of link formation, rather than the next occurrence of a link in a dynamic setting. The premise of this approach is that social networks tend to have structural similarities throughout the network that can be exploited to discover links that are either missing, or likely to form in the future. Liben-Nowell and Kleinberg (Liben-Nowell and Kleinberg, 2003) were one of the first to define the link prediction problem for social networks. They split a dynamic network along the time axis to form training and testing networks. Various structural node and edge measures were computed on the static network created from the first half of the dynamic network. Unobserved edges were then ranked by their probability of occurrence in the test network. They report promising results at predicting link formation using certain graph-theoretic measures, compared to a random and other simple baseline predictors.

A minor variation of the previous approach using weighted graph measures is described in Murata and Moriyasu (Murata and Moriyasu, 2007). Kashima and Abe (Kashima and

Abe, 2006) build upon the work of Liben-Nowell and Kleinberg (Liben-Nowell and Kleinberg, 2003) and others by first defining a model of network evolution that describes each edge as a probability. An edge label function $\phi^{(t)}$ is used to assign probabilities of existence to each edge, and the evolution of ϕ over time is assumed to be a Markov process. Specifically, a node in the network decides to 'transfer' an edge probability of one of its incident edges to an edge incident on another node in the network. The parameters of the model are the probabilities of each edge pair engaging in such a transfer. The authors describe a transductive algorithm based on Expectation-Maximization that jointly estimates the probabilities of each 'test' edge as the model is learned. It is worth noting that this model is superficially related to the *triadic closure* in social network analysis, which is the assumption that open triangles in a social graph tend to close with high probability (Wasserman and Faust, 1994; Kossinets and Watts, 2006).

O'Madadhain et al. (O'Madadhain et al., 2005) address link prediction in explicitly temporal event data, as well as the evolution of entity rank (importance) over time. For the link prediction component of their paper, the methodology used is similar to Liben-Nowell and Kleinberg (Liben-Nowell and Kleinberg, 2003) in that a dynamic network is split into two segments along the time axis for training and testing. However, instead of using structural graph features to rank unseen edges as in (Liben-Nowell and Kleinberg, 2003), O'Madadhain et al. treat the problem as a classification problem, and use both arbitrary entity attributes (*e.g.* geographic proximity, similarity of publication patterns (O'Madadhain et al., 2005))

as well as structural features of the graph for link prediction. The learning techniques used are Naive Bayes and logistic regression.

Finally, a number of approaches to link prediction are based on the recent paradigm of *statistical relational learning* (Getoor and Taskar, 2007). Wang et al. (Wang et al., 2007) learn undirected graphical models in the neighborhood of a pair of nodes whose edge incidence is to be predicted. Popescul and Ungar (Popescul and Ungar, 2003) approach the task by combining logistic regression with a feature generation algorithm that aggregates SQL queries on node attributes to predict future edges in a bibliographic database. An overview of work in this area can be found in Getoor et al. (Getoor et al., 2003) and Jensen and Neville (Jensen and Neville, 2003).

4.2.1.2 Dynamic networks

Dynamic link prediction is defined as the prediction of *when* previously observed and new interaction are going to occur again in the future, based on temporal and structural correlations. There are two related, but not necessarily equivalent, variants of the structure prediction problem:

1. **(Next Step Prediction)** Given a dynamic network $\mathcal{G}$ of t timesteps, predict a set of interactions (edges) that will occur exactly at timestep $t + 1$.
2. **(Next Occurrence Prediction)** Given a dynamic network $\mathcal{G}$ of t timesteps, predict the future timestep when each interaction will next occur.

Huang and Lin (Huang and Lin, 2009) describe a method for both variants using a combination of static graph measures and standard time-series prediction techniques (Chatfield,

2004). Specifically, they fit an ARIMA[1] model to the time series of binary occurrences or occurrence frequencies of each edge, independently of all other edges. The edge occurrence scores for the next timestep are then blended with the occurrence scores based on various static graph measures, like the ones described in Liben-Nowell and Kleinberg (Liben-Nowell and Kleinberg, 2003). Due to large number of edges in a typical dynamic network, and the parameter optimization required to fit a single ARIMA model, this approach is unlikely to scale well to realistic networks. Furthermore, their results suggest that incorporating the structure of the network, such as the inter-dependencies between edges, into a predictive model would be advantageous. Although time series models are generally interpretable and applicable to mining coupled edges, the approach here would only allow the mining of coupled autocorrelations (*i.e.,* it would not be possible to mine a coupling between two *different* edges).

An interesting application of network prediction is the approach of Bunke et al. (Bunke, 2003; Bunke et al., 2005) in network anomaly detection. Given a stream of network traffic data, they use decision trees and 'median graphs' to detect connection anomalies. The trees are learned from a sequence of training data, where no anomalies are presumed to be present. This is extended by Pincombe (Pincombe, 2005), who uses Auto-Regressive Moving Average (ARMA) techniques instead. Since their focus is on predictive accuracy, their approach is not applicable to our mining problem.

---

[1]Autoregressive Integrated Moving Average

Phithakkitnukoon and Dantu (Phithakkitnukoon and Dantu, 2007) propose a method for predicting whether a phone user will receive an incoming call at various hours of the day, without attempting to predict who the call is originating from. They make various simplifying assumptions, such as the call inter-arrival time and the number of outgoing calls per incoming call both following a Gaussian distribution. They report an error rate of approximately 5% on the call logs of 20 individuals, with no recall or specificity values.

Lahiri and Berger-Wolf (Lahiri and Berger-Wolf, 2007) describe an online technique that probabilistically estimates the delay between pairs of edges and uses this delay to predict a set of edges that will appear at an arbitrary point in the future. To aid the tractability of the approach, they measure the delay between frequent subgraph pairs instead of edge pairs. A simple heuristic mechanism dynamically adjusts which edge/frequent subgraphs pairs are used to make predictions. A limitation of this approach is that frequent subgraphs, or generally subgraphs of interest, have to be pre-specified. This limitation can be overcome by restricting the edge pair correlation to pairs of edges that are likely to be correlated, e.g. those edge pairs that share a common vertex.

Finally, Acar *et al.* (Acar et al., 2009) describe the use of matrix and tensor decompositions for solving both static and dynamic link prediction problems. For the static variant, they use a truncated low-rank Singular Value Decomposition (SVD) of a time-weighted graph, and then predict future edge formation by the scores of each (i, j) entry in the reconstructed matrix. Although the SVD of a sparse matrix can be computed very quickly, re-assembling all link prediction scores in the most general case requires iterating over all

cells in the adjacency matrix, an intractable $O(V^2)$ operation for large networks. For the dynamic variant of the problem, they use standard time series prediction on the coefficients of a CANDECOMP/PARAFAC (CP) tensor decomposition of a dynamic network (see (Faber et al., 2003) for a recent review). Although this requires choosing some parameters, it has the advantage that CP tensor decompositions carry a lot of easily interpretable information and can be used for data mining.

As we noted before, the prediction algorithms presented here, with the exception of the CP tensor decomposition of (Acar et al., 2009) and the structure prediction approach of (Lahiri and Berger-Wolf, 2007), are generally black boxes from which it is difficult to extract meaningful relationships between edges.

4.2.2 Event Prediction

Event prediction aims to learn signatures that precede certain target events in a stream of symbolic or numerical data. The overall goal is to be able to predict occurrences of specific events before they happen, such as fraud, hardware failure, or intrusion into a computer network, by monitoring a stream of log messages. It can be seen as a special case of dynamic network next-step prediction, where we do not want to predict the entire graph at timestep t in a dynamic network, but just a specific part of it. y It is also more specialized in the sense that it generally deals with low-dimensionality streams, and can sometimes be treated as a conventional supervised learning problem. As a result, much of the work in this area has focused on specific applications instead of general techniques.

Fawcett and Provost (Fawcett and Provost, 1999) call this class of problems 'activity monitoring', and outline several important issues in how it differs from conventional classification problems. Among their contributions are a generic framework for modeling event prediction problems and an illustration of how prediction performance should be quantified. They point out that using accuracy or error to judge the merits of an event prediction model has its shortcomings, and suggest using ROC curves (Fawcett, 2004) or a specialized variation for activity monitoring, the AMOC curve (Fawcett and Provost, 1999).

A number of approaches to event prediction use conventional classifiers like Support Vector Machines (SVMs) or rule-based ones built from mining association rules, especially in the context of hardware failure prediction from event logs and network intrusion prediction. Vilalta and Ma (Vilalta and Ma, 2002) mine frequent events from a time window preceding target events, and then combine these rules into a classifier. Domeniconi *et al.* (Domeniconi et al., 2002) combine Singular Value Decomposition on the co-occurrence of events with an SVM to phrase the problem as a conventional classification problem. Liang *et al.* (Liang et al., 2006) look for simple correlations in failure patterns in supercomputer hardware event logs. A more sophisticated approach to the same problem using Hidden Semi-Markov Models is adopted by Salfner and Malek (Salfner and Malek, 2007). Finally, an interesting approach in a different, numerical time-series domain involves using a support feature machine to perform subset selection as well as prediction of target events (Chaovalitwongse et al., 2007).

Various forms of intrusion detection in computer network traffic analysis can also be thought of as event prediction, where the 'event' being predicted is an anomalous usage pattern indicative of an intrusion. Kannadiga *et al.* (Kannadiga et al., 2007) treat the intrusion prediction problem in much the same way as the event prediction approaches mentioned earlier, although the connection is not explicit. Given a stream of network traffic data, Bunke *et al.* (Bunke, 2003; Bunke et al., 2005) use decision trees and median graphs to detect anomalies. This is extended by Pincombe (Pincombe, 2005), who uses Auto-Regressive Moving Average (ARMA) techniques instead.

A major limiting factor in modeling dynamic network data as a log stream is scalability. Naively, one might consider each edge in a dynamic network as a type of log symbol, much like the itemset representation mapping we used in Chapter 3. However, there are two problems with this: all structural information about the graph is lost, and the number of symbols scales with the number of unique edges. Since many of the approaches mentioned in this section were designed with relatively small alphabets in mind, they might not scale well when the alphabet size increases to hundreds of thousands or even millions of edges. Furthermore, since we cannot exploit network structure to reduce the space of dependencies, the sheer number of possible dependencies between symbols quickly becomes intractable.

4.2.3 Sequential Patterns, Frequent Episodes and Association Rules

Mining association rules and various forms of frequent patterns are probably among the oldest and best studied problems in data mining. Association rules in the form of $a \rightarrow b$ are expressions of the conditional probability of co-occurrence of sets of events (Agrawal

and Srikant, 1994; Brin et al., 1997). A generalization of this question considers temporal association rules where a and b occur in different transactions or timesteps, usually accompanied by an estimate of the distance or delay between them (Tung et al., 1999; Oates et al., 1997; Oates and Cohen, 1996). Frequent episodes are frequently occurring temporal partial orders of events, where the entire partial ordered must be matched within a fixed time window (Mannila et al., 1997b). Frequent sequential patterns are also closely related to frequent episodes (Pei et al., 2004; Harms and Deogun, 2004).

Tung *et al.* (Tung et al., 1999) introduce the mining of *inter-transactional association rules*, where the antecedent and consequent of the rule consist of disjoint itemsets and are separated by a temporal interval. They describe an algorithm where a user-defined parameter W controls the maximum allowable delay between the sides of each rule, and the *confidence* of the rule expressed the conditional probability of seeing the antecedent in *at most* W timesteps. Similarly, Oates *et al.* (Oates et al., 1997; Oates and Cohen, 1996) introduce almost identical notation for what they call *temporal dependencies*. The difference from Tung *et al.* is that instead of having a window in which both antecedent and consequent must occur, the rules express the actual delay to expect between the antecedent and the consequent.

Both association rules and frequent episodes can used to build rule-based predictive models, usually by taking some subset of the most frequently occurring associations. An early system for building such a rule-based prediction model was CBA (Liu et al., 1998), which is a non-temporal methodology that operates on transactional databases. Associ-

ation rules can also be used as a type of feature generation process to aid further classification (Lesh et al., 1999). More recently, a connection between frequent episodes and Hidden Markov Models has been made, with applications in building a predictive model for streaming data (Laxman et al., 2008; Laxman et al., 2005). They model a stream of events using a mixture model of frequent episodes mined from the stream, with sequential correlation between the mixtures. We build on their ideas by modeling the delays between interactions in a network, instead of modeling the occurrences of events themselves.

Mining algorithms for association and related rules generally use support (empirical frequency) as the property of interest, although a wide variety of other measures have been developed (Geng and Hamilton, 2006). Our mining formulation varies from these approaches in that it does not use descriptive statistics on a single dataset as the mining criteria, but rather predictive models and performance on an unseen segment of data. Our approach also differs from mining approaches such as (Oates et al., 1997) and (Laxman et al., 2008) because we explicitly take advantage of the network structure of data. As mentioned in the previous section, although it is possible to retrofit dynamic networks into a multidimensional discrete symbol stream that can be mined by some of these algorithms, all connectivity information (and subsequently the 'symbol' dependency structure) is lost in such a transformation. The space of possible rules is therefore much larger when the methods described in this section are applied, making the algorithms more intractable. Finally, even though our coupled edge rules superficially resemble the $a \rightarrow b$ rules mined by (Oates et al.,

1997), the $\rightarrow$ operator in our case can be a complex HMM-modeled sequential dependency, instead of a fixed constant as used by (Oates et al., 1997) and (Tung et al., 1999).

4.3 Modeling time delays

We now present our primary contribution in this chapter: a flexible model of regular behavior, which encompasses many previously studied forms of predictable patterns, is insensitive to the timescale at which such patterns occur, and does not require time to be discretized. It is a means for solving the prediction part of the mining problem described in Definition 4.1.2, and is the core of our mining algorithm.

Consider the two timelists shown in Figure 30, which are typical examples of interactions that can be modeled easily for future prediction. The upper timelist is an example of an interaction that occurs in 'bursts', such as a phone call between two people that occurs mainly during business hours. The lower timelist is an example of a periodically recurring interaction. Our model is based on three assumptions about these timelists: first, that the time delays between consecutive occurrences are drawn from a mixture distribution, second, that there is significant sequential correlation between consecutive time delays, and finally, that the dynamics of a particular edge are stationary for as long as the edge persists.

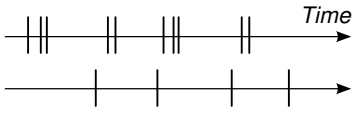


Figure 30: Examples of timelists.

We describe a novel Hidden Markov Model (Rabiner, 1989) (HMM) formulation for modeling time delays between a *trigger* and a *response* edge, denoted $E_T = (u, v)$ and $E_R = (x, y)$ respectively, and collectively referred to as an *edge pair*. E_T and E_R do not have to be solitary or even distinct interactions, but could also be, for example, frequently occurring subgraphs of interactions, or external events, such as $E_T = \{\texttt{First day of month}\}$ and $E_R = (a, b)$.

We model independent temporal dependencies over two types of edge pairs: those in which $E_T = E_R = (u, v)$, or autocorrelations, and pairs in which $E_T \neq E_R$. On observing an occurrence of E_T at time t, an HMM with continuous emissions specific to the $E_T \rightarrow E_R$ edge pair is used to generate a time delay $\delta \in \mathbb{R}^+$ until the next occurrence of E_R, thus generating a prediction of E_R at time $t + \delta$. Depending on its state, however, the HMM might also not generate a prediction at all, which allows more complex relationships to be modeled. We refer the reader to Rabiner (Rabiner, 1989) for more background on canonical HMMs and the notation used here.

A key contribution of our formulation is to model *time delays* between interactions using an HMM, instead of the more conventional approach of directly modeling the presence or absence of an interaction at each discrete timestep (Bunke et al., 2005; Huang and Lin, 2009). By dealing with this higher-order representation, we overcome two problems that affect dynamic network analysis: the need to quantize interactions into discrete timesteps, and the fact that regular behavior can exist at different timescales (Lahiri and Berger-Wolf, 2008), which can be missed by approaches that use fixed length windows of real time for

learning. As a result, it offers significant benefits over other conventional approaches and mining methodologies: there is no need to determine how far back in real time a model should retain data for learning, as is the case with time-series models (Huang and Lin, 2009), and there is no need to determine how much real time should be quantized into a timestep (Lahiri and Berger-Wolf, 2007; Huang and Lin, 2009; Bunke et al., 2005).

4.3.1 Model Setup

Given a trigger timelist $\mathcal{T}(E_T) = \langle t_1, ..., t_A \rangle$ and a response timelist $\mathcal{T}(E_R) = \langle r_1, ..., r_B \rangle$, we first compute a set of *delay pairs* that generate or approximate $\mathcal{T}(E_R)$ based on $\mathcal{T}(E_T)$. Each delay pair is generated by a trigger event and is of the form $\langle c_t, d_t \rangle$, where c_t is a binary value and d_t is a non-negative continuous value. The delay pair represents the action that a prediction algorithm should take to generate a response event at the correct time. If $c_t = 0$, then no response should be predicted. Otherwise, a response should be predicted to occur after a delay of d_t from the current time. The following algorithm computes a sequence of delay pairs from two timelists.

1. Sort the two timelists together into a single timelist $\mathcal{T} = \langle t_1, ..., t_{A+B} \rangle$. In cases of ties, sort responses *before* triggers. Let $type(t_i) =$ `trig` if element t_i came from the trigger timelist, and $type(t_i) =$ `resp` otherwise.

2. For each element t_i in the sorted timelist, where $i < A + B$ and $type(t_i) =$ `trig` :

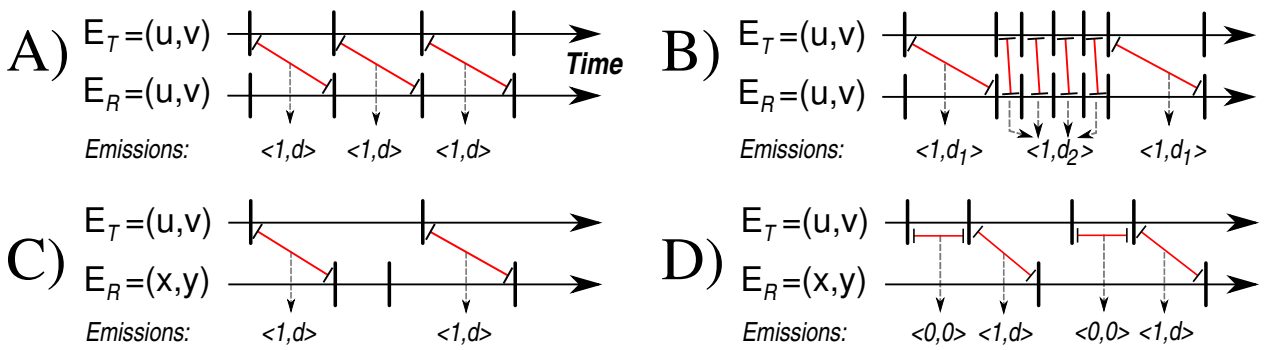
   (a) If $type(t_{i+1}) =$ `trig`, output the delay pair $\langle 0, 0 \rangle$.

   (b) Otherwise, output the delay pair $\langle 1, t_{i+1} - t_i \rangle$.

N	number of HMM states
$A_{N\times N}$	HMM state transition matrix
π_k	Initial probability of HMM state k
μ_k, σ_k^2	mean and variance of Gaussian emission distribution in state k
τ_k	probability of a continuous delay emission in state k
λ	set of all HMM parameters (all the above)
$q_t \in [1, N]$	HMM state at time t
$\phi(\mu, \sigma^2)$	Gaussian density with parameters (μ, σ^2)
$o_t = \langle c_t, d_t \rangle$	HMM emission at time t consisting of binary c_t and continuous $d_t \geq 0$
$b_k(o_t)$	Probability of emission o_t from state k
$\alpha_i(t)$	Forward variable; joint probability of observations up to time t and final state i
$\Gamma_i(t)$	Probability that emission at time t was generated by state i, given data up to t
$\gamma_i(t)$	Probability of being in state i at time t, given the entire sequence of observations

TABLE VI: NOTATION USED FOR HMM FORMULATION.

Note that if $E_T = E_R$, then $c_t = 1$ always. Figure 31 illustrates the generation of delays pairs from a trigger and response timelist.



Special case	*HMM states*	*Example Transition Matrix*	*Emission Probabilities*
(A) Partial periodic patterns	1	$\begin{pmatrix} 1 \end{pmatrix}$	$\mu_1 = d$ (period) $\tau_1 = 1$
(B) Bursty behavior	2	$\begin{pmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{pmatrix}$	$\mu_1 = d_1 \quad \tau_1 = 1$ $\mu_2 = d_2 \quad \tau_2 = 1$
(C) Temporal association rules	1	$\begin{pmatrix} 1 \end{pmatrix}$	$\mu_1 = d_1 \quad \tau_1 = 1$
(D) Complex temporal association rules	2	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\mu_1 = 0 \quad \tau_1 = 0$ $\mu_2 = d \quad \tau_2 = 1$

Figure 31: Special cases of structure prediction, with corresponding HMM delay pair emissions. Red lines indicate time delays. The table shows examples of delay prediction HMMs that handle the special cases.

4.3.2 HMM Structure

Given a sequence of delay pairs, we use an HMM to model the sequence, where the emissions of the HMM are the delay pairs, and the hidden states correspond to different distributions (clusterings) over time delays. Let the emission of the HMM at position t be denoted $o_t = \langle c_t, d_t \rangle$, where $c_t \in \{0, 1\}$ and $d_t \in \mathbb{R}$. Unlike a typical HMM which generates emissions at every timestep of a discrete time process, our formulation calls for an emission on every incident of a trigger E_T being observed, with the continuous part of the emission specifying a time delay to the next occurrence of the response E_R, conditional on the binary part of the delay pair being true. When $c_t = 0$, the value of d_t is irrelevant, and we define it to be 0 for notational convenience. We chose a univariate Gaussian distribution to model the continuous delays emitted in each hidden state, conditional on $c_t = 1$, but any univariate continuous distribution may be substituted in its place without changing the framework.[1]

---

[1]Note that although call *durations* appear to be distributed according to a log-logistic function (Vaz de Melo et al., 2010), we model the time between call *initiations*.

The distribution of emissions in each hidden state is governed by three parameters: τ_k, μ_k, σ_k.

$$P[c_t = 1|q_t = k] = \tau_k$$

$$P[c_t = 0, d_t = 0|q_t = k] = 1 - \tau_k$$

$$P[d_t|q_t = k, c_t = 1] \sim \phi(\mu_k, \sigma_k^2)$$

Assume that the HMM has N hidden states, with λ representing the set of all HMM parameters (see Table Table VI). Based on the expression above, the probability b_k of an emission $o_t = \langle c_t, d_t \rangle$ from state k is defined as:

$$\begin{aligned} P[o_t|q_t = k, \lambda] &= b_k(\langle c_t, d_t \rangle) \\ &= \begin{cases} \tau_k \phi(\mu_k, \sigma_k^2) & : c_t = 1 \\ (1 - \tau_k) & : c_t = 0 \end{cases} \end{aligned} \tag{4.1}$$

In standard HMM theory, the Baum-Welch algorithm is used to estimate maximum-likelihood parameters for an HMM from data, given the number of states N and a suitable parametric form of the emission density b_k at each state k. The emission density in Equation Equation 4.1 is one such suitable parametric form, and the standard Baum-Welch algorithm can be used to estimate HMM parameters from a sequence of paired emissions $O = \langle o_1 = \langle c_t, d_t \rangle, ..., o_T \rangle$ calculated from the training network segment. The derivation is

identical to that of the canonical Baum-Welch algorithm (Rabiner, 1989), and we only list the final parameter update equations in terms of delay pairs (the M-step of the Baum-Welch EM algorithm) for completeness. Refer to Table Table VI for notation.

$$\mu_i = \frac{\sum_{t=1}^{T} \gamma_i(t)(c_t \cdot d_t)}{\sum_{t=1}^{T} \gamma_i(t) \cdot c_t}$$
$$\sigma_i^2 = \frac{\sum_{t=1}^{T} \gamma_i(t)(c_t \cdot (d_t - \mu_i)^2)}{\sum_{t=1}^{T} \gamma_i(t) \cdot c_t}$$
$$\tau_i = \frac{\sum_{t=1}^{T} \gamma_i(t) \cdot c_t}{\sum_{t=1}^{T} \gamma_i(t)}$$

4.3.3 Prediction

Once the Baum-Welch algorithm has been used to estimate HMM parameters on the training segment, we use the learned model to make predictions on the validation or test segments. We know from the Forward procedure in HMM training that the forward variable $\alpha_i(t) = P[O_t, q_t = i|\lambda]$, which is the joint probability of the observation sequence ending in the state i at time t. We also know that $P[O_T|\lambda] = \sum_{i=1}^{N} \alpha_i(T)$, which is exactly the output of the Forward procedure. In an online setting, marginalizing the α variables yields the probability of being in a particular state i at time t, which we denote $\Gamma_i(t)$[1].

$$\Gamma_i(t) = P[q_t = i|O_t, \lambda] = \frac{\alpha_i(t)}{\sum_j \alpha_j(t)} \tag{4.2}$$

---

[1] $\Gamma_i(t)$ is distinct from the $\gamma_i(t)$ variable in the Baum-Welch algorithm, which takes backward probabilities into account.

The Γ variables define a distribution over the states that caused the last observed emission. For prediction, however, we require the next likely emission, which in turn requires a distribution over the next state, *i.e.,* $P[q_{t+1}|O_t, \lambda]$. This is easily computed from the HMM transition matrix and the $\alpha_i(t)$ variables (Rabiner, 1989), following which we use two different methods to generate the next emission (delay prediction):

1. *(Expectation)* We first compute the expected probability of the next emission having a continuous component by averaging over the continuous emission probability τ_i of each state:

$$E[c_{t+1}] = \sum_i P[q_{t+1} = i|O_t, \lambda] \cdot \tau_i$$

If $E[c_{t+1}] > 0.5$, then we assume that the next emission will produce a time delay, so we compute d_t as the expected delay over all HMM states.

$$E[d_{t+1}] = \sum_i P[q_{t+1} = i|O, \lambda] \cdot \mu_i$$

The final prediction is then

$$o_{t+1} = \begin{cases} \langle 1, E[d_{t+1}] \rangle & : E[c_{t+1}] > 0.5 \\ \langle 0, 0 \rangle & : E[c_{t+1}] \leq 0.5 \end{cases}$$

2. *(Maximum a posteriori)* Instead of computing an expected delay over all states, we can instead pick the most likely next state q' and generate an emission from it. This might be preferable in some cases, since delay predictions will not lie in between the

expected delays of each HMM state, but will rather be drawn from a single HMM state.

$$q' = \arg\max_i P[q_{t+1} = i | O, \lambda]$$

$$o_{t+1} = \begin{cases} \langle 1, \mu_{q'} \rangle & : \tau_{q'} > 0.5 \\ \langle 0, 0 \rangle & : \tau_{q'} \leq 0.5 \end{cases}$$

4.3.4 Model selection and special cases

The only parameter in the learning and prediction processes described in the previous sections, other than the global (F_1, MAE) trade-off parameter η, is the number of HMM states N for each trigger-response pair, and whether to use maximum *a posteriori* (MAP) or expectation to generate predictions. In general, the number of HMM states and its connection topology is generally determined either by prior domain knowledge or other, sometimes heuristic, mechanisms (Rabiner, 1989). However, since our objective is prediction, we fit models of various state sizes up to a small maximum number, using both MAP and expectation to generate predictions, and then choose the combination that produces the highest ranked (F_1, MAE) pair on the validation segment. The allows us to select a specific model, and also helps avoid overfitting on the training data.

The value that we suggest for the maximum number of HMM states is based on both a theoretical and an empirical argument. Figure 31 shows a number of special case HMMs that cover many previously studied types of regular behavior. In all the cases shown, two

HMM states are sufficient to model interaction dynamics, suggesting that the maximum number of states to attempt fitting can be similarly low. Furthermore, a 3-state HMM in our formulation has 17 non-trivial parameters to estimate, so if the timelists of interactions are not sufficiently long, then the inference of parameters is likely to be noisy. In the datasets we examine, we found typical timelists to be quite short, which again supports the use of a small maximum number of HMM states. For these reasons, we suggest that a maximum of 3-state HMMs be used to model interaction dynamics in networks comparable to the ones we analyze here.

4.3.5 Tractability

With single interactions comprising E_T and E_R, the approach above is not computationally tractable even for relatively small networks with about 10^4 unique edges. We overcome this using two methods: the first is to only consider edges with timelists long enough to support meaningful statistical inference. Generally, this means that the number of available timelist points should be at least a reasonable constant factor larger than the number of HMM parameters to be estimated. As we will show in Section 1.3, this eliminates a large number of the edges in real networks. The second method is to only consider temporal correlations between pairs of interactions where, (a) $E_T = E_R = (u, v)$, or autocorrelations, and (b) $E_T = (u, v)$ and $E_R = (u, w)$, or pairs of interactions that share a common node.

Modeling correlations between pairs of edges that share a common vertex is tractable as a side effect of the skewed degree distributions observed in many real-world networks (Newman, 2003). The number of delay pairs D is equal to the number of edge pairs that share

a vertex, which is exactly the number of edges in the line graph[1] of the original network. This number D in turn is proportional to the sum of the squares of degrees in the original graph (Lehot, 1974), *i.e.,* $D = \sum_{i=1}^{|V|} \frac{d_i(d_i-1)}{2}$, where d_i is the degree of vertex i. Modeling all such dependencies in real-world networks is therefore tractable if the assumption of a degree distribution skewed towards smaller degrees holds, as it generally has been found to (Newman, 2003; Leskovec and Horvitz, 2008; Nanavati et al., 2006; Chakrabarti and Faloutsos, 2006).

We also note that our algorithm is trivially parallelizable, and should scale linearly with the number of processors used.

4.4 Experimental Results

We ran our mining algorithm on the datasets described in the previous section to determine the extent to which edges in real networks are predictable. We describe two types of results: those specific to our learning algorithm, and applications of mining predictable interactions in general.

In all cases, we designated consecutive thirds of the timelist of each edge to serve as training, validation, and test segments, with the results of mining determined by performance only on the test segment. The maximum number of HMM states to fit is set to 4, in line with the reasons described in Section 4.3. We also require that all edges have

---

[1]Recall that the line graph of graph $G = (V, E)$ contains a vertex v'_i for every edge $e \in E$ in G, with an edge connecting v'_1 and v'_2 in the line graph if the corresponding edges $e_1, e_2 \in E$ in the original graph share a common vertex.

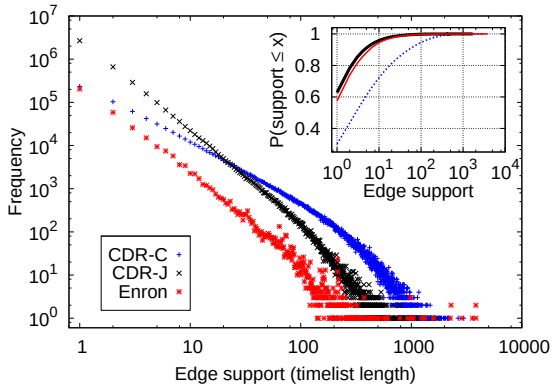


Figure 32: Edge support distributions: histogram on a doubly logarithmic scale, and empirical cumulative distribution on a partial logarithmic scale (inset).

timelists of length at least 40 in order to be used for learning and prediction. As shown in Figure 32, all three datasets show a commonality in terms of heavy skews in the *support* (timelist length) distribution of edges: only a small percentage of edges in each dataset have timelists of 40 occurrences or more.

Finally, we determined empirically that the η parameter has little impact on prediction performance, either quantitatively in terms of the mean and median (F_1, MAE) scores over all edges, or qualitatively in terms of the most predictable edges. We omit the results for brevity, but one possible reason for the insensitivity of the algorithm to η is that there are few edge pairs where different HMMs or parameter sets offer Pareto-optimal performance for predicting interactions. We therefore use a value of $\eta = 1000$ for our experiments, which corresponds to a trade-off, in our implementation, of at most 15 minutes in MAE for an increase of 0.1 in F_1-score.

We ran two sets of experiments to test our algorithm. The first set is on synthetic data to show that our algorithm is able to distinguish between noise and regular interactions of the type described in Section 4.3. The second set of experiments is performed on the real datasets described in the previous section, and illustrates a number of practical applications.

4.4.1 Synthetic Data

We created synthetic data by randomly generating HMMs of the type described in Section 4.3. We created a different edge for each HMM, and placed 1,000 instances of the interaction on a timeline by iterating the HMM to generate delays. A total of 600 such interactions were generated by HMMs of 1, 2, and 3 states, in equal proportions, which encompasses the special cases shown in Figure 31. Finally, 200 additional 'noise' interactions were created by placing instances of the interaction uniformly at random on the timeline.

We designated consecutive thirds of the timeline of each interaction to serve as training, validation, and test segments. Figure 33 shows the distribution of F-1 and MAE scores of the fitted models for each type of edge on the test segment, where the boxes represent the interquartile range of the data. There is a clear differentiation between the MAEs of interactions generated by HMMs and those generated randomly. In the left figure, the 'noise' edges have the lowest F-1 score and the highest MAE, corresponding to the lower left part of the evaluation plane. Edges generated by a 1-state HMM, which is essentially an autocorrelated edge with a Gaussian time delay, have the highest scores. Even 3-state HMMs, which can generate a complex pattern of delays, are detected with greater efficacy –

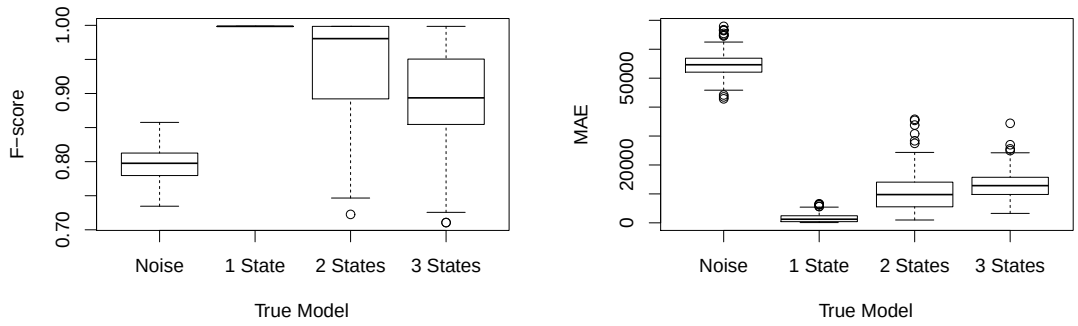


Figure 33: Fitted model test error on synthetic data.

certainly in terms of MAE, and in a statistically significant way (as indicated by interquartile ranges) in terms of F-1 scare. This confirms that our algorithm and mining methodology can identify the class of regular behavior that encompassed by delay-generating HMMs.

4.4.2 Edge coupling on real data

4.4.2.1 Evaluation Planes

Figure 34 shows bivariate histograms of the evaluation planes for each dataset, truncated above an MAE value of 6 hours. Recall that the point $(1, 0)$ represents perfect prediction, so we are mainly interested in the distribution of edges at the lower right corner of the plane. Our first observation is that edges do exist in this region, implying that networks contain edges whose occurrences can be predicted quite accurately. We examine the detailed structure of these predictable edges shortly, in Section 4.4.3.

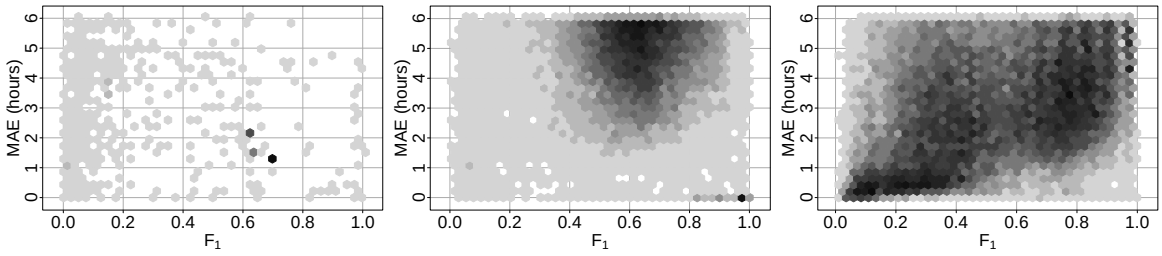
The CDR-C dataset shows two notable features: the dense semi-elliptical region of predictable interactions, and a small set of highly predictable edges with $F_1 > 0.8$ and very low MAE values. The former is possibly an artifact of the data collection bias. Since the CDR-C dataset sampled heavy users who made at least 3 calls a day, MAE values of 3-6 hours are not surprising due just to the frequency of calls. The latter feature is more interesting, and it is not clear from Figure 34 whether it is somehow characteristic of the dataset, or caused by, for example, automated systems.

The CDR-J dataset, unlike CDR-C, has no frequency-based sampling bias. It is interesting to note that in spite of the fact that CDR-J contains call records of *all* customers in a large geographical region, there does not seem to be a pronounced strip of highly predictable interactions like the one in CDR-C. Instead, the cluster around $F_1 \sim 0.8$ and MAE ~ 2.5 hours implies that there are a large number of edges that can be predicted with reasonable accuracy.

Finally, the Enron dataset appears to be sparse in terms of predictable edges, although it should be noted that the corpus is derived solely from the personal mailboxes of about 150 former Enron executives, and thus only a partial view of dynamics in the system. There are three prominent bins of edges, which we examine in more detail in Section 4.4.3.

4.4.2.2 Predictive Relationships

Recall from Section 4.3 that we model temporal dependencies between two types of edge pairs: autocorrelations, where an edge is both the trigger and the response, and correlations between pairs of edges that share a common vertex. Figure 35 shows a breakdown by type



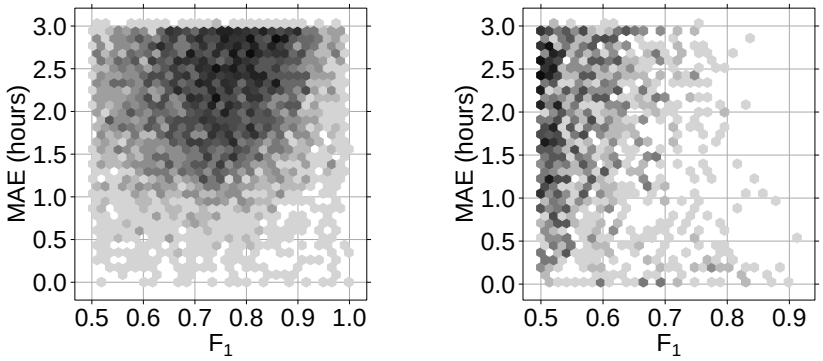
(a) Enron (b) CDR-C (c) CDR-J

Figure 34: Hexagonally binned bivariate histograms of the evaluation plane for each dataset, showing the predictability of mined edges with MAE less than 6 hours on test data. Darker bins represent more edges.

of edge pair of a window of the evaluation plane of the CDR-J dataset. In this window, autocorrelations of the form $(a, b) \rightarrow (a, b)$ are more prominent than pairwise correlations of the form $(a, b) \rightarrow (a, c)$, although a number of instances of the latter rank highly in terms of predictability. This suggests that structural correlations, in addition to temporal correlations, do exist in dynamic networks, and that these can be exploited for prediction purposes, in agreement with the qualitative and quantitative results of prior studies (Huang and Lin, 2009; Lahiri and Berger-Wolf, 2007).

4.4.2.3 Global performance and the η parameter

The η parameter, defined in Section 4.1.2, is the maximum allowable gain in MAE for a unit gain in F_1-score for one predicted timeline to be considered better than another. Since it is difficult to analytically determine an 'optimal' value for this parameter, we ran a series of experiments to assess the impact of the η parameter. In Table Table VII, we report the



(a) $(a, b) \rightarrow (a, b)$ (b) $(a, b) \rightarrow (a, c)$

Figure 35: Distribution of different types of edge pairs for the CDR-J dataset.

mean values of the F_1-score and MAE over all edge pairs for different values of η. Our findings indicate that the parameter has little impact on prediction performance, either quantitatively from Table Table VII, or qualitatively from scatterplots of the evaluation plane.

η (sec.)	F_1	Mean MAE	# States	Median # States
100	0.5430	3.365×10^5	2.36	2
1000	0.5446	3.366×10^5	2.42	2
10000	0.5440	3.365×10^5	2.45	2

TABLE VII: EFFECT OF VARYING η ON THE ENRON DATASET.

One possible reason for the insensitivity of the algorithm to η is that there are few edge pairs where different HMMs offer Pareto-optimal performance. Since the η parameter is only invoked during model selection when one HMM does not trivially dominate another, the insensitivity of performance to η implies that this scenario does not arise often enough to cause a significant difference in performance. We therefore use the value of $\eta = 1000$ for the remainder of our experiments, which corresponds to a trade-off of at most 15 minutes in MAE for a 10% gain in F_1-score when choosing the number of HMM states.

4.4.2.4 Fitted HMMs

Table Table VIII lists summary statistics of the learned models. Recall from our model description in Section 4.3 that the validation segment is used to choose the number of HMM states, and also between MAP and expectation-based prediction. It is interesting to note that while expectation-based prediction yields better results on the Enron dataset, predicting using the MAP method yields better performance in the CDR datasets, most notably in CDR-J. One possible explanation is that the CDR datasets contain interactions with highly clustered and separated delays; using an expected delay computed over multiple states could result in a predicted delay that lies between clusters, yielding poor performance relative to the MAP method. The reasons for this imbalance, particularly in CDR-J, warrant further study. We also note that the mean number of states is quite low, given the allowable range of $N = [1, 3]$, which indicates that our model selection strategy is successfully preventing overfitting to some degree.

Dataset	HMM states	Prediction method		
	Mean	MAP	Expect.	Equiv.
Enron	2.4 ± 1.07	30%	41%	29%
CDR-C	2.5 ± 1.09	40%	35%	25%
CDR-J	2.7 ± 1.16	56%	19%	25%

TABLE VIII: FITTED MODEL PARAMETERS.

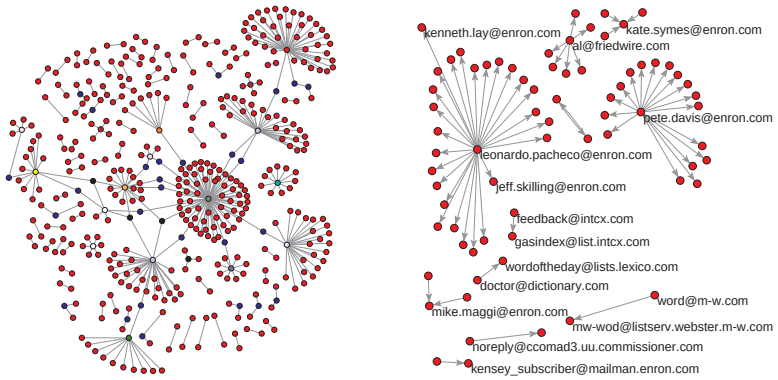
4.4.3 Applications of coupled edges

4.4.3.1 The Structure of Predictable Interactions

Figure 34 shows that there are a number of accurately predictable edges in all three networks. However, an important question is whether this is the result of genuinely interesting behavioral patterns and relationships of a wide range of individuals, or whether it is an artifact caused by, for example, an automated telemarketing or spam robot.

Figure 36 shows graph layouts of the highest ranked edges for the CDR-C and Enron datasets. In particular, Figure 36a shows the edges of CDR-C that are predictable with an average error of less than 10 minutes, and which comprise the dense strip in the bottom right corner of Figure 34b. There are several hub-and-spoke structures, which could represent, for example, call centers or businesses, but there are also a large number of isolated edges. It is difficult to determine what kind of specific behaviors are being discovered without further information about these nodes.

In the Enron dataset, however, we have at least partial information about the nodes. In particular, it is generally possible to tell whether an e-mail address corresponds to an



(a) CDR-C, $F_1 \geq 0.9$, MAE $\leq$ 10 minutes. Node colors correspond to degree: red (1), blue (2), black (3), etc.

(b) Enron, $F_1 \geq 0.6$, MAE $\leq$ 2 hours. Unlabeled nodes are also `@enron.com` addresses.

Figure 36: The structure of highly predictable edges.

automated program or to an Enron employee. Perhaps contrary to intuition, Figure 36b shows that the hub-and-spoke structures originate mainly from human accounts, and that several of the isolated edges are mailing lists.

Finally, the CDR-J graph layout (not pictured here) had 2,018 edges with $F_1 \geq 0.7$ and MAE $\leq$ 2 hours, the vast majority of which were comprised of degree-1 nodes, or isolated edges. There was a conspicuous dearth of hub-and-spoke structures, with just one such occurrence. This is surprising because CDR-J is a more complete dataset than CDR-C in terms of sample selection, and also more extensive in terms of geographical coverage, but does not seem to contain highly predictable hub-and-spoke structures. We can only

speculate that known cultural or economic differences between the regions are responsible for this discrepancy.

4.4.3.2 Community Identification

A node in a dynamic network may have many frequent contacts with a subset of its neighbors, but it is often more interesting to ask what *type* of relationship exists between nodes. For example, a person's most frequent e-mail contacts could be mailing lists and close associates. It might be difficult to tell these apart based solely on the frequency of communication, but it may be possible to infer more based on the predictability of interactions. The same applies to CDR datasets, where predictable phone calls suggest that the parties involved have a specific reason for maintaining their schedule.

Among the most predictable interactions in the Enron dataset are the e-mails sent from the `pete.davis@enron.com` e-mail address. Figure 37 shows the complete set of outlinks for this e-mail address, as well as the predictive performance for each edge. Based on performance in the test segment, there seem to be three types of edges associated with this address. The first is a set of addresses shown in the top-left quadrant for which no model could be learned, either because the edges were too infrequent, or if HMM training repeatedly resulted in singularity solutions.[1] While this may not be considered a specific community, the second and largest subset of addresses is shown in the bottom two quadrants, and is characterized by extremely high F_1-scores and low MAE values, implying a

[1]This is a well-known issue with the EM/Baum-Welch algorithm in general, see (Fraley and Raftery, 2007; Rabiner, 1989) for example.

Model			Performance	
a	b	μ_1, σ_1 (min.)	F_1	MAE (min.)
ka..symes	ev..metoyer	74.5 ± 36.6	0.631	293
mi..maggi	mi..nelson	1.6 ± 3.37	0.686	49
ka..symes	ke..thompson	72.3 ± 37.3	0.622	58.6
jo..griffith	al..villarreal	2.18 ± 1.9	0.615	82.2
al..villarreal	jo.griffith	6.77 ± 11.8	0.68	98.2

TABLE IX: HIGHEST RANKED $(a, b) \rightarrow (b, a)$ PAIRS IN THE ENRON DATASET. μ_1 AND σ_1 ARE THE MEAN AND VARIANCE OF ONLY THE MOST DOMINANT HMM STATE.

high degree of predictability. The final subset is shown in the top right quadrant and is characterized by much higher MAE values.

4.4.3.3 Relationship Classification

Table Table IX shows five of the highest-ranked edge pairs of the form $(a, b) \rightarrow (b, a)$ in the Enron dataset. In this case, the trigger is an e-mail, and its response is literally the reply. In all the cases listed, the HMM consisted of one clearly dominant state. It is interesting to note different styles of working relationships – the mean reply times vary between a few minutes to approximately an hour. Furthermore, finding symmetric pairs among the highest ranked interactions seems to be quite rare. This could be a result of fundamentally asymmetric relationships – for example, between an assistant and his superior – or a result of the fact that the Enron dataset offers only a partial view of all e-mail traffic.

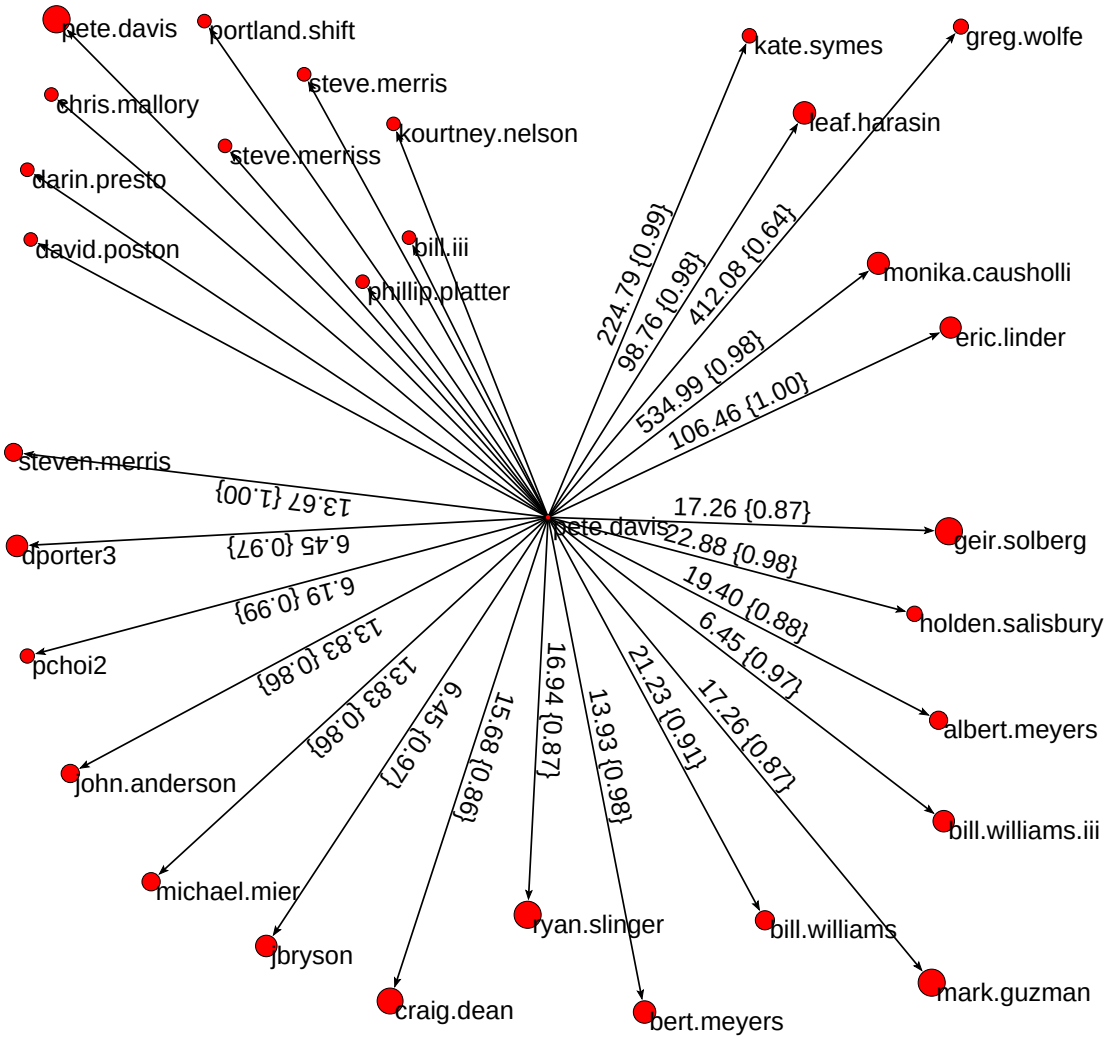


Figure 37: An egocentric e-mail network centered on `pete.davis@enron.com`, showing three possible communities. Edge labels are of the form $MAE(F_1)$ with MAE in minutes; vertex sizes are proportional to total number of e-mails sent.

4.5 Summary

We have described a novel framework and algorithm for mining edges in dynamic networks that exhibit temporally predictable interaction patterns, which we call *coupled edges*. Although there are a number of practical applications to our technique, we see its value as an exploratory data analysis tool. Its power stems from using a transparent prediction model (HMMs) with an easily interpretable structure that is grounded in the interaction dynamics of real systems. Furthermore, even if HMMs are the wrong model for the vast majority of edge dynamics, with apologies to the statistician George Box, there are certainly useful for the edges that we do mine. The following is a summary of our contributions in this chapter:

1. We formally defined the mining problem and an evaluation framework that deals with dynamic network data in continuous time.
2. We proposed an algorithm that predicts the future occurrence time of an edge, based on the prior occurrence times of either itself or another edge. Our algorithm models time delays between occurrences of edges, and is thus insensitive to timescales. It is also capable of representing and mining previously studied types of predictable patterns, such as partial periodic patterns (Han et al., 1999; Lahiri and Berger-Wolf, 2008), and temporal association rules (Oates et al., 1997).
3. We modeled temporal dependencies between edges that share a common vertex. This allows us to partially incorporate the structure of the network in an intuitive way, and is rendered tractable because of the skewed degree distribution observed in real

networks (Newman, 2003). Modeling all possible edge dependencies in real networks is intractable.

4. We demonstrated the applicability of our method on two industrial phone call datasets, and a publicly available e-mail dataset. Our experiments reveal that a large number of interactions are indeed predictable, sometimes to within a few hours, and that the structure of these interactions is quite complex. This suggests that although digitally recorded social networks are massive, there exists within them an embedded backbone of predictable connections that might have special meaning.

CHAPTER 5

CONCLUSION

In this thesis, we developed two new techniques for exploratory data analysis in dynamic networks, and exposed inherent measurement biases in a third. The focus has been on analyzing massive dynamic network datasets to understand facets of the physical systems that they represent. As researchers find more ways to record temporal information about network datasets, often on a continuous streaming basis, the need for exploratory tools that are agnostic to the source of data becomes apparent. These tools must make minimal assumptions about the underlying system, and be computationally and statistically tractable. The two new tools we develop fall into this category: the Fourier-like decomposition for dynamic networks in Chapter 2 assumes only the existence of locally periodic behavior, and the coupled edges we develop in Chapter 3 extend the notion of periodicity to other types of regular behavior. In this concluding chapter, we briefly touch upon some interesting open questions related to each of the three techniques we explore in this thesis.

In Chapter 2, we described an inherent measurement bias in a common method for measuring the graph-theoretic properties of a network over time, in the presence of even small amounts of missing temporal data. Since most network datasets lack a complete temporal history, this can be a significant problem when the observed (but not necessarily true) trends in graph theoretic properties are used for model building or algorithmic optimizations, for

example. We were able to show this bias through Monte Carlo simulations, and conjecture that there is no general way to correct it. This leads to quite a few open questions:

- *Are real dynamic network measurements using the growing network method accurate?* This is probably the most significant open question, and might require either looking for secondary markers of the bias caused by a missing temporal history, domain-specific information, or more advanced models of the dynamics of networks that take both repeated discovery of edges and vertices, as well as the addition of new edges and vertices, into account.
- *Are there network properties that are not affected by the missing data bias?* Almost all the network properties analyzed in the literature exhibit some form of bias with a small amount of missing data. However, spectral properties seem more resilient to these biases. An analytical exploration of which network measures converge to their true trends would be extremely valuable.
- *How else can we measure network properties?* Given that there are serious issues with missing data in real networks, can alternative sampling schemes be developed to give a more accurate picture of the evolution of network properties? For example, to measure trends in the average shortest path length in a network over time, would a sampling strategy that only considers shortest paths between specific pairs of nodes yield a more accurate picture of network evolution than measuring the entire network?

In Chapter 3, we proposed a method for finding both the important periodicities as well as periodic patterns in a dynamic network. In particular, our problem formulation is the first

one that aims to describe periodic patterns without redundancy. As a result, we proved that the enumeration and counting variants of our mining problem are in the complexity class P, unlike several other periodic pattern mining formulations that are either #P-complete for counting, or NP-hard for enumeration. We described an efficient algorithm to mine these patterns and periodicities, and found a number of very intuitive periodicities in real datasets: e-mail traffic has principal periodicities of 1 day and 7 days, web server access logs have their highest peaks at 1 day, 7 days, and then 2 and 3 days in decreasing order. Unsurprisingly, repeated patterns of celebrity sightings occur most frequently approximately every 365 days.

Given the success of our algorithm at detecting periodicities in a number of diverse datasets, there are a number of algorithmic and theoretical extensions that can be made:

- *How can approximate periodicity be quantified?* We proposed a heuristic mechanism to analyze approximate periodicities, but did not analyze it theoretically. A strict combinatorial definition of approximate periodicity is likely to be difficult, so statistical approaches might present a better alternative.
- *Can we develop optimal, approximate, parallel or streaming versions of the mining algorithm?* These are natural extensions of the mining algorithm we proposed.
- *Can we detect communities by finding periodic semi-patterns?* A semi-pattern is one which does not appear in its entirety at each timestep. For example, a group of individuals might meet on a strictly periodic schedule, but not all members of the group will attend every meeting. Can these 'visitors' be detected by association with

the members? One way to achieve this would be to augment networks with pseudo-nodes for periodic patterns, with edges to all vertices that are part of that pattern, and then apply static graph link prediction to infer which other nodes are likely to be a part of the pattern.

Finally, in Chapter 4, we described the mining of temporally coupled edges, where the occurrence of an edge at a particular time predicts the occurrence of another edge at a later time. We used edge dependencies defined by the line graph transformation of a graph, the first-order differenced time series of edge occurrence times, and a novel Hidden Markov Model formulation for this task. We were able to show that a number of edges in real networks are predictable to a high degree, and that the network structure of these edges is non-trivial, suggesting the existence of a 'predictable core' sub-network. The most prominent open question relates to increasing the accuracy of the predictive models used.

- *Can predictive models for coupled edges be made more accurate?* The HMM model we describe is a step in this direction, and extensions of HMMs such as Hidden Semi-Markov Models (Salfner and Malek, 2007), or even completely different predictive models, could boost the number of coupled edges that can be found.
- *What are other applications for coupled edges?* In Chapter 4, we demonstrated a number of practical applications for mining coupled edges. It would be interesting to see the extent to which they can be found in other datasets, and what temporally predictive edges correspond to.

- *What is the network structure of coupled edges?* The network structure of these edges, in aggregate, would also be illuminating should it have a non-trivial structure or span a large portion of the vertices in the network. We conjecture, based on the limited experiments with our datasets, that such a backbone of predictable interactions exists in information networks; what does this sub-network correspond to in reality, and how effectively does it connect the network?

There are many other interesting open questions, and the increasing quantity and diversity of network data makes robust analytical tools indispensable. This thesis is a small step in the development of a standard suite of such tools.

CITED LITERATURE

Acar, E., Dunlavy, D., and Kolda, T.: Link prediction on evolving data using matrix and tensor factorizations. In IEEE Intl. Conf. on Data Mining Wkshps., pages 262–269. IEEE, 2009.

Achlioptas, D., Clauset, A., Kempe, D., and Moore, C.: On the bias of traceroute sampling: or, power-law degree distributions in regular graphs. J.ACM, 56(4):1–28, 2009.

Adar, E. and Adamic, L.: Tracking information epidemics in blogspace. In Proc. of the 2005 IEEE/WIC/ACM Intl. Conf. on Web Intelligence, pages 207–214. IEEE Computer Society, 2005.

Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In Proc. of the 20th Intl. Conf. on Very Large Data Bases, pages 487–499, San Francisco, CA, 1994. Morgan Kaufmann Publishers Inc.

Agrawal, R. and Srikant, R.: Mining sequential patterns. In Proc. of the 11th Intl. Conf. on Data Engineering, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

Ahn, Y.-Y., Han, S., Kwak, H., Moon, S., and Jeong, H.: Analysis of topological characteristics of huge online social networking services. In Proc. of the 16th Intl. Conf. on World Wide Web, pages 835–844, New York, NY, USA, 2007. ACM.

Akoglu, L. and Faloutsos, C.: RTG: a recursive realistic graph generator using random typing. Machine Learning and Knowledge Discovery in Databases, pages 13–28, 2009.

Akoglu, L., McGlohon, M., and Faloutsos, C.: RTM: Laws and a recursive generator for weighted time-evolving graphs. In Proc. of the 8th IEEE Intl. Conf. on Data Mining, pages 701–706. IEEE, 2008.

Albert, R. and Barabási, A.: Statistical mechanics of complex networks. Reviews of modern physics, 74(1):47–97, 2002.

Albert, R., Jeong, H., and Barabási, A. L.: Diameter of the World Wide Web. Nature, 401, September 1999.

Almendral, J. and Díaz-Guilera, A.: Dynamical and spectral properties of complex networks. New Journal of Physics, 9:187, 2007.

Andersen, D., Feamster, N., Bauer, S., and Balakrishnan, H.: Topology inference from BGP routing dynamics. In Proc. of the 2nd ACM SIGCOMM Wkshp. on Internet measurment, pages 243–248. ACM, 2002.

Aspnes, J., Rustagi, N., and Saia, J.: Worm versus alert: Who wins in a battle for control of a large-scale network? Principles of Distributed Systems, pages 443–456, 2007.

Banerjee, A. and Jost, J.: Spectral characterization of network structures and dynamics. Dynamics On and Of Complex Networks, pages 117–132, 2009.

Barabási, A. L. and Albert, R.: Emergence of scaling in random networks. Science, 286(5439):509, 1999.

Barabási, A. L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A., and Vicsek, T.: Evolution of the social network of scientific collaborations. Physica A: Statistical Mechanics and its Applications, 311(3–4):590–614, 2002.

Barrat, A., Barthelemy, M., Pastor-Satorras, R., and Vespignani, A.: The architecture of complex weighted networks. Proc. of the National Academy of Sciences of the United States of America, 101(11):3747, 2004.

Barthelemy, M., Barrat, A., Pastor-Satorras, R., and Vespignani, A.: Characterization and modeling of weighted networks. Physica A: Statistical Mechanics and its Applications, 346(1-2):34–43, 2005.

Beyene, Y., Faloutsos, M., Chau, D., and Faloutsos, C.: The eBay graph: How do online auction users interact? In Proc. of the IEEE Conf. on Computer Communications Wkshps., pages 1–6, 2008.

Biggs, N.: Algebraic graph theory. Cambridge Univ Pr, 1993.

Bilgic, M. and Getoor, L.: Effective label acquisition for collective classification. In Proc. of the 14th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 43–51. ACM, 2008.

Bilke, S. and Peterson, C.: Topological properties of citation and metabolic networks. Physical Review E, 64(3):036106, 2001.

Bluetooth SIG, Inc.: Bluetooth: Technical comparison, Feb. 2009. http://www.bluetooth.com/Bluetooth/Technology/Works/Compare/Technical/.

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.: Complex networks: Structure and dynamics. Physics Reports, 424(4-5):175–308, 2006.

Bollobás, B. and Riordan, O.: The diameter of a scale-free random graph. Combinatorica, 24(1):5–34, 2004.

Bollobás, B.: Modern graph theory. Springer Verlag, 1998.

Bollobás, B., Riordan, O., Spencer, J., and Tusnády, G.: The degree sequence of a scale-free random graph process. Random Structures and Algorithms, 18(3):279–290, 2001.

Bonacich, P. and Lloyd, P.: Eigenvector-like measures of centrality for asymmetric relations. Social Networks, 23(3):191–201, 2001.

Bonato, A., Hadi, N., Horn, P., Prałat, P., and Wang, C.: A dynamic model for on-line social networks. Algorithms and Models for the Web-Graph, pages 127–142, 2009.

Boros, E., Gurvich, V., Khachiyan, L., and Makino, K.: On the complexity of generating maximal frequent and minimal infrequent sets. In Proc. of the 19th Annual Symposium on Theoretical Aspects of Computer Science, pages 133–141, London, UK, 2002. Springer-Verlag.

Brin, S., Motwani, R., Ullman, J., and Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In Proc. of the 1997 ACM SIGMOD Intl. Conf. on Management of data, pages 255–264. ACM Press New York, NY, USA, 1997.

Brin, S. and Page, L.: The anatomy of a large-scale hypertextual web search engine. In Proc. of the seventh Intl. Conf. on World Wide Web 7, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

Bringmann, B. and Zimmermann, A.: One in a million: picking the right patterns. Knowledge and Information Systems, 18(1):61–81, 2009.

Broder, A.: How hard is it to marry at random? (On the approximation of the permanent). In Proc. of the eighteenth annual ACM symposium on Theory of computing, pages 50–58. ACM, 1986.

Bunke, H.: Recent developments in graph matching. In Proc. of the 15th Intl. Conf. on Pattern Recognition, page 2117. Published by the IEEE Computer Society, 2000.

Bunke, H.: Graph-based tools for data mining and machine learning. Lecture Notes in Computer Science, pages 7–19, 2003.

Bunke, H., Dickinson, P., Irniger, C., and Kraetzl, M.: Analysis of time series of graphs: Prediction of node presence by means of decision tree learning. In Proc. of the 4th Intl. Conf. on Machine Learning and Data Mining in Pattern Recognition, volume 3587, pages 366–375. Springer, 2005.

Buriol, L., Castillo, C., Donato, D., Leonardi, S., and Millozzi, S.: Temporal analysis of the Wikigraph. In Proc. Web. Intell. 2006, pages 45–51, 2006.

Burt, R.: Decay functions. Social Networks, 22(1):1–28, 2000.

Callaway, D., Hopcroft, J., Kleinberg, J., Newman, M., and Strogatz, S.: Are randomly grown graphs really random? Physical Review E, 64(4):41902, 2001.

Carpineto, C. and Romano, G.: Concept Data Analysis: Theory and Applications. John Wiley & Sons, 2004.

Chakrabarti, D., Faloutsos, C., and McGlohon, M.: Graph mining: Laws and generators. Managing and Mining Graph Data, pages 69–123, 2010.

Chakrabarti, D. and Faloutsos, C.: Graph mining: Laws, generators, and algorithms. ACM Comput. Surv., 38(1):2, 2006.

Chaovalitwongse, W. A., Fan, Y.-J., and Sachdeo, R. C.: Support feature machine for classification of abnormal brain activity. In Proc. of the 13th ACM SIGKDD Intl. Conf. on knowl. discovery and data mining, pages 113–122, New York, NY, USA, 2007. ACM.

Chapanond, A., Krishnamoorthy, M. S., and Yener, B.: Graph theoretic and spectral analysis of Enron email data. Comput. Math. Organ. Theory, 11(3):265–281, 2005.

Chatfield, C.: The analysis of time series: an introduction. CRC press, 2004.

Chen, S. and Liu, J.: Statistical applications of the poisson-binomial and conditional bernoulli distributions. Statistica Sinica, 7:875–892, 1997.

Chen, Y., Lim, K., Katz, R., and Overton, C.: On the stability of network distance estimation. ACM SIGMETRICS Performance Evaluation Review, 30(2):21–30, 2002.

Cheng, J., Ke, Y., and Ng, W.: A survey on algorithms for mining frequent itemsets over data streams. Knowledge and Information Systems, 16(1):1–27, 2008.

Chung, F.: Spectral graph theory (CBMS Regional Conf. Series in Mathematics, No. 92). American Mathematical Society, 3:8, 1997.

Chung, F., Faber, V., and Manteuffel, T.: An upper bound on the diameter of a graph from eigenvalues associated with its Laplacian. SIAM J. Discrete Math., 7:443, 1994.

Clauset, A. and Eagle, N.: Persistence and periodicity in a dynamic proximity network. DIMACS/DyDAn Wkshp. on Comput. Meth. for Dynamic Interaction Networks, 2007.

Clauset, A., Shalizi, C. R., and Newman, M. E. J.: Power-law distributions in empirical data. SIAM Review, 51:661, 2009.

Costenbader, E. and Valente, T.: The stability of centrality measures when networks are sampled. Social networks, 25(4):283–307, 2003.

De Choudhury, M., Mason, W. A., Hofman, J. M., and Watts, D. J.: Inferring relevant social networks from interpersonal communication. In Proc. of the 19th Intl. Conf. on World Wide Web, WWW '10, pages 301–310, New York, NY, USA, 2010. ACM.

Delvenne, J., Yaliraki, S., and Barahona, M.: Stability of graph communities across time scales. Proc. of the National Academy of Sciences, 107(29):12755, 2010.

Dhamdhere, A. and Dovrolis, C.: Ten years in the evolution of the internet ecosystem. In Proc. of the 8th ACM SIGCOMM Conf. on Internet measurement, pages 183–196. ACM New York, NY, USA, 2008.

Dickinson, P. J., Bunke, H., Dadej, A., and Kraetzl, M.: On Graphs with Unique Node Labels, volume 2726 of Lecture Notes in Computer Science, pages 409–437. Springer Berlin, 2003.

Diesner, J. and Carley, K. M.: Exploration of Communication Networks from the Enron Email Corpus. In Proc. of the 2005 SIAM Wkshp. on Link Analysis, Counterterrorism and Security, pages 3–14, 2005.

Dietterich, T. G.: Machine learning for sequential data: A review. In Proc. of the Joint IAPR Intl. Wkshp. on Structural, Syntactic, and Statistical Pattern Recognition, pages 15–30, London, UK, 2002. Springer-Verlag.

Domeniconi, C., Perng, C.-S., Vilalta, R., and Ma, S.: A classification approach for prediction of target events in temporal sequences. In Proc. of the 6th European Conf. on Principles of Data Mining and Knowl. Disc., pages 125–137, London, UK, 2002. Springer-Verlag.

Domingos, P. and Richardson, M.: Mining the network value of customers. In Proc. 7th ACM SIGKDD, pages 57–66, 2001.

Dong, Z., Wu, W., Ma, X., Xie, K., and Jin, F.: Mining the structure and evolution of the airport network of China over the past twenty years. In Proc. of the 5th Intl. Conf. on Advanced Data Mining and Applications, page 115. Springer, 2009.

Dorogovtsev, S., Goltsev, A., and Mendes, J.: Critical phenomena in complex networks. Reviews of Modern Physics, 80(4):1275–1335, 2008.

Du, N., Wang, H., and Faloutsos, C.: Analysis of large multi-modal social networks: patterns and a generator. Machine Learning and Knowledge Discovery in Databases, pages 393–408, 2010.

Eagle, N. and Pentland, A.: Reality mining: sensing complex social systems. Personal and Ubiquitous Computing, 10(4):255–268, 2006.

Ebel, H., Mielsch, L.-I., and Bornholdt, S.: Scale-free topology of e-mail networks. Physical Review E, 66(3):035103–+, Sep. 2002.

Efraimidis, P. S. and Spirakis, P. G.: Weighted random sampling with a reservoir. Information Processing Letters, 97:181–185, March 2006.

Elfeky, M. G., Aref, W. G., and Elmagarmid, A. K.: Periodicity detection in time series databases. IEEE Trans. on Knowledge and Data Engineering, 17(7):875–887, 2005.

Elfeky, M. G., Aref, W. G., and Elmagarmid, A. K.: WARP: Time warping for periodicity detection. In Proc. of the Fifth IEEE Intl. Conf. on Data Mining, pages 138–145, Washington, DC, USA, 2005. IEEE Computer Society.

Elmacioglu, E. and Lee, D.: On six degrees of separation in DBLP-DB and more. ACM SIGMOD Record, 34(2):33–40, 2005.

Erdös, P. and Rényi, A.: On random graphs. Publicationes Mathematicae, 6(26):290–297, 1959.

Erdős, P. and Rényi, A.: On the evolution of random graphs. Publications of the Mathematical Institute of the Hungarian Academy of Sciences, 5:17–61, 1960.

Eubank, S., Kumar, V. S. A., Marathe, M. V., Srinivasan, A., and Wang, N.: Structural and algorithmic aspects of massive social networks. In Proc. of the 15th. annual ACM-SIAM symposium on discrete algorithms, pages 718–727, 2004.

Faber, N., Bro, R., and Hopke, P.: Recent developments in CANDECOMP/PARAFAC algorithms: a critical review. Chemometrics and Intelligent Laboratory Systems, 65(1):119–137, 2003.

Faloutsos, M., Faloutsos, P., and Faloutsos, C.: On power-law relationships of the internet topology. In Proc. of the Conf. on Applications, technologies, architectures, and protocols for computer communication, pages 251–262, New York, NY, 1999. ACM.

Farkas, I., Derenyi, I., Barabasi, A., and Vicsek, T.: Spectra of "real-world" graphs: Beyond the semicircle law. Physical Review E, 64(2):26704, 2001.

Fawcett, T.: ROC graphs: Notes and practical considerations for researchers. Machine Learning, 31, 2004.

Fawcett, T. and Provost, F.: Activity monitoring: noticing interesting changes in behavior. In Proc. of the 5th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 53–62, New York, NY, USA, 1999. ACM.

Fischhoff, I. R., Sundaresan, S. R., Cordingley, J., Larkin, H. M., Sellier, M.-J., and Rubenstein, D. I.: Social relationships and reproductive state influence leadership roles in movements of Plains zebra, *Equus burchellii*. Animal Behaviour, 73(5):825–831, 2007.

Fraley, C. and Raftery, A. E.: Bayesian regularization for normal mixture estimation and model-based clustering. J. Classif., 24(2):155–181, 2007.

Freeman, L.: Finding social groups: A meta-analysis of the southern women data. In Dynamic Social Network Modeling and Analysis: workshop summary and papers, pages 39–77, 2003.

Frias-Martinez, E. and Karamcheti, V.: A prediction model for user access sequences. In Proc. WebKDD Wkshp., 2002.

Frias-Martinez, E. and Karamcheti, V.: Reduction of user perceived latency for a dynamic and personalized site using web-mining techniques. In Proc. WebKDD Wkshp., 2003.

Galil, Z.: Efficient algorithms for finding maximum matching in graphs. ACM Comput. Surv., 18(1):23–38, 1986.

Garofalakis, M., Rastogi, R., and Shim, K.: SPIRIT: Sequential pattern mining with regular expression constraints. In Proc. of the Intl. Conf. on very large data bases, pages 223–234, 1999.

Geng, L. and Hamilton, H.: Interestingness measures for data mining: A survey. ACM Computing Surveys, 38(3):9, 2006.

Getoor, L. and Taskar, B.: Introduction to Statistical Relational Learning. MIT Press, 2007.

Getoor, L., Friedman, N., Koller, D., and Taskar, B.: Learning probabilistic models of link structure. Journal of Machine Learning Research, 3:679–707, 2003.

Ghani, A., Donnelly, C., and Garnett, G.: Sampling biases and missing data in explorations of sexual partner networks for the spread of sexually transmitted diseases. Statistics in medicine, 17(18):2079–2097, 1998.

Goldstein, M., Morris, S., and Yen, G.: Problems with fitting to the power-law distribution. The European Physical Journal B, 41(2):255–258, 2004.

Good, P. and Wang, R.: Permutation, parametric and bootstrap tests of hypotheses. Springer New York, 2005.

Hall, B. H., Jaffe, A. B., and Trajtenberg, M.: The NBER patent citations data file: Lessons, insights and methodological tools. NBER Working Papers 8494, National Bureau of Economic Research, Inc., 2001.

Han, J., Cheng, H., Xin, D., and Yan, X.: Frequent pattern mining: Current status and future directions. Data Mining and Knowledge Discovery, 15(1):55–86, 2007.

Han, J., Yin, Y., and Dong, G.: Efficient mining of partial periodic patterns in time series database. In Proc. of the 15th Intl. Conf. on Data Engineering, pages 106–115, Los Alamitos, CA, 1999. IEEE Computer Society.

Hanhijärvi, S., Garriga, G. C., and Puolamäki, K.: Randomization techniques for graphs. In Proc. of the Ninth SIAM Intl. Conf. on Data Mining, pages 780–791. SIAM, 2009.

Harms, S. K. and Deogun, J. S.: Sequential association rule mining with time lags. Journal of Intelligent Information Systems, 22(1):7–22, 2004.

Hastie, T., Tibshirani, R., and Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2001.

Hill, R. and Dunbar, R.: Social network size in humans. Human Nature, 14(1):53–72, 2003.

Hoff, P. D., Raftery, A. E., and Handcock, M. S.: Latent space approaches to social network analysis. Journal of the American Statistical Association, 97(460):1090–1098, 2002.

Holme, P., Edling, C., and Liljeros, F.: Structure and time evolution of an internet dating community. Social Networks, 26(2):155–174, 2004.

Hu, H. and Wang, X.: Evolution of a large online social network. Physics Letters A, 373:1105–1110, March 2009.

Huang, J., Zhuang, Z., Li, J., and Giles, C. L.: Collaboration over time: characterizing and modeling network evolution. In Proc. WSDM '08, pages 107–116, 2008.

Huang, K.-Y. and Chang, C.-H.: SMCA: A general model for mining asynchronous periodic patterns in temporal databases. IEEE Transactions on Knowledge and Data Engineering, 17(6):774–785, 2005.

Huang, Z. and Lin, D. K. J.: The time-series link prediction problem with applications in communication surveillance. INFORMS Journal on Computing, 21(2):286–303, 2009.

Inokuchi, A., Washio, T., and Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In Proc. of the 4th European Conf. on Principles of Data Mining and Knowledge Discovery, pages 13–23, 2000.

Jensen, D. and Neville, J.: Data Mining in Social Networks. In Dynamic Social Network Modeling and Analysis: Wkshp. Summary and Papers. National Academy Press, 2003.

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., and Rubenstein, D. I.: Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. ACM SIGPLAN Notices, 37(10):96–107, 2002.

Kannadiga, P., Zulkernine, M., and Haque, A.: E-NIPS: An event-based network intrusion prediction system. LECTURE NOTES IN COMPUTER SCIENCE, 4779:37, 2007.

Kashima, H. and Abe, N.: A parameterized probabilistic model of network evolution for supervised link prediction. In Proc. of the Sixth IEEE Intl. Conf. on Data Mining, pages 340–349, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

Kempe, D., Kleinberg, J., and Tardos, É.: Maximizing the spread of influence through a social network. In Proc. 9th ACM SIGKDD, pages 137–146, 2003.

Kleinberg, J. M.: Navigation in a small world. Nature, 406(6798):845, 2000.

Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A.: The web as a graph: Measurements, models, and methods. In Proc. of the 5th annual Intl. Conf. on Computing and combinatorics, pages 1–17. Springer-Verlag, 1999.

Kossinets, G.: Effects of missing data in social networks. Social Networks, 28(3):247–268, 2006.

Kossinets, G. and Watts, D. J.: Empirical analysis of an evolving social network. Science, 311(5757):88–90, January 2006.

Krishnamurthy, B., Gill, P., and Arlitt, M.: A few chirps about twitter. In Proc. of the first workshop on Online social networks, WOSP '08, pages 19–24, New York, NY, USA, 2008. ACM.

Kumar, A., Xu, J., and Zegura, E.: Efficient and scalable query routing for unstructured peer-to-peer networks. In INFOCOM 2005: Proc. of the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies, volume 2, pages 1162 – 1173 vol. 2, march 2005.

Kumar, R., Novak, J., and Tomkins, A.: Structure and evolution of online social networks. In Proc. of the 12th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 611–617. ACM New York, NY, USA, 2006.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A., and Upfal, E.: The Web as a graph. In Proc. of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 1–10. ACM, 2000.

Kumar, R., Novak, J., and Tomkins, A.: Structure and evolution of online social networks. In Proc. ACM SIGKDD '06, pages 611–617, 2006.

Kuramochi, M. and Karypis, G.: Frequent subgraph discovery. In Proc. of the 2001 IEEE Intl. Conf. on Data Mining, pages 313–320, 2001.

Lahiri, M. and Berger-Wolf, T. Y.: Structure prediction in temporal networks using frequent subgraphs. In Proc. of IEEE Symposium on Computational Intelligence and Data Mining, pages 35–42, 2007.

Lahiri, M. and Berger-Wolf, T.: Periodic subgraph mining in dynamic networks. Knowledge and Information Systems, 24(3):467–497, 2010.

Lahiri, M., Tantipathananandh, C., Warungu, R., Rubenstein, D., and Berger-Wolf, T.: Biometric animal databases from field photographs: Identification of individual zebra in the wild. In Proc. of the ACM Intl. Conf. on Multimedia Retr.. ACM Press, 2011.

Lahiri, M. and Berger-Wolf, T. Y.: Mining periodic behavior in dynamic social networks. In Proc. of the IEEE Intl. Conf. on Data Mining, pages 373–382, 2008.

Langville, A., Meyer, C., and Fernández, P.: Google's PageRank and beyond: the science of search engine rankings. The Mathematical Intelligencer, 30(1):68–69, 2008.

Latapy, M. and Magnien, C.: Complex network measurements: Estimating the relevance of observed properties. In Proc. IEEE INFOCOM '08, pages 1660–1668, 2008.

Latapy, M. and Magnien, C.: Measuring fundamental properties of real-world complex networks. CoRR, abs/cs/0609115, 2006.

Latora, V. and Marchiori, M.: Efficient behavior of small-world networks. Phys. Rev. Lett., 87(19):198701, Oct 2001.

Laxman, S., Sastry, P., and Unnikrishnan, K.: Discovering frequent episodes and learning hidden markov models: A formal connection. IEEE Transactions on Knowledge and Data Engineering, 17(11):1505–1517, 2005.

Laxman, S., Tankasali, V., and White, R. W.: Stream prediction using a generative model based on frequent episodes in event sequences. In KDD '08: Proceeding of the 14th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 453–461, New York, NY, USA, 2008. ACM.

Lehot, P. G. H.: An optimal algorithm to detect a line graph and output its root graph. Journal of the ACM, 21(4):569–575, 1974.

Lesh, N., Zaki, M. J., and Ogihara, M.: Mining features for sequence classification. In KDD '99: Proc. of the fifth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 342–346, New York, NY, USA, 1999. ACM.

Leskovec, J. and Faloutsos, C.: Sampling from large graphs. In Proc. of the 12th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, page 636. ACM, 2006.

Leskovec, J., Kleinberg, J., and Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Trans. on Knowledge Discovery from Data, 1(1):2, 2007.

Leskovec, J. and Horvitz, E.: Planetary-scale views on a large instant-messaging network. In Proceeding of the 17th Intl. Conf. on World Wide Web, pages 915–924, New York, NY, USA, 2008. ACM.

Leskovec, J., Kleinberg, J. M., and Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In Proc. of the 11th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining, pages 177–187, 2005.

Ley, M.: The DBLP computer science bibliography: Evolution, research issues, perspectives. In SPIRE 2002: Proc. of the 9th Intl. Symposium on String Processing and Information Retrieval, pages 1–10, London, UK, 2002. Springer-Verlag.

Liang, Y., Zhang, Y., Sivasubramaniam, A., Jette, M., and Sahoo, R.: Bluegene/l failure analysis and prediction models. Intl. Conf. on Dependable Systems and Networks, 0:425–434, 2006.

Liben-Nowell, D. and Kleinberg, J. M.: The link prediction problem for social networks. In Proc. of the twelfth Intl. Conf. on Information and knowledge management, pages 556–559, New York, NY, USA, 2003. ACM.

Liu, B., Hsu, W., and Ma, Y.: Integrating classification and association rule mining. In Proc. of the 4rd Intl. Conf. Knowledge Discovery and Data Mining (KDD-98), pages 80–86. AAAI Press, 1998.

Ma, S. and Hellerstein, J. L.: Mining partially periodic event patterns with unknown periods. In Proc. of the 17th Intl. Conf. on Data Engineering, pages 205–214, Washington, DC, USA, 2001. IEEE Computer Society.

Malmgren, R. D., Hofman, J. M., Amaral, L. A., and Watts, D. J.: Characterizing individual communication patterns. In Proc. of the 15th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining, pages 607–616, 2009.

Mannila, H., Toivonen, H., and Inkeri Verkamo, A.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1(3):259–289, 1997.

Mannila, H., Toivonen, H., and Verkamo, A. I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1(3):259–289, 1997.

Menezes, G. V., Ziviani, N., Laender, A. H., and Almeida, V.: A geographical analysis of knowledge production in computer science. WWW 09: Proc. of the 18th Intl. Conf. on World Wide Web, page 1041, 2009.

Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P., and Bhattacharjee, B.: Measurement and analysis of online social networks. In Proc. of the 7th ACM SIGCOMM Conf. on Internet measurement, pages 29–42, 2007.

Moody, J.: The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. American Sociological Review, 69(2):213, 2004.

Moreno, J. L.: Who Shall Survive?. Washington, D.C., Nervous and Mental Disease Publishing Company, 1934.

Murata, T. and Moriyasu, S.: Link prediction of social networks based on weighted proximity measures. In Proc. of the IEEE/WIC/ACM Intl. Conf. on Web Intelligence, pages 85–88, Washington, DC, USA, 2007. IEEE Computer Society.

Nanavati, A. A., Gurumurthy, S., Das, G., Chakraborty, D., Dasgupta, K., Mukherjea, S., and Joshi, A.: On the structural properties of massive telecom call graphs: findings and implications. In Proc. of the 15th ACM Intl. Conf. on Information and knowledge management, pages 435–444, New York, NY, USA, 2006. ACM.

Nascimento, M., Sander, J., and Pound, J.: Analysis of SIGMOD's co-authorship graph. ACM SIGMOD Record, 32(3):8–10, 2003.

Naumov, V., Baumann, R., and Gross, T.: An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In MobiHoc '06: Proc. of the 7th ACM Intl. symposium on Mobile ad hoc networking and computing, pages 108–119, New York, NY, USA, 2006. ACM.

Nerur, S., Sikora, R., Mangalaraj, G., and Balijepally, V.: Assessing the relative influence of journals in a citation network. Commun. ACM, 48:71–74, November 2005.

Newman, M. E. J.: Clustering and preferential attachment in growing networks. Physical Review E, 64(2):25102, 2001.

Newman, M. E. J.: From the cover: The structure of scientific collaboration networks. Proc. of the National Academy of Science, 98:404–409, January 2001.

Newman, M. E. J.: The structure and function of complex networks. SIAM Review, 45(2):167–256, 2003.

Newman, M. E. J. and Leicht, E. A.: Mixture models and exploratory analysis in networks. Proc. of the National Academy of Sciences, 104(23):9564, 2007.

Newman, M.: Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. Physical review E, 64(1):16132, 2001.

Newman, M.: Analysis of weighted networks. Physical Review E, 70(5):56131, 2004.

Oates, T., Schmill, M., Jensen, D., and Cohen, P.: A family of algorithms for finding temporal structure in data. In 6th Intl. Wkshp. on A.I. and Statistics, 1997.

Oates, T. and Cohen, P. R.: Searching for structure in multiple streams of data. In In Proc. of the Thirteenth Intl. Conf. on Machine Learning, pages 346–354. Morgan Kaufmann, 1996.

O'Madadhain, J., Hutchins, J., and Smyth, P.: Prediction and ranking algorithms for event-based network data. ACM SIGKDD Explorations Newsletter, 7(2):23–30, 2005.

Özden, B., Ramaswamy, S., and Silberschatz, A.: Cyclic association rules. In Proc. of the Fourteenth Intl. Conf. on Data Engineering, pages 412–421, Washington, DC, USA, 1998. IEEE Computer Society.

Pallis, G., Katsaros, D., Dikaiakos, M., Loulloudes, N., and Tassiulas, L.: On the structure and evolution of vehicular networks. In IEEE Intl. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems, pages 1–10, 2009.

Park, S., Pennock, D., and Giles, C.: Comparing static and dynamic measurements and models of the Internet's AS topology. In Proc. IEEE INFOCOM, volume 3, pages 1616–1627, 2004.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Information Systems, 24(1):25–46, 1999.

Pedarsani, P., Figueiredo, D. R., and Grossglauser, M.: Densification arising from sampling fixed graphs. In Proc. of the ACM SIGMETRICS Intl. Conf. on measurement and modeling of computer systems, pages 205–216. ACM, 2008.

Pei, J. and Han, J.: Can we push more constraints into frequent pattern mining? In Proc. of the sixth ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 350–354. ACM New York, NY, USA, 2000.

Pei, J., Han, J., and Wang, W.: Mining sequential patterns with constraints in large databases. In Proc. of the eleventh Intl. Conf. on Information and knowledge management, pages 18–25. ACM New York, NY, USA, 2002.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C.: Mining sequential patterns by pattern-growth: The PrefixSpan approach. IEEE Transactions on Knowledge and Data Engineering, 16(11):1424–1440, 2004.

Perra, N. and Fortunato, S.: Spectral centrality measures in complex networks. Physical Review E, 78(3):36107, 2008.

Phithakkitnukoon, S. and Dantu, R.: Predicting calls–new service for an intelligent phone. In Real-Time Mobile Multimedia Services, volume 4787 of LNCS, pages 26–37. Springer, 2007.

Pincombe, B.: Anomaly detection in time series of graphs using ARMA processes. ASOR Bulletin, 24(4):2–10, 2005.

Popescul, A. and Ungar, L. H.: Statistical relational learning for link prediction. In IJCAI03 Wkshp. on Learning Statistical Models from Relational Data, 2003.

Rabiner, L. R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proc. of the IEEE, 77:257–286, 1989.

Raftery, A.: Bayes factors and BIC. Sociological Methods & Research, 27(3):411–417, 1999.

Raney, B., Voellmy, A., Cetin, N., Vrtic, M., and Nagel, K.: Towards a microscopic traffic simulation of all of Switzerland. In ICCS '02: Proc. of the Intl. Conf. on Computational Science-Part I, pages 371–380, London, UK, 2002. Springer-Verlag.

Ryan, T.: Modern regression methods. Wiley-Interscience, 2008.

Salfner, F. and Malek, M.: Using hidden semi-markov models for effective online failure prediction. pages 161–174, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

Sarkar, P. and Moore, A. W.: Dynamic social network analysis using latent space models. ACM SIGKDD Explorations Newsletter, 7(2):31–40, 2005.

Seber, G. A. and Lee, A. J.: Linear Regression Analysis. Wiley-Interscience, 2003.

Sharan, U. and Neville, J.: Exploiting time-varying relationships in statistical relational models. In Proc. of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, pages 9–15. ACM, 2007.

Shetty, J. and Adibi, J.: The Enron email dataset database schema and brief statistical report. Technical report, Information Sciences Institute, University of Southern California, 2004.

Shetty, J. and Adibi, J.: Discovering important nodes through graph entropy the case of enron email database. In LinkKDD '05: Proc. of the 3rd Intl. workshop on Link discovery, pages 74–81, New York, NY, USA, 2005. ACM.

Shi, X., Tseng, B., and Adamic, L.: Looking at the blogosphere topology through different lenses. In ICSWM 07: Proc. of the Intl. Conf. on Weblogs and Social Media, 2007.

Soffer, S. N. and Vázquez, A.: Network clustering coefficient without degree-correlation biases. Phys. Rev. E, 71(5):057101, May 2005.

Sulo, R., Berger-Wolf, T., and Grossman, R.: Meaningful selection of temporal resolution for dynamic networks. In Proc. of the Eighth Wkshp. on Mining and Learning with Graphs, MLG '10, pages 127–136, New York, NY, USA, 2010. ACM.

Sun, J., Tao, D., and Faloutsos, C.: Beyond streams and graphs: dynamic tensor analysis. In Proc. of the 12th ACM SIGKDD intl. conf. on Knowledge discovery and data mining, pages 374–383. ACM, 2006.

Sun, J., Faloutsos, C., Papadimitriou, S., and Yu, P. S.: GraphScope: parameter-free mining of large time-evolving graphs. In Proc. of the 13th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining, pages 687–696, New York, NY, USA, 2007. ACM.

Sundaresan, S. R., Fischhoff, I. R., Dushoff, J., and Rubenstein, D. I.: Network metrics reveal differences in social organization between two fission–fusion species, Grevy's zebra and onager. Oecologia, 151(1):140–149, 2007.

Tantipathananandh, C., Berger-Wolf, T. Y., and Kempe, D.: A framework for community identification in dynamic social networks. In Proc. of the 13th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining, pages 717–726, 2007.

Tatti, N.: Maximum entropy based significance of itemsets. Knowledge and Information Systems, 17(1):57–77, 2008.

Thomas, A. C. and Blitzstein, J. K.: Valued ties tell fewer lies: Why not to dichotomize network edges with thresholds. ArXiv e-prints 1101.0788v2, January 2011.

Tong, H., Prakash, B., Tsourakakis, C., Eliassi-Rad, T., Faloutsos, C., and Chau, D.: On the vulnerability of large graphs. In 2010 IEEE Intl. Conf. on Data Mining, pages 1091–1096. IEEE, 2010.

Tukey, J. W.: We need both exploratory and confirmatory. The American Statistician, 34(1):23–25, 1980.

Tung, A. K., Lu, H., Han, J., and Feng, L.: Breaking the barrier of transactions: mining inter-transaction association rules. In Proc. of the 5th ACM SIGKDD Intl. Conf. on Knowl. disc. and data mining, pages 297–301, New York, NY, USA, 1999. ACM.

Vaz de Melo, P., Akoglu, L., Faloutsos, C., and Loureiro, A.: Surprising patterns for the call duration distribution of mobile phone users. Machine Learning and Knowledge Discovery in Databases, pages 354–369, 2010.

Vázquez, A., Pastor-Satorras, R., and Vespignani, A.: Large-scale topological and dynamical properties of the Internet. Physical Review E, 65(6):66130, 2002.

Vilalta, R. and Ma, S.: Predicting rare events in temporal domains. 2nd IEEE Intl. Conf. on Data Mining, page 474, 2002.

Wang, C., Satuluri, V., and Parthasarathy, S.: Local probabilistic models for link prediction. In Proc. of the Seventh IEEE Intl. Conf. on Data Mining, pages 322–331, 2007.

Wang, Y., Chakrabarti, D., Wang, C., and Faloutsos, C.: Epidemic spreading in real networks: An eigenvalue viewpoint. IEEE Symposium on Reliable Distributed Systems, 0:25, 2003.

Wang, Y.: On the number of successes in independent trials. Statistica Sinica, 3(2):295–312, 1993.

Wasserman, S. and Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, 1994.

Watts, D. J. and Strogatz, S. H.: Collective dynamics of 'small-world' networks. Nature, 393(6684):440–442, 1998.

Wei, C. and Chiu, I.: Turning telecommunications call details to churn prediction: a data mining approach. Expert systems with applications, 23(2):103–112, 2002.

West, D. B.: Introduction to graph theory. Prentice Hall, NJ, 2001.

Yan, X., Cheng, H., Han, J., and Yu, P. S.: Mining significant graph patterns by leap search. In SIGMOD '08: Proc. of the 2008 ACM SIGMOD Intl. Conf. on Management of data, pages 433–444, New York, NY, USA, 2008. ACM.

Yang, G.: The complexity of mining maximal frequent itemsets and maximal frequent patterns. In Proc. of the tenth ACM SIGKDD Intl. Conf. on knowledge discovery and data mining, pages 344–353, New York, NY, 2004. ACM.

Yang, J., Wang, W., and Yu, P. S.: InfoMiner: mining surprising periodic patterns. In Proc. of the 7th ACM SIGKDD Intl. Conf. on Knowledge discovery and data mining, pages 395–400, New York, NY, 2001. ACM.

Yang, J., Wang, W., and Yu, P. S.: InfoMiner+: Mining partial periodic patterns with gap penalties. In Proc. of the 2002 IEEE Intl. Conf. on Data Mining, page 725, Washington, DC, 2002. IEEE Computer Society.

Yang, J., Wang, W., and Yu, P. S.: Mining asynchronous periodic patterns in time series data. IEEE Transactions on Knowledge and Data Engineering, 15(3):613–628, 2003.

Zachary, W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research, 33(4):452–473, 1977.

Zhang, B. and Horvath, S.: A general framework for weighted gene co-expression network analysis. Statistical applications in genetics and molecular biology, 4(1):1128, 2005.

Zhu, F., Yan, X., Han, J., and Yu, P. S.: gPrune: a constraint pushing framework for graph pattern mining. Lecture Notes in Computer Science, 4426:388, 2007.

VITA

Name

Mayank Lahiri

Education

B.S. (Hons.), Computer Science, Lafayette College, Easton, Pennsylvania, 2005
Ph.D., Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2011

Honors

University of Illinois at Chicago

- UIC Image of Research competition, 1^{st} place, 2010.
- Provost's Award for Graduate Research, 2010
- Dean's Scholar University Fellowship, 2009 - 10
- Department of Computer Science Outstanding Teaching Assistant Award, 2008
- Department of Computer Science Leadership and Service Award, 2008
- UIC Graduate Research Forum, 2^{nd} place in the Physical Sciences, 2008
- UIC Graduate Research Forum, 1^{st} place in the Physical Sciences, 2007
- Graduate Student Council Travel Awards, 2007 & 2008

Lafayette College

- James P. Schwar Prize for Excellence in Computer Science, 2003
- EXCEL Research Scholar, May 2002 - May 2005

Research and Industrial Experience

Facebook, Inc., Palo Alto, CA, USA

- Summer intern, May to August 2010.

Telefonica Research and Development, Madrid, Spain

- Research intern, Data Mining and User Modeling Group, June to Sept. 2009

University of Illinois at Chicago, Chicago, IL, USA

- Research assistant, Computational Population Biology Laboratory, January 2006 to May 2011
- Research assistant, Discrete Event Systems Laboratory, May 2005 to December 2005

Lafayette College, Easton, PA, USA

- EXCEL research scholar, Dept. of Computer Science, May 2003 to May 2015
- EXCEL research scholar, Dept. of Physics, May 2002 to January 2003

Publications

Lahiri, M., Tantipathananandh, C., Warungu, R., Rubenstein, D. I., Berger-Wolf, T. Y.: Biometric Animal Databases from Field Photographs: Identification of Individual Zebra in the Wild. In Proc. of the ACM International Conference on Multimedia Retrieval, Trento, Italy, 2011.

Lahiri, M. and Cebrian, M.: The genetic algorithm as a general diffusion model for social networks. In Proc. of the 24th AAAI Conference on Artificial Intelligence, Atlanta, GA, 2010.

Cebrian, M., Lahiri, M., Oliver, N., Pentland, A.: Measuring the Collective Potential of Populations from Dynamic Interaction Data. IEEE Journal of Special Topics in Signal Processing. 4(4): 677–686, 2010.

Lahiri, M. and Berger-Wolf, T. Y.: Periodic Subgraph Mining in Dynamic Social Networks. Journal of Knowledge and Information Systems. 24(3): 467–497, 2009.

Lahiri, M. and Berger-Wolf, T. Y.: Mining Periodic Behavior in Dynamic Social Networks. In Proc. 8th IEEE International Conference on Data Mining, Pisa, Italy. IEEE Press, 2008.

Lahiri, M., Maiya, A.S., Sulo, R., Habiba, and Berger-Wolf, T. Y.: The Impact of Structural Changes on Predictions of Diffusion in Networks. In Proc. of the ICDM Workshop on the Analysis of Dynamic Networks, Pisa, Italy. IEEE Press, 2008.

Lahiri, M. and Berger-Wolf, T. Y.: Structure prediction in temporal networks using frequent subgraphs. In Proc. of IEEE Symposium on Computational Intelligence and Data Mining, pages 35–42, 2007.

Liew, C.W. and Lahiri, M.: Exploration or Convergence? Another Meta-Control Mechanism for GAs. In Proc. of the 18th International FLAIRS Conference, Clearwater, FL. AAAI Press, 2005.

Dougherty, A. and Lahiri, M.: Shape of ammonium chloride dendrite tips at small supersaturation. Journal of Crystal Growth. 274(2005): 233–240