Dynamic Learning from Time-Varying Social Networks

A DISSERTATION

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Brian Baingana

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Professor Georgios B. Giannakis, Advisor

May 2016

Acknowledgments

First, I would like to express my deepest gratitude to Professor Georgios B. Giannakis for giving me the opportunity to do my doctoral research under his guidance. I have benefited tremendously from his extensive expertise, clarity of thought, and his impressive ability to connect seemingly unrelated dots. More importantly, I would like to thank him for sparking in me the interest to embark on my doctoral research, as well as his patience with my shortcomings.

In a special way, I would like to extend my appreciation to Professors Mostafa Kaveh, Nikos Sidiropoulos, and George Karypis who agreed to take time out of their busy schedules and serve on my doctoral committee. Thanks also go to several other professors in the departments of Electrical Engineering, Computer Science, and Mathematics whose graduate level courses opened up my mind and prepared me to embark on this area of research.

My journey as a researcher has benefited from numerous collaborations and discussions with current and former members of SPiNCOM. Prof. Gonzalo Mateos, Dr. Emiliano Dall'Anese, Prof. Juan Andrés Bazerque, Dr. Daniele Angelosante, Dr. Pedro Forero, Prof. Ketan Rajawat, and Dr. Eric Msechu were instrumental during the earlier stages of my doctoral work. The rest of the SPiNCOM family has been quite supportive, namely: Profs. Konstantinos Slavakis, Vassilis Kekatos, Seung-Jun Kim, Antonio Marques, Ioannis Schizas, Nikos Gatsis, Hao Zhu, Drs. Alfonso Cano, Shahrokh Farahmand, Nasim Yahya Soltani, Morteza Mardani, Yu Zhang, Daniel Romero, Yunlong Wang, as well as students Fatemeh Sheikholeslami, Donghoon Lee, Panagiotis Traganitis, Dimitris Berberidis, Tianyi Chen, Liang Zhang, Yanning Shen, Georgios Karanikolas, Gang Wang, Meng Ma, and Vassilis Ioannidis.

My family has been the greatest source of inspiration along this journey. Special thanks to my beloved wife Paula for her constant love, patience, care, and unwavering belief in me, and my daughter Elizabeth, whose smiles and screams always brighten my day, and remind me of my blessings. Thanks go to my siblings, Wycliffe and Patricia, for always standing by my side through good and bad. Last but not least, words cannot express my appreciation for the priceless guidance I have received over the years from my wonderful guardians Dr. Josephine Namaganda and Mr. Ezra Bunyenyezi, who always encouraged me to dream bigger.

Brian Baingana, Minneapolis, May 5, 2016

Dedicated to my beloved parents, the late Drs. Elidad & Goretti Baingana.

Abstract

One of the foremost intellectual challenges of this century is to understand the collective behavior of complex systems. Such systems are ubiquitous, and range from "engineered systems" including the Internet and online social grids, to complex natural phenomena such as neural connections in the brain, and interactions between genes. Networks lie at the heart of complex systems, encoding pairwise interactions between their constituent components. In this regard, complexity captures the fact that it is difficult to derive holistic system behavior from knowledge of individual components. The key premise of network science is that despite the diversity of complex systems, the behavior of their underlying networks is driven by a common set of laws.

Contemporary studies focus on models and tools to understand, predict, and control the behavior of networks. However, most of these approaches are tailored to analysis of static networks, whose node and link structure does not change with time. Cognizant of the dynamic nature of most real-world networks, analysts mostly focus on static snapshots or aggregate views of studied systems, and meaningful insights cannot be guaranteed. Indeed, the recently growing trend in analysis of dynamic networks is testament to the critical need to live up to this challenge. Moreover, issues arising from temporal network evolution are exacerbated by inherent Big Data challenges. Many large-scale networks comprise billions of nodes, which are typically associated with high-dimensional, and streaming features. Furthermore, it is often impractical to observe the entire network, and analyses must be conducted on manageable or easily accessible samples of the network.

Acknowledging these limitations, this dissertation leverages recent advances in statistical signal processing, optimization, and machine learning to address the aforementioned challenges. Emphasis is placed on statistical learning approaches capable of exploiting sparsity, or low rank, attributes that have been shown useful for complexity reduction. Focusing on canonical network inference tasks such as topology identification, detection of communities, and unveiling anomalous nodes, this dissertation puts forth novel statistical models, and develops efficient algorithms for dynamic network analytics. Motivated by the need for real-time processing, online renditions of the developed algorithms are advocated for handling streaming network data. For each of the research themes considered, extensive tests are conducted on simulated and real data, while pertinent comparisons with competing approaches are drawn wherever possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Network science has been heralded as crucial to understanding complex systems [24, 40, 53, 78]. Looking back over the last 150 years, networks have underpinned many ground-breaking technological revolutions e.g., the electric power grid, telecommunications, and the world-wide web, to name a few. More recently, there has been growing interest in analysis and characterization of the behavior of a more diverse class of networks e.g., social networks capturing human connections, gene and protein interactions in cells of living organisms, as well as brain networks comprising billions of neurons. This interest has mostly been fueled by two major factors. First, modern large-scale networks have made it possible to cheaply collect unprecedented amounts of data e.g., the Internet [44], the world wide web [39], and online social networks such as Facebook [125]. Second, rapid computational advances and plummeting memory costs have facilitated cheap archival of networks and networked data.

Network science is an inherently multidisciplinary effort that leverages tools from fields as diverse as graph theory, statistics, signal processing, machine learning, and numerical optimization. This multifaceted nature is hardly surprising since complex systems permeate virtually all spheres of life. Early network studies pioneered in statistical physics sought to unveil universal laws explaining pervasive properties observed in real networks. For example, why do empirical node degree distributions often follow power laws [2, 3, 54, 101], or why does small-world behavior naturally emerge within large-scale real-world networks [1, 5–7, 25, 27–29, 76, 77, 100]? Other studies focused on the clustering behavior of nodes into natural communities [59, 62, 102, 104], developing random models

capable of generating networks that exhibit some of the observed properties [78, Ch. 6], ranking of nodes (and edges) in order of "importance", and estimation of network-wide metrics from a sample of observable nodes and edges, to name just a few.

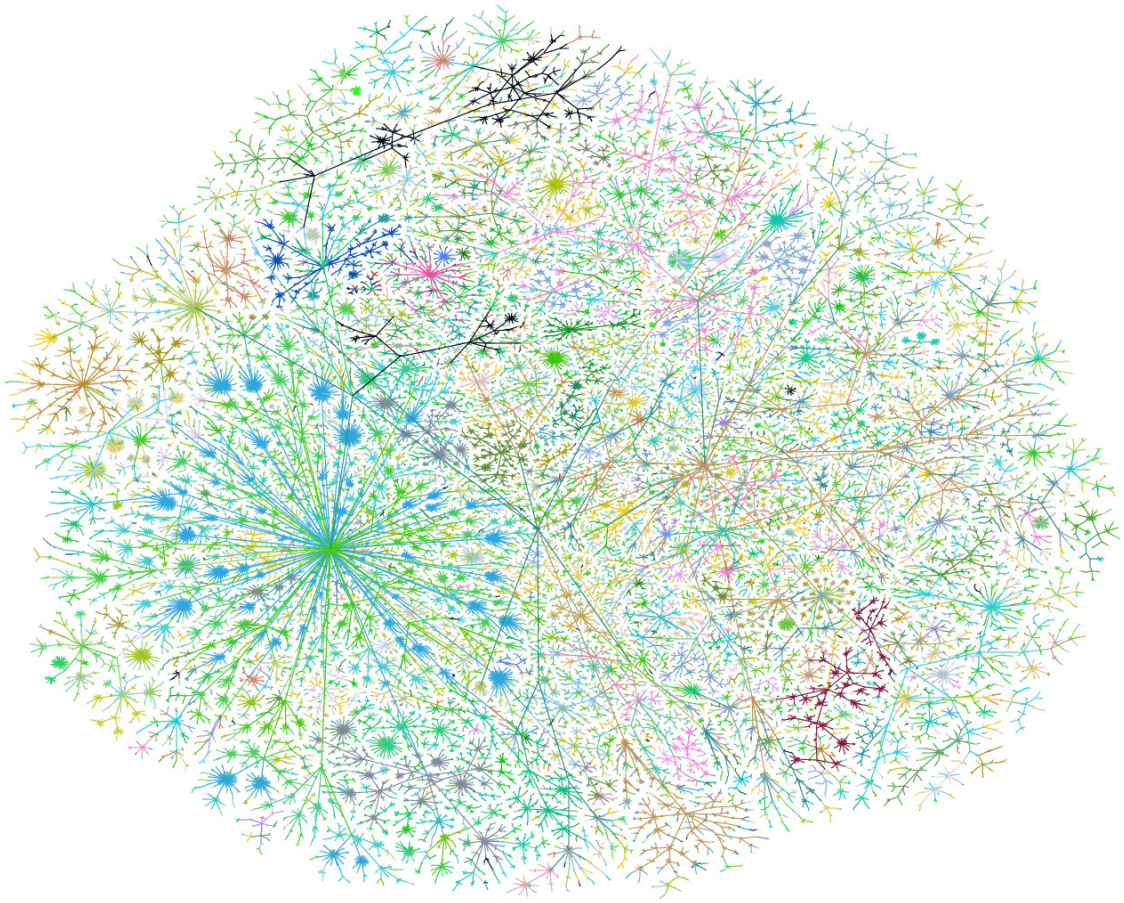


Figure 1.1: Map of the Internet at the level of resolution of autonomous systems (AS) based on data collected in August 1998. Nodes depict Internet service providers (ISPs), and links capture router-based connections between ISPs. The Internet is a classical example of a complex system whose behavior is best studied when abstracted as a network.

From a data analytics perspective, modern networks present ample challenges and opportunities. Most contemporary approaches assume that networks are static. In reality most networks are dynamic, with time-varying topologies e.g., the world-wide web grows daily with new pages and links constantly emerging, and people connect with new friends while severing old ties in online social networks. Analysts typically construct static "aggregate" views as abstractions of dynamic

networks, an approach that discards useful temporal information.

Furthermore, "Big Data" challenges abound in real-world networks consisting of millions of nodes and edges. Even when these big networks are fully observable, their nodes and edges are often associated with data that is both highly dimensional, and streaming in nature. Examples of such data include text, images, and videos posted daily by Facebook users, *tweets* that permeate the Twitter network, or blog content continually posted on the web. This data deluge far outpaces capabilities of contemporary storage and data processing tools available to network operators. Many distributed platforms (e.g., Hadoop/MapReduce and GraphLab) have emerged to tame the sheer scale of data. Nevertheless, only a few algorithms are readily implementable on these platforms, and development of more versatile architectures remains an active area of research.

Despite recent interest in analysis of dynamic networks (see e.g., [4, 85, 92, 111, 115, 134, 135]), it is only the beginning, and a unified modeling and algorithmic framework to address the aforementioned challenges remains elusive. In lieu of these challenges, this dissertation focuses on addressing some of the aforementioned challenges that especially arise in dynamic social network applications. Building upon advances in statistical learning, optimization, and signal processing, a novel framework that encompasses several efficient approaches for analysis of dynamic networks arising in such settings is advocated.

1.1 Motivation and context

Statistical learning and signal processing constitute a rich framework within which the identified challenges can be readily addressed. Recent advances have formalized models and algorithms that exploit properties inherent to a number of complex and big data sets. For example, most high-dimensional data exhibit sparsity in some domain, while redundancies in data often lead to low rank properties. It is also well-documented that many high-dimensional data tend to lie within lower dimensional manifolds. Exploiting these properties has unlocked the potential to develop very efficient algorithms, capable of learning unknown parameters of interest with markedly reduced computational complexity.

Not surprisingly, some of these desirable properties carry over to network settings. For instance, edge sparsity is inherent to networks since each node only links with a relatively small subset of all

other nodes. Existence of cohesive subgroups or communities of nodes yields matrix representations (e.g., adjacency matrices and graph Laplacians) that are often rank deficient. Low rank is also manifested in temporal networks whose topologies vary slowly between consecutive time intervals. Capitalizing on such properties, this dissertation develops a toolkit of approaches that extend the merits of recent advances in statistical learning to inference tasks for large-scale dynamic networks. The rest of this section identifies two motivating application settings in the context of dynamic social networks.

1.1.1 Diffusion of cascades over networks

Networks often provide the backbone for diffusion of complex processes or *contagions*. Examples include the spread of popular news over the web, sharing of a viral *meme* among Twitter users, adoption of buying trends among consumers, or epidemics within a population [53, 113]. A *cascade* emerges when such a contagion rapidly diffuses over a significant subset of nodes in the network. The underlying network topology is often unknown and dynamic, but "social signals" such as the purchase time of a new product, or when a blog first mentioned a popular piece of news are readily observable, and easily measurable. Note that such networks of causal influence may exist on top of other more apparent networks. For example, a network representation of web pages and links connecting them is different from one in which edges represent directions of causal influence. Similar examples can be identified for large-scale social networks such as Facebook and Twitter.

From a practical perspective, network operators equipped with comprehensive tools for tracking unknown dynamic topologies can make more informed decisions e.g., identifying which set of initiators in a social network will render an advertising campaign viral, or which injecting drug users should be vaccinated to curb the rapid spread of a disease within their hidden community. Other examples include assessment of the reliability of heavily interconnected systems like power grids, or risk exposure among investment banks in a highly inter-dependent global economy. In general, knowledge of topologies that facilitate diffusion of network processes leads to useful insights about the behavior of complex systems. While cognizant of the aforementioned challenges, this thesis develops a novel framework for tracking dynamic networks of causal influence from cascade measurements.

1.1.2 Evolution of network communities

Cohesive behavior is pervasive in observed real-world networks, with tendencies for nodes to cluster into groups known as *communities* [59, 101, 102]. Node clusters exhibit markedly higher intra-group edge density compared with the rest of the network. This phenomenon has been observed in a number of systems e.g., social networks, protein-protein interaction networks, and the world-wide web. Identification of communities finds diverse practical applications e.g., grouping social media subscribers into functional roles for more informative advertising, or clustering blogs into content groups, facilitating improved recommendations to readers. Communities tend to overlap, or to hierarchically contain smaller communities. For example, a Facebook subscriber may be jointly affiliated with work colleagues, high school friends, and family - different, but overlapping communities. In addition, communities naturally evolve over time as a consequence of time-varying network topologies. As a result, it is important to devise community models that capture temporal variations, as well as efficient algorithms to track the hidden and possibly overlapping communities. Building upon recent advances in identification of temporal communities, this thesis puts forth a novel "robust" model for tracking dynamic communities.

1.2 Thesis outline and contributions

The results of this thesis contribute to the advancement of the theory and methods for analysis of complex networks in several ways. Fundamentally, the motivation is to develop parsimonious models and efficient algorithms, capable of dynamic inference from temporal networks. This is accomplished by leveraging recent advances in statistical signal processing, numerical optimization, and machine learning. For example sparsity- and rank-promoting regularizers are adopted to incorporate prior knowledge about the structure of network topologies and their inherent communities. Motivated by challenges arising from big data analytics, emphasis on first-order and real-time tracking algorithms is adopted as the guiding philosophy for all developed algorithms. Furthermore, extensive tests on both synthetic and real data are conducted to demonstrate the efficacy of the novel approaches. Accordingly, the contributions of this thesis can be divided into three broad thrusts:

T1: Tracking slowly-varying network topologies. In most settings, changes to network topolo-

gies emerge slowly between observation intervals. Capitalizing on this prior knowledge, a novel temporal model for propagation of cascading events over networks with hidden topologies is postulated. The temporal model explicitly captures the fact that cascade propagation results from both causal topological interactions between nodes, and exogenous sources of influence. Efficient tracking algorithms that are capable of real-time operation are developed.

T2: Identification of switched network topologies. Ample evidence suggests that topologies of certain well-known networks switch suddenly between discrete topological states. Cascade behavior over such networks reflects the underlying switching dynamics. For instance, certain users only follow sports news and tweets during big sporting events, yet they are usually indifferent to such news on other days. This thrust advocates a novel systematic framework for modeling cascade propagation over switching networks, and identification of both the underlying topological states and the switching sequence that governs the network dynamics.

T3: Joint tracking of dynamic communities and anomalies. Real networks often contain anomalous nodes that distort the true underlying community structure. Examples include spammers in e-mail networks, or phishing accounts in online social networks. In this thrust, a robust community model that accounts for the presence of such distortive anomalies is postulated. Assuming that both the network topology, and the unknown communities evolve slowly, novel algorithms capable of jointly identifying communities and anomalous nodes are developed.

For each of the thrusts, extensive experiments are conducted with synthetically-generated data to corroborate the effectiveness of the developed algorithms. In addition, further experiments on real publicly-accessible datasets are conducted, and interesting insights are revealed in all cases. The rest of this section is devoted to an elaborate discussion of T1-T3, along with a detailed literature review per thrust. Pertinent contributions in context are clearly outlined as well.

1.2.1 Tracking slowly-varying network topologies

Inference of networks using temporal traces of infection events has recently become an active area of research. According to the taxonomy in [78, Ch. 7], this can be viewed as a problem involving inference of *association* networks. Two other broad classes of network topology identification

problems entail link prediction, or, tomographic inference. This thrust proposes a novel model to capture the evolution of cascades over unknown, dynamic networks, and develops several efficient inference algorithms for tracking the evolution of their time-varying topologies. In most cases, dynamic networks over which cascades propagate evolve slowly with time. For example, social networks facilitating the spread of rumors and viral memes are relatively stable, since the majority of subscribers connect with new acquaintances over long time horizons.

Capitalizing on this prior knowledge, a novel temporal model for propagation of cascading events over a hidden dynamic network is postulated. This is accomplished by resorting to structural equation models (SEMs), a class of statistical methods that capture causal relations between entities in a complex system; see e.g., [65] and references therein. Originally advocated for studies pertaining to discovery of causal factors responsible for differences in guinea pig breeding [131], SEMs found useful causal modeling applications in psychology, econometrics and biology [37, 69, 109, 119]. Thanks to their inherent simplicity, SEMs have also been advocated for inference of directed network topologies e.g., gene regulatory networks [42, 43, 51, 88, 89, 133], and effective connectivity studies for brain networks [41, 95].

Leveraging this rich framework, the first contribution is a novel dynamic SEM (DSEM), capable of modeling cascade propagation over temporal networks whose topologies vary slowly. The second contribution pertains to exploiting the ability of SEMs to incorporate exogenous inputs in causal systems, in order to account for non-topological factors that are responsible for cascade propagation e.g., sports blogs are expected to play a bigger role in propagating sports-related news events than political blogs regardless of their connectivity. This is a step ahead of contemporary network inference methods which assume that dynamics of processes propagating over networks are entirely due to topology-based interactions. Finally, efficient and parallelizable topology tracking algorithms, capable of online operation are developed.

Although network inference has been studied extensively in the past, most approaches dwell on static networks; see [111], [97], [78, Ch. 7] and references therein. Nevertheless, there is recent growing interest in dynamic inference of temporal hidden networks from diffusion events. Network Granger causality with group sparsity is advocated for inference of causal networks with inherent grouping structure in [31], while causal influences are inferred by modeling historical network

events as multidimensional *Hawkes* processes in [140]. A prominent school of thought postulates probabilistic epidemic models for cascade propagation, with time-varying but unknown parameters that capture the evolving topology; see e.g., [112]. This family of approaches develops maximum likelihood estimators for tracking the distribution parameters, which is tantamount to dynamic inference of the network. Specifically, [112] advocates stochastic gradient descent iterations for the MLE problem, which facilitates scaling to big data. In [52], a Bayesian framework that tracks a low-dimensional node embedding in Euclidean space is advocated for inference of edge probabilities. Other popular Bayesian approaches assume dynamic stochastic block models (DSBM) for edge generation, with topology inference boiling down to tracking the model parameters [57, 70, 134]. All in all, these approaches presume knowledge of the probability distributions responsible for edge evolution. The main novelty in the advocated framework lies in leveraging DSEMs as a statistical formalism for capturing causal interactions responsible for propagation of contagions, while jointly accounting for the influence of exogenous inputs.

1.2.2 Tracking switched dynamic network topologies from information cascades

Network topologies may sometimes "jump" between a finite number of discrete states, as manifested by sudden changes in cascade behavior. For example, an e-mail network may switch topologies from predominantly work-based connections during the week, to friend-based connections over the weekend. Connection dynamics between bloggers may switch suddenly at the peak of sports events (e.g., "Superbowl"), or presidential elections. In such settings, contemporary approaches assuming that network dynamics arise as a result of slow topology variations may yield unpredictable results. This thrust capitalizes on this prior knowledge, and puts forth a novel *switched* dynamic SEM to account for propagation of information cascades in such scenarios. The proposed approach builds upon T1, where it is tacitly assumed that node infection times depend on both the switching topologies and exogenous influences e.g., external information sources, or prior predisposition to certain cascades.

Connections are drawn to identification of hybrid systems, whose behavior is driven by interaction between continuous and discrete dynamics; see e.g., [107] and references therein. Switched linear models have emerged as a useful framework to capture piecewise linear input-output rela-

tions in control systems [23, 114, 126]. The merits of these well-grounded approaches are extended to temporal network inference from state-driven cascade dynamics. Under reasonable assumptions, conditions under which the underlying network states are identifiable are derived. Leveraging edge sparsity, an efficient approach for discovery of the network topologies, and tracking of the switching sequence is advocated. Extensive tests on both simulated and real web cascades corroborate the effectiveness of the novel approach.

1.2.3 Joint tracking of dynamic communities and anomalies

Although closely related to the graph partitioning problem [71–73], community detection is a fundamentally more challenging task due to the ambiguity associated with the proper definition of a network community. In the classical static setup, detection of communities is a well-studied problem with several methods proposed over the years (see [55] for a comprehensive survey). These approaches range from standard data clustering algorithms such as hierarchical clustering algorithms [64], [68], and spectral clustering [90], to modularity-based algorithms [103]. Recently, non-negative matrix factorization (NMF) methods have been advocated for detection of overlapping communities in social networks [92, 115, 135]. NMF is a general unsupervised learning paradigm that has recently received a lot of attention for dimensionality reduction and clustering [67, 82, 83, 105, 106, 128]. Most initial applications of NMF have involved discovery of latent factors in non-negative data such as images and web data amenable to collaborative filtering [61, 116, 120, 138, 139]. By endowing networks with appropriate basis expansions, NMF turns out to be a useful tool for modeling overlapping network communities.

Despite the existence of many dynamic networks, only a few studies have focused on studying dynamic community structure. A state evolution model for community membership parameters of a dynamic stochastic block model for social networks was proposed in [134]. Boiling down to a parameter tracking task, an extended Kalman filter was developed to detect evolving communities arising in several networks. In [92], NMF was proposed for discovery of overlapping dynamic communities. Other representative works for dynamic tracking of network communities include [30, 60, 87, 123].

In addition to temporal variations, "true" communities are often distorted by aberrant nodes that

exhibit anomalous behavior, often manifested by unusually strong, uni-directional edge connectivity across communities e.g., e-mail spammers, or phishing accounts in online social networks. Contemporary community tracking approaches are generally incognizant of the presence of such distortive anomalies. The main contribution of this thrust, is to postulate a robust model, as well as efficient algorithms for jointly tracking evolving communities, while compensating for the effect of anomalies. Dynamic overlapping communities within temporal networks are identified by capitalizing on NMF. Leveraging sparsity introduced by anomalous nodes, robust models are put forth for joint tracking of communities and anomalies. Efficient algorithms capable of both online and decentralized implementation are developed, and extensive numerical tests on synthetic and real networks are conducted.

1.3 Published results

The present Ph.D. work on dynamic learning from time-varying networks has resulted in the publication of several journal papers in the Institute of Electrical and Electronic Engineers (IEEE) Transactions on Signal Processing [16, 17], and the IEEE Journal of Special Topics in Signal Processing [20]. The work has also been disseminated at pertinent conferences and workshops, where a total of 10 articles have been accepted for presentation [10–15, 18, 19, 21, 32]. Finally contents of this dissertation have been accepted for publication as part of a chapter titled *Big Data Analytics for Social Networks* in the published book *Graph Analysis for Social Media* [22].

1.4 Notational conventions

The following notational conventions will be adopted throughout subsequent chapters in this dissertation. Bold uppercase letters will denote matrices, whereas bold lowercase letters will stand for column vectors. Scalar quantities will be denoted by plain lowercase letters of both the English and Greek alphabets, whereas calligraphic letters will be used to denote sets. Operators $\mathbf{X}^T$ denotes transpose of matrix $\mathbf{X}$, and $\mathbf{X}^\dagger$ denotes its *Penrose-Moore pseudoinverse*; $\mathrm{Tr}\,(\mathbf{X})$ is the trace operator of matrix $\mathbf{X}$; $\|\mathbf{x}\|_2$ is the Euclidean norm of $\mathbf{x}$; $\mathbf{e}_i$ is the i-th column of the identity matrix, $\mathbf{I}$. $\mathbf{X}_{\mathcal{I},\mathcal{J}}$ is the submatrix of $\mathbf{X}$ obtained by retaining the common elements in rows indexed by the

elements of $\mathcal{I}$ and columns indexed by $\mathcal{J}$. Lastly, $\partial f(.)$ denotes the subdifferential of $f(.)$.

Chapter 2

Inference of Dynamic Causal Network Topologies

2.1 Introduction

Propagation of cascades such as viral news, popular tweets, or fashion trends is often facilitated by implicit large-scale networks, that are generally directed and dynamic, but unobservable. However, the time when a node is infected by a given cascade is observable, and will henceforth be referred to as the "infection time." This chapter develops algorithms for tracking the underlying temporal topologies from observable cascade traces or diffusion processes. Motivated by practical considerations in settings such as the world-wide web and online social networks, it will be assumed that network topologies have sparse edges, and evolve slowly over time.

Structural equation models (SEMs) are a general family of approaches that effectively capture causal relationships between interacting variables in complex systems. These directional effects are often not revealed by standard linear models that leverage symmetric associations between random variables, such as those represented by covariances or correlations [9,56,79,96]. Building upon this well-known causal modeling framework, this chapter proposes a novel *dynamic* SEM to account for directed networks over which cascades propagate. Key emphasis is placed on describing how node infection times depend on both topological (endogenous) and external (exogenous) influences.

Topological influences are modeled in Section 2.2 as linear combinations of infection times of

other nodes in the network, whose weights correspond to entries in the time-varying asymmetric adjacency matrix. Accounting for external influences is well motivated by drawing upon examples from online media, where established news websites depend more on on-site reporting than blog references. External influence data is also useful for model identifiability, since it has been shown necessary to resolve directional ambiguities; see Chapter 3 for details. Supposing the network varies slowly with time, parameters in the proposed dynamic SEM are estimated adaptively by minimizing a sparsity-promoting exponentially-weighted least-squares (LS) criterion (Section 2.3.1). To account for the inherently sparse connectivity of social networks, an ℓ_1-norm regularization term that promotes sparsity on the entries of the network adjacency matrix is incorporated in the cost function; see also [8, 9, 46, 80].

Novel algorithms to jointly track the network's adjacency matrix and the weights capturing the level of external influences are developed in Section 2.3, by minimizing the resulting non-differentiable cost function via proximal optimization methods; see e.g., [34, 47, 48, 50, 108]. The developed algorithms are provably convergent, entail closed-form updates, and offer ample opportunities for parallel operation.

Further algorithmic improvements are outlined in Section 2.4. These include enhancing the algorithms' rate of convergence through Nesterov's acceleration techniques [34, 98, 99] (Section 2.4.1), and also adaptation for real-time operation (Section 2.4.2). When minimal computational complexity is at a premium, a stochastic gradient descent (SGD) algorithm is developed in Section 2.4.3, which adaptively minimizes an instantaneous (noisy) approximation of the ensemble LS cost.

Numerical tests with synthetically-generated data demonstrate the effectiveness of the novel algorithms in unveiling sparse dynamically-evolving topologies (Section 2.5.1). Experiments in Section 2.5.2 involve real temporal traces of popular global events that propagated on news websites and blogs in 2011 [86]. For instance, topologies inferred from cascades associated with the meme "Kim Jong-un" exhibit an abrupt increase in the number of edges following the appointment of the new North Korean ruler.

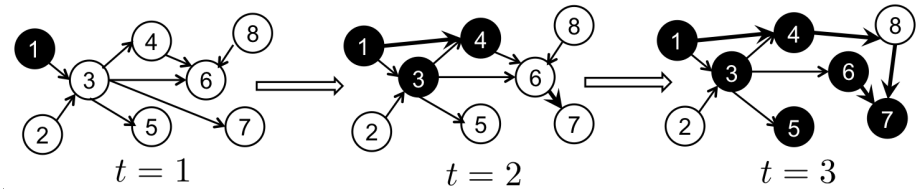


Figure 2.1: A single cascade propagating over a dynamic network, with "infected" nodes depicted in black.

2.2 Model and Problem Statement

Consider a dynamic network with N nodes observed over time intervals $t = 1, \ldots, T$, whose abstraction is a graph with topology described by an unknown, time-varying, and weighted adjacency matrix $\mathbf{A}^t \in \mathbb{R}^{N \times N}$. Entry (i, j) of $\mathbf{A}^t$ (henceforth denoted by a_{ij}^t) is nonzero only if a directed edge connects nodes i and j (pointing from j to i) during the time interval t, as illustrated in the 8-node network in Fig. 2.1. As a result, one in general has $a_{ij}^t \neq a_{ji}^t$, i.e., matrix $\mathbf{A}^t$ is generally non-symmetric, which is suitable to model directed networks. For instance, if i denotes a news blog maintained by a journalism student, whereas j represents the web portal of a mainstream newspaper, then it is likely that $a_{ij}^t \gg a_{ji}^t \approx 0$ for those t where $a_{ij}^t \neq 0$. Probably, the aforementioned directionality would have been reversed during Nov.-Dec. 2010, if i instead represents the famous *Wikileaks* blog. Note that the model tacitly assumes that the network topology remains fixed during any given time interval t, but can change across time intervals.

Suppose C contagions are sampled out of all contagions that propagate over the network during the observation interval, and the difference between infection time of node i by contagion c and the earliest observation time is denoted by $y_{ic}^t \geq 0$. In online media, y_{ic}^t can be obtained by recording the time when website i mentions news item c. For uninfected nodes at interval t, y_{ic}^t is infinite and is set to a large positive value for practical considerations. Assume that the susceptibility x_{ic} of node i to external (non-topological) infection by contagion c is known and time invariant over the observation interval. In the web context, x_{ic} can be set to the search engine rank of website i with respect to (w.r.t.) keywords associated with c.

In order to track the unknown network topology, this chapter postulates that y_{ic}^t is linearly related to x_{ic} and the infection times of its single-hop neighbors. Events that adhere to this model of network-facilitated propagation abound on the web where mention of e.g., a major baseball event

by a blog will not only depend on the times when similar blogs first reported the event, but also the level of interest of the blogger in baseball as a sport. In epidemiological studies, an individual's infection time by an infectious disease depends on the infection times of her immediate contacts as well as her level of immunity to the disease. Consequently, y_{ic}^t is modeled according to the following linear *dynamic* structural equation model (SEM)

$$y_{ic}^t = \sum_{j \neq i} a_{ij}^t y_{jc}^t + b_{ii}^t x_{ic} + e_{ic}^t \tag{2.1}$$

where b_{ii}^t captures the time-varying level of influence of external sources, and e_{ic}^t accounts for measurement errors and unmodeled dynamics. It follows from (2.1) that if $a_{ij}^t \neq 0$, then y_{ic}^t is affected by the value of y_{jc}^t. Rewriting (2.1) for the entire network leads to the vector model

$$\mathbf{y}_c^t = \mathbf{A}^t \mathbf{y}_c^t + \mathbf{B}^t \mathbf{x}_c + \mathbf{e}_c^t \tag{2.2}$$

where the $N \times 1$ vector $\mathbf{y}_c^t := \left[y_{1c}^t, \ldots, y_{Nc}^t\right]^\top$ collects the node infection times by contagion c during interval t, and $\mathbf{B}^t := \text{diag}(b_{11}^t, \ldots, b_{NN}^t)$. Similarly, $\mathbf{x}_c := [x_{1c}, \ldots, x_{Nc}]^\top$ and $\mathbf{e}_c^t := \left[e_{1c}^t, \ldots, e_{Nc}^t\right]^\top$. Collecting observations for all C contagions yields the dynamic matrix SEM

$$\mathbf{Y}^t = \mathbf{A}^t \mathbf{Y}^t + \mathbf{B}^t \mathbf{X} + \mathbf{E}^t \tag{2.3}$$

where $\mathbf{Y}^t := \left[\mathbf{y}_1^t, \ldots, \mathbf{y}_C^t\right]$, $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_C]$, and $\mathbf{E}^t := \left[\mathbf{e}_1^t, \ldots, \mathbf{e}_C^t\right]$ are all $N \times C$ matrices. Note that the same network topology $\mathbf{A}^t$ is adopted for all contagions, which is suitable e.g., when different information cascades are formed around a common meme or trending (news) topic over the web; see also the real data tests in Section 2.5.2. For this same reason, it is natural to assume that all conditional error variances are equal.

Given $\{\mathbf{Y}^t\}_{t=1}^T$ and $\mathbf{X}$, the goal is to track the underlying network topology $\{\mathbf{A}^t\}_{t=1}^T$ and the effect of external influences $\{\mathbf{B}^t\}_{t=1}^T$. To this end, the novel algorithm developed in the next section assumes slow time variation of the network topology and leverages the inherent sparsity of edges that is typical of social networks.

2.3 Topology Tracking Algorithm

This section deals with a regularized LS approach to estimating $\{\mathbf{A}^t, \mathbf{B}^t\}$ in (2.3). In a *static* setting with all measurements $\{\mathbf{Y}^t\}_{t=1}^T$ available, one solves the batch problem

$$\begin{aligned} \{\hat{\mathbf{A}}, \hat{\mathbf{B}}\} = \arg\min_{\mathbf{A},\mathbf{B}} \quad & \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{Y}^t - \mathbf{A}\mathbf{Y}^t - \mathbf{B}\mathbf{X}\|_F^2 + \lambda\|\mathbf{A}\|_1 \\ \text{s. to} \quad & a_{ii} = 0,\ b_{ij} = 0,\ \forall i \neq j \end{aligned} \tag{2.4}$$

where $\|\mathbf{A}\|_1 := \sum_{i,j} |a_{ij}|$ is a sparsity-promoting regularization, and $\lambda > 0$ controls the sparsity level of $\hat{\mathbf{A}}$. Absence of a self-loop at node i is enforced by the constraint $a_{ii} = 0$, while having $b_{ij} = 0,\ \forall i \neq j$, ensures that $\hat{\mathbf{B}}$ is diagonal as in (2.2).

2.3.1 Exponentially-weighted LS estimator

In practice, measurements are typically acquired in a sequential manner over large social networks with thousands or even millions of nodes, calling for estimation algorithms with minimal storage requirements. Recursive solvers enabling sequential inference of the underlying network topology are thus preferred. Moreover, introducing a "forgetting factor" that assigns more weight to the most recent residuals makes it possible to track slow temporal variations of the topology. Note that the batch estimator (2.4) yields single estimates $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}\}$ that best fit the data $\{\mathbf{Y}^t\}_{t=1}^T$ and $\mathbf{X}$ over the whole measurement horizon $t = 1, \ldots, T$, and as such (2.4) neglects potential network variations across time intervals.

For $t = 1, \ldots, T$, the sparsity-regularized exponentially-weighted LS estimator (EWLSE)

$$\begin{aligned} \{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\} = \arg\min_{\mathbf{A},\mathbf{B}} \quad & \frac{1}{2}\sum_{\tau=1}^{t} \beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2 + \lambda_t\|\mathbf{A}\|_1 \\ \text{s. to} \quad & a_{ii} = 0,\ b_{ij} = 0,\ \forall i \neq j \end{aligned} \tag{2.5}$$

where $\beta \in (0, 1]$ is the forgetting factor that forms estimates $\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\}$ using all measurements acquired until time t. Whenever $\beta < 1$, past data are exponentially discarded thus enabling tracking of dynamic network topologies. The first summand in the cost corresponds to an exponentially-weighted moving average (EWMA) of the squared model residuals norms. The EWMA can be seen as an average modulated by a sliding window of equivalent length $1/(1-\beta)$, which clearly

grows as $\beta \to 1$. In the so-termed infinite-memory setting whereby $\beta = 1$, (2.5) boils down to the batch estimator (2.4). Notice that λ_t is allowed to vary with time in order to capture the generally changing edge sparsity level. In a linear regression context, a related EWLSE was put forth in [8] for adaptive estimation of sparse signals; see also [80] for a projection-based adaptive algorithm. Before moving on to algorithms, a couple remarks are in order.

Remark 1 (Modeling slow network variations via sparsity)**.** To explicitly model slow topological variations across time intervals, a viable approach is to include an additional regularization term $\mu_t \|\mathbf{A} - \hat{\mathbf{A}}^{t-1}\|_1$ in the cost of (2.5). This way, the estimator penalizes deviations of the current topology estimate relative to its immediate predecessor $\hat{\mathbf{A}}^{t-1}$. Through the tuning parameter μ_t, one can adjust the level of smoothness exhibited by the admissible topology variations from interval to interval. With a similar goal but enforcing temporal smoothness via kernels with adjustable bandwidth, an ℓ_1-norm-regularized logistic regression approach was put forth in [79].

Remark 2 (Selection of λ_t)**.** Selection of the (possibly time-varying) tuning parameter λ_t is an important aspect of regularization methods such as (2.5), because λ_t controls the sparsity level of the inferred network and how its structure may change over time. For sufficiently large values of λ_t one obtains the trivial solution $\hat{\mathbf{A}}^t = \mathbf{O}_{N\times N}$, while increasingly more dense graphs are obtained as $\lambda_t \to 0$. An increasing λ_t will be required for accurate estimation over extended time-horizons, since for $\beta \approx 1$ the norm of the LS term in (2.5) grows due to noise accumulation. This way the effect of the regularization term will be down-weighed unless one increases λ_t at a suitable rate, for instance proportional to $\sqrt{\sigma^2 t}$ as suggested by large deviation tail bounds when the errors are assumed $e_{ic}^t \sim \mathcal{N}(0, \sigma^2)$, and the problem dimensions N, C, T are sufficiently large [8, 94, 96]. In the topology tracking experiments of Section 2.5, a time-invariant value of λ is adopted and typically chosen via trial and error to optimize performance. This is justified since smaller values of β are selected for tracking network variations, which also implies that past data (and noise) are discarded faster, and the norm of the LS term in (2.5) remains almost invariant. As future research it would be interesting to delve further into the choice of λ_t using model selection techniques such as cross-validation [43], Bayesian information criterion (BIC) scores [79], or the minimum description length (MDL) principle [110], and investigate how this choice relates to statistical model consistency in a dynamic setting.

2.3.2 ADMM solver

Exploiting the problem structure in (2.5), an alternating-direction method of multipliers (ADMM) algorithm is developed to track the network topology. Leaving the equality constraints ($a_{ii} = 0,\ b_{ij} = 0,\ \forall i \neq j$) temporarily implicit and introducing a dummy variable $\mathbf{C} \in \mathbb{R}^{N\times N}$, leads to the following equality constrained optimization problem:

$$\begin{aligned}\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t, \hat{\mathbf{C}}^t\} = \underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\arg\min} \quad & \frac{1}{2}\sum_{\tau=1}^{t}\beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2 + \lambda_t\|\mathbf{C}\|_1 \\ \text{s. to} \quad & \mathbf{A} = \mathbf{C}. \end{aligned} \tag{2.6}$$

With $\rho > 0$ denoting the penalty parameter and $\mathbf{\Gamma}$ the matrix of dual variables, the augmented Lagrangian for (2.6) is

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{\Gamma}) := \frac{1}{2}\sum_{\tau=1}^{t}\beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2 + \lambda_t\|\mathbf{C}\|_1 \\ + \text{Tr}\left(\mathbf{\Gamma}^\top(\mathbf{A}-\mathbf{C})\right) + \frac{\rho}{2}\|\mathbf{A}-\mathbf{C}\|_F^2. \end{aligned} \tag{2.7}$$

During iteration $k+1$ of the ADMM algorithm, alternating minimization of $\mathcal{L}_\rho(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{\Gamma})$ with respect to $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, followed by an update of the dual variables yields

$$\begin{aligned}\mathbf{A}[k+1] &= \underset{\mathbf{A}}{\arg\min}\ (1/2)\sum_{\tau=1}^{t}\beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}[k]\mathbf{X}\|_F^2 \\ &\quad + \ (\rho/2)\|\mathbf{A}-\mathbf{C}[k]\|_F^2 + \text{Tr}\left(\mathbf{\Gamma}^\top[k](\mathbf{A}-\mathbf{C}[k])\right) \qquad (2.8a) \\ \mathbf{B}[k+1] &= \underset{\mathbf{B}}{\arg\min}\ (1/2)\sum_{\tau=1}^{t}\beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}[k+1]\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2 \qquad (2.8b) \\ \mathbf{C}[k+1] &= \underset{\mathbf{C}}{\arg\min}\ (\rho/2)\|\mathbf{A}[k+1]-\mathbf{C}\|_F^2 \\ &\quad + \ \text{Tr}\left(\mathbf{\Gamma}^\top[k](\mathbf{A}[k+1]-\mathbf{C})\right) + \lambda_\tau\|\mathbf{C}\|_1 \qquad (2.8c) \\ \mathbf{\Gamma}[k+1] &= \mathbf{\Gamma}[k] + \rho\left(\mathbf{A}[k+1]-\mathbf{C}[k+1]\right). \qquad (2.8d)\end{aligned}$$

As shown next, subproblems (2.8a)-(2.8c) can be solved in closed form and the resulting algorithm provably converges to global optima for each t [38]. Let $(\mathbf{y}_i^\tau)^\top$ and $\mathbf{x}_i^\top$ denote row i of $\mathbf{Y}^\tau$ and $\mathbf{X}$, respectively; while $\mathbf{a}_{-i}^\top$ denotes the $1\times(N-1)$ vector obtained by removing entry i from row i of $\mathbf{A}$, and likewise $\mathbf{Y}_{-i}^\tau$ is the $(N-1)\times C$ matrix obtained by removing row i from $\mathbf{Y}^\tau$. At time

interval t, consider the data-related exponentially-weighted moving averages (EWMAs)

$$\mathbf{\Sigma}^t := \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{Y}^\tau (\mathbf{Y}^\tau)^\top, \quad \boldsymbol{\sigma}_i^t := \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{Y}^\tau \mathbf{y}_i^\tau, \quad \bar{\mathbf{Y}}^t := \sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{Y}^\tau.$$

In addition, let $(\bar{\mathbf{y}}_i^t)^\top$ denote the i-th row of $\bar{\mathbf{Y}}^t$, and $\bar{\mathbf{Y}}_{-i}^t$ the $(N-1) \times C$ matrix obtained by removing row i from $\bar{\mathbf{Y}}^t$. Further, let $\mathbf{\Sigma}_{-i}^t$ denote the $(N-1) \times (N-1)$ matrix obtained by removing the i-th row and i-th column from $\mathbf{\Sigma}^t$. Turning attention to solving for $\mathbf{A}[k+1]$, note that (2.8a) can be recast as

$$\mathbf{A}[k+1] = \underset{\mathbf{a}_1,\ldots,\mathbf{a}_N}{\arg\min} \sum_{i=1}^{N} \left[(1/2)\mathbf{a}_i^\top (\mathbf{\Sigma}^t + \rho \mathbf{I}_N) \mathbf{a}_i - \mathbf{a}_i^\top (\rho \mathbf{c}_i[k] + \boldsymbol{\sigma}_i^t - \boldsymbol{\gamma}_i[k] - b_{ii}[k] \bar{\mathbf{Y}}^t \mathbf{x}_i) \right] \quad (2.9)$$

where constant terms with respect to $\mathbf{A}$ have been dropped, and $\mathbf{a}_i^\top$, $\mathbf{c}_i^\top[k]$, and $\boldsymbol{\gamma}_i^\top[k]$ correspond to row i of $\mathbf{A}$, $\mathbf{C}[k]$, and $\mathbf{\Gamma}[k]$ respectively. The EWMAs $\mathbf{\Sigma}^t$, $\boldsymbol{\sigma}_i^t$, and $\bar{\mathbf{Y}}^t$ can be recursively updated as follows:

$$\mathbf{\Sigma}^t = \beta \mathbf{\Sigma}^{t-1} + \mathbf{Y}^t (\mathbf{Y}^t)^\top \quad (2.10a)$$

$$\boldsymbol{\sigma}_i^t = \beta \boldsymbol{\sigma}_i^{t-1} + \mathbf{Y}^t \mathbf{y}_i^t \quad (2.10b)$$

$$\bar{\mathbf{Y}}^t = \beta \bar{\mathbf{Y}}^{t-1} + \mathbf{Y}^t. \quad (2.10c)$$

The quadratic cost in (2.9) decouples across rows of $\mathbf{A}$, and can be efficiently solved per row in parallel and in closed form. For row i, the constraint $a_{ii} = 0$ is incorporated by solving

$$\mathbf{a}_{-i}[k+1] = \underset{\mathbf{a}_{-i}}{\arg\min} \ (1/2)\mathbf{a}_{-i}^\top (\mathbf{\Sigma}_{-i}^t + \rho \mathbf{I}_{N-1}) \mathbf{a}_{-i} - \mathbf{a}_{-i}^\top \mathbf{w}_{-i}^t \quad (2.11)$$

where $\mathbf{w}_{-i}^t$ is obtained by removing entry i from $\rho \mathbf{c}_i[k] + \boldsymbol{\sigma}_i^t - \boldsymbol{\gamma}_i[k] - b_{ii}[k] \bar{\mathbf{Y}}^t \mathbf{x}_i$. Solving (2.11) yields

$$\mathbf{a}_{-i}[k+1] = (\mathbf{\Sigma}_{-i}^t + \rho \mathbf{I}_{N-1})^{-1} \mathbf{w}_{-i}^t \quad (2.12)$$

and row i of $\mathbf{A}[k+1]$ is updated by zero-padding $\mathbf{a}_{-i}[k+1]$, i.e.,

$$\mathbf{a}_i^\top[k+1] = [a_{-i,i1}[k+1], \ldots, a_{-i,ii-1}[k+1], 0, a_{-i,ii}[k+1], \ldots, a_{-i,iN-1}[k+1]]. \quad (2.13)$$

Next, setting $b_{ij} = 0$ for all off-diagonal entries of $\mathbf{B}$, it turns out that (2.8b) amounts to solving N scalar problems

$$\underset{b_{11},\ldots,b_{NN}}{\arg\min} \sum_{i=1}^{N} \sum_{\tau=1}^{t} \beta^{t-\tau} \left[(1/2) b_{ii}^2 \mathbf{x}_i^\top \mathbf{x}_i - b_{ii} (\mathbf{y}_i^\tau)^\top \mathbf{x}_i + b_{ii} \mathbf{a}_i[k+1]^\top \mathbf{Y}^\tau \mathbf{x}_i \right] \quad (2.14)$$

which yields the per-entry closed-form solution

$$b_{ii}[k+1] = \frac{(\bar{\mathbf{y}}_i^t)^\top \mathbf{x}_i - \mathbf{a}_i[k+1]^\top \bar{\mathbf{Y}}^t \mathbf{x}_i}{\mu^t \mathbf{x}_i^\top \mathbf{x}_i} \tag{2.15}$$

where $\mu^t := (1-\beta^t)/(1-\beta)$.

To solve (2.8c) it is prudent to rewrite the cost function in terms of the rows of matrices $\mathbf{A}[k+1]$, $\mathbf{\Gamma}[k]$ and $\mathbf{C}$, leading to

$$\underset{\mathbf{c}_1,\ldots,\mathbf{c}_N}{\arg\min} \sum_{i=1}^{N} \left[(\rho/2)\|\mathbf{a}_i[k+1] - \mathbf{c}_i\|_2^2 - \mathbf{c}_i^\top \boldsymbol{\gamma}_i[k] + \lambda_t \|\mathbf{c}_i\|_1 \right]. \tag{2.16}$$

Upon defining $\boldsymbol{\alpha}_i[k+1] := \mathbf{a}_i[k+1] + (1/\rho)\boldsymbol{\gamma}_i[k]$ and completing the square in (2.16), the update step for matrix $\mathbf{C}$ can be cast as

$$\underset{\mathbf{c}_1,\ldots,\mathbf{c}_N}{\arg\min} \sum_{i=1}^{N} \left[\frac{1}{2}\|\boldsymbol{\alpha}_i[k+1] - \mathbf{c}_i\|_2^2 + (\lambda_t/\rho)\|\mathbf{c}_i\|_1 \right] \tag{2.17}$$

which amounts to solving

$$\underset{\mathbf{c}_i}{\arg\min} \quad (1/2)\|\boldsymbol{\alpha}_i, [k+1] - \mathbf{c}_i\|_2^2 + (\lambda_t/\rho)\|\mathbf{c}_i\|_1 \tag{2.18}$$

per row i. The Lasso problem (2.18) admits a closed-form solution, in terms of the *soft-thresholding* operator:

$$\mathbf{c}_i[k+1] := \text{soft}(\boldsymbol{\alpha}_i[k+1], \lambda_\tau/\rho) \tag{2.19}$$

whose j-th entry is

$$c_{ij}[k+1] = |\alpha_{ij}[k+1]| - \frac{\lambda_t}{\rho}\text{sign}(\alpha_{ij}[k+1]) \quad \text{if} \quad |\alpha_{ij}[k+1]| > \frac{\lambda_t}{\rho} \tag{2.20}$$

otherwise $c_{ij}[k+1] = 0$. Algorithm 1 summarizes the steps outlined in this section for tracking the dynamic network topology.

Online operation in delay-sensitive applications may not tolerate running multiple inner ADMM iterations per time interval, while the matrix inversions in (2.12) incur $\mathcal{O}(N^3)$ complexity except when $\beta = 1$. These considerations motivate the following potential improvements to Algorithm 2:

I1. Single-iteration ADMM, i.e., drop the **repeat** loop so that lines $6-16$ in Algorithm 1 are run once per time interval; and

Algorithm 1 ADMM solver for topology tracking

Input: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, ϵ, β, ρ
Initialize $\mathbf{\Sigma}^0$, $\bar{\mathbf{Y}}^0$, $\mathbf{B}^0$, $\mathbf{\Gamma}^0$, $\mathbf{C}^0$, μ^0, λ_0
for $t = 1, \ldots, T$ **do**
  Initialize $k = 0$
  Update $\mathbf{\Sigma}^t$, $\bar{\mathbf{Y}}^t$, λ_t, μ^t
  repeat
    for $i = 1 \ldots N$ **do**
      Compute $\mathbf{w}_{-i}^t$, $\mathbf{\Sigma}_{-i}^t$, $\boldsymbol{\sigma}_i^t$
      Compute $\mathbf{a}_{-i}[k+1] = (\mathbf{\Sigma}_{-i}^t + \rho \mathbf{I}_{N-1})^{-1} \mathbf{w}_{-i}^t$
      Update $\mathbf{a}_i[k+1]$ via (2.13)
      Update $b_{ii}[k+1]$ via (2.15)
      Update $\mathbf{c}_i[k+1]$ via (2.19)
    end for
    $\mathbf{\Gamma}[k+1] = \mathbf{\Gamma}[k] + \rho(\mathbf{A}[k+1] - \mathbf{C}[k+1])$
    $k = k + 1$
  until $\|\mathbf{A}[k+1] - \mathbf{C}[k+1]\|_F \leq \epsilon$
  Return $\mathbf{A}^t = \mathbf{A}[k]$, $\mathbf{B}^t = \mathbf{B}[k]$
end for

I2. Leveraging the strict convexity of (2.6) with respect to $\mathbf{A}$, adopt the alternating-minimization algorithm (AMA) [124], which amounts to minimizing the *ordinary* Lagrangian with respect to $\mathbf{A}$ [instead of the augmented Lagrangian in (2.8a)]. This leads to updates $\mathbf{a}_{-i}[k+1] = (\mathbf{\Sigma}_{-i}^t)^{-1} \mathbf{w}_{-i}^t$, which can be recursively obtained with complexity $\mathcal{O}(N^2)$ using the matrix inversion lemma.

Recent advances in proximal optimization methods have led to development of efficient first-order approaches, that are very attractive for big data settings; see [108] for a comprehensive tutorial treatment. In the sequel, this chapter leverages the framework of *proximal gradient* methods to develop provably convergent, first order algorithms that do not involve costly matrix inversions.

2.3.3 Proximal gradient algorithm

Proximal gradient (PG) methods have been popularized for ℓ_1-norm regularized linear regression problems, through the class of iterative shrinkage-thresholding algorithms (ISTA); see e.g., [50, 132]. The main advantage of ISTA over off-the-shelf interior point methods is its computational simplicity. Iterations boil down to matrix-vector multiplications involving the regression matrix, followed by a soft-thresholding operation [64, p. 93].

In this section, an ISTA algorithm is developed for the sparsity regularized dynamic SEM formulation (2.5) at time t. Based on this module, a (pseudo) real-time algorithm for tracking the dynamically-evolving network topology over the horizon $t = 1, \ldots, T$ is obtained as well. The resulting algorithm's memory storage requirement and computational cost per data sample $\{\mathbf{Y}^t, \mathbf{X}\}$ does not grow with t.

Solving (2.5) for a single time interval t. Introducing the optimization variable $\mathbf{V} := [\mathbf{A}\ \mathbf{B}]$, observe that the gradient of $f(\mathbf{V}) := \frac{1}{2}\sum_{\tau=1}^{t} \beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2$ is Lipschitz continuous with a (minimum) Lipschitz constant $L_f = \lambda_{\max}(\sum_{\tau=1}^{t} \beta^{t-\tau}[(\mathbf{Y}^\tau)^\top\ (\mathbf{X})^\top]^\top[(\mathbf{Y}^\tau)^\top\ (\mathbf{X})^\top])$, i.e., $\|\nabla f(\mathbf{V}_1) - \nabla f(\mathbf{V}_2)\| \leq L_f\|\mathbf{V}_1 - \mathbf{V}_2\|, \forall\ \mathbf{V}_1,\ \mathbf{V}_2$ in the domain of f. The Lipschitz constant is time varying, but the dependency on t is kept implicit for notational convenience. On the other hand, the regularizer $g(\mathbf{V}) := \lambda_t\|\mathbf{A}\|_1$ is non-smooth. Instead of directly optimizing the cost in (2.5), PG algorithms minimize a sequence of overestimators evaluated at judiciously chosen points $\mathbf{U}$ (typically the current iterate, or a linear combination of the two previous iterates as discussed in Section 2.4.1). From the Lipschitz continuity of ∇f, for any $\mathbf{V}$ and $\mathbf{U}$ in the domain of f, it holds that $f(\mathbf{V}) \leq Q_f(\mathbf{U}, \mathbf{V}) := f(\mathbf{U}) + \langle \nabla f(\mathbf{U}), \mathbf{V} - \mathbf{U}\rangle + (L_f/2)\|\mathbf{V} - \mathbf{U}\|_F^2$. Next, form the quadratic approximation of the cost $f(\mathbf{V}) + g(\mathbf{V})$ [cf. (2.5)] at a given point $\mathbf{U}$

$$\begin{aligned} Q(\mathbf{V}, \mathbf{U}) &:= Q_f(\mathbf{V}, \mathbf{U}) + g(\mathbf{V}) \\ &= \frac{L_f}{2}\|\mathbf{V} - \mathbf{G}(\mathbf{U})\|_F^2 + g(\mathbf{V}) + f(\mathbf{U}) - \frac{\|\nabla f(\mathbf{U})\|_F^2}{2L_f} \end{aligned} \tag{2.21}$$

where $\mathbf{G}(\mathbf{U}) := \mathbf{U} - (1/L_f)\nabla f(\mathbf{U})$, and clearly $f(\mathbf{V}) + g(\mathbf{V}) \leq Q(\mathbf{V}, \mathbf{U})$ for any $\mathbf{V}$ and $\mathbf{U}$. Note that $\mathbf{G}(\mathbf{U})$ corresponds to a gradient-descent step taken from $\mathbf{U}$, with step-size equal to $1/L_f$.

With $k = 1, 2, \ldots$ denoting iterations, PG algorithms set $\mathbf{U} := \mathbf{V}[k-1]$ and generate the

following sequence of iterates

$$\begin{aligned}\mathbf{V}[k] :=& \arg\min_{\mathbf{V}} Q(\mathbf{V}, \mathbf{V}[k-1]) \\ =& \arg\min_{\mathbf{V}} \left\{ \frac{L_f}{2}\|\mathbf{V} - \mathbf{G}(\mathbf{V}[k-1])\|_F^2 + g(\mathbf{V}) \right\} \end{aligned} \tag{2.22}$$

where the second equality follows from the fact that the last two summands in (2.21) do not depend on $\mathbf{V}$. The optimization problem (2.22) is known as the *proximal operator* of the function g/L_f evaluated at $\mathbf{G}(\mathbf{V}[k-1])$, and is denoted as $\text{prox}_{g/L_f}(\mathbf{G}(\mathbf{V}[k-1]))$. Henceforth adopting the notation $\mathbf{G}[k-1] := \mathbf{G}(\mathbf{V}[k-1])$ for convenience, the PG iterations can be compactly rewritten as

$$\mathbf{V}[k] = \text{prox}_{g/L_f}(\mathbf{G}[k-1]). \tag{2.23}$$

A key element to the success of PG algorithms stems from the possibility of efficiently solving the sequence of subproblems(2.22), i.e., evaluating the proximal operator.

Upon computing the gradients (see Appendix A for the detailed derivation)

$$\nabla_{\mathbf{a}_{-i}} f[k] = \mathbf{\Sigma}_{-i}^t \mathbf{a}_{-i}[k] + \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i b_{ii}[k] - \boldsymbol{\sigma}_{-i}^t \tag{2.24}$$

$$\nabla_{b_{ii}} f[k] = \mathbf{a}_{-i}^\top[k] \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i + \frac{(1-\beta^t)}{1-\beta} b_{ii}[k] \|\mathbf{x}_i\|_2^2 - (\bar{\mathbf{y}}_i^t)^\top \mathbf{x}_i \tag{2.25}$$

it turns out that the parallel ISTA iterations are

$$\mathbf{a}_{-i}[k+1] = \mathcal{S}_{\lambda_t/L_f}\left(\mathbf{a}_{-i}[k] - (1/L_f)\nabla_{\mathbf{a}_{-i}} f[k]\right) \tag{2.26}$$

$$b_{ii}[k+1] = b_{ii}[k] - (1/L_f)\nabla_{b_{ii}} f[k] \tag{2.27}$$

where $\text{soft}_\mu(\mathbf{M})$ with entry (i,j) given by $\text{sign}(m_{ij})\max(|m_{ij}| - \mu, 0)$ denotes the soft-thresholding operator. The iterations are provably convergent to the globally optimal solution $\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\}$ of (2.5), as per the general convergence results available for PG methods and ISTA in particular [50, 108].

Computation of the gradients in (2.24)-(2.25) requires one matrix-vector mutiplication by $\mathbf{\Sigma}_{-i}^t$ and one by $\bar{\mathbf{Y}}_{-i}^t$, in addition to three vector inner-products, plus a few (negligibly complex) scalar and vector additions. Both the update of $b_{ii}[k+1]$ as well as the soft-thresholding operation in (2.26) entail negligible computational complexity. All in all, the simplicity of the resulting iterations

should be apparent. Per iteration, the actual rows of the adjacency matrix are obtained by zero-padding the updated $\mathbf{a}_{-i}[k]$ (see (2.13)). This way, the desired SEM parameter estimates at time t are given by $\hat{\mathbf{A}}^t = [\mathbf{a}_1^\top[k], \ldots, \mathbf{a}_N^\top[k]]^\top$ and $\hat{\mathbf{B}}^t = \text{diag}(b_{11}[k], \ldots, b_{NN}[k])$, for k large enough so that convergence has been attained.

Remark 3 (General sparsity-promoting regularization)**.** Beyond $g(\mathbf{A}) = \lambda_t \|\mathbf{A}\|_1$, the algorithmic framework here can accommodate more general *structured sparsity*-promoting regularizers $\gamma(\mathbf{A})$ as long as the resulting proximal operator $\text{prox}_{\gamma/L_f}(\cdot)$ is given in terms of scalar or (and) vector soft-thresholding operators. In addition to the ℓ_1-norm (Lasso penalty), this holds e.g., for the sum of the ℓ_2-norms of vectors with groups of non-overlapping entries of $\mathbf{A}$ (group Lasso penalty [136]), or, a linear combination of the aforementioned two – the so-termed hierarchical Lasso penalty that encourages sparsity across and within the groups defined over $\mathbf{A}$ [122]. These types of regularization could be useful if one e.g., has a priori knowledge that some clusters of nodes are more likely to be jointly (in)active [112].

Solving (2.5) over the entire time horizon $t = 1, \ldots, T$. To track the dynamically-evolving network topology, one can go ahead and solve (2.5) sequentially for each $t = 1, \ldots, T$ as data arrive, using (2.24)-(2.27). (The procedure can also be adopted in a batch setting, when all $\{\mathbf{Y}^t\}_{t=1}^T$ are available in memory.) Because the network is assumed to vary slowly across time intervals, it is convenient to warm-restart the ISTA iterations, that is, at time t initialize $\{\mathbf{A}[0], \mathbf{B}[0]\}$ with the previous solution $\{\hat{\mathbf{A}}^{t-1}, \hat{\mathbf{B}}^{t-1}\}$. Since the sought estimates are expected to be close to the initial points, one expects convergence to be attained after few iterations.

To obtain the new SEM parameter estimates via (2.24)-(2.27), it suffices to update (possibly) λ_t and the Lipschitz constant L_f, as well as the data-dependent EWMAs $\mathbf{\Sigma}^t$ ($\boldsymbol{\sigma}_i^t$ is the i-th column of $\mathbf{\Sigma}^t$), and $\bar{\mathbf{Y}}^t$. Interestingly, the potential growing-memory problem in storing the entire history of data $\{\mathbf{Y}^t\}_{t=1}^T$ can be avoided by performing the recursive updates

$$\mathbf{\Sigma}^t = \beta \mathbf{\Sigma}^{t-1} + \mathbf{Y}^t (\mathbf{Y}^t)^\top \tag{2.28}$$

$$\bar{\mathbf{Y}}^t = \beta \bar{\mathbf{Y}}^{t-1} + \mathbf{Y}^t. \tag{2.29}$$

Note that the complexity in evaluating the Gram matrix $\mathbf{Y}^t(\mathbf{Y}^t)^\top$ dominates the per-iteration computational cost of the algorithm. To circumvent the need of recomputing the Lipschitz constant

Algorithm 2 Pseudo real-time ISTA for topology tracking

Require: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, β.

Initialize $\hat{\mathbf{A}}^0 = \mathbf{0}_{N\times N}, \hat{\mathbf{B}}^0 = \mathbf{\Sigma}^0 = \mathbf{I}_N, \bar{\mathbf{Y}}^0 = \mathbf{0}_{N\times C}, \lambda_0$.
for $t = 1, \ldots, T$ **do**
  Update λ_t, L_f and $\mathbf{\Sigma}^t$, $\bar{\mathbf{Y}}^t$ via (2.28)-(2.29).
  Initialize $\mathbf{A}[0] = \hat{\mathbf{A}}^{t-1}$, $\mathbf{B}[0] = \hat{\mathbf{B}}^{t-1}$, and set $k = 0$.
  while not converged **do**
    for $i = 1 \ldots N$ (in parallel) **do**
      Compute $\mathbf{\Sigma}^t_{-i}$ and $\bar{\mathbf{Y}}^t_{-i}$.
      Form gradients at $\mathbf{a}_{-i}[k]$ and $b_{ii}[k]$ via (2.24)-(2.25).
      Update $\mathbf{a}_{-i}[k+1]$ via (2.26).
      Update $b_{ii}[k+1]$ via (2.27).
      Update $\mathbf{a}_i[k+1]$ via (2.13).
    end for
    $k = k + 1$.
  end while
  return $\hat{\mathbf{A}}^t = \mathbf{A}[k]$, $\hat{\mathbf{B}}^t = \mathbf{B}[k]$.
end for

per time interval (that in this case entails finding the spectral radius of a data-dependent matrix), the step-size $1/L_f$ in (2.26)-(2.27) can be selected by a line search [108]. One possible choice is the backtracking step-size rule in [34], under which convergence of (8)-(2.27) to $\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\}$ can be established as well.

Algorithm 2 summarizes the steps outlined in this section for tracking the dynamic network topology, given temporal traces of infection events $\{\mathbf{Y}^t\}_{t=1}^T$ and susceptibilities $\mathbf{X}$. It is termed *pseudo real-time* ISTA, since in principle one needs to run multiple (inner) ISTA iterations till convergence per time interval $t = 1, \ldots, T$. This will in turn incur an associated delay, that may (or may not) be tolerable depending on the specific network inference problem at hand. Nevertheless, numerical tests indicate that in practice $5 - 10$ inner iterations suffice for convergence; see also Fig.

2.2 and the discussion in Section 2.4.2.

2.4 Algorithmic Enhancements and Variants

This section deals with various improvements to Algorithm 1, that pertain to accelerating its rate of convergence and also adapting it for real-time operation in time-sensitive applications. In addition, a stochastic-gradient algorithm useful when minimal computational complexity is at a premium is also outlined.

2.4.1 Accelerated proximal gradient method and fast ISTA

In the context of sparsity-regularized inverse problems and general non-smooth optimization, there have been several recent efforts towards improving the sublinear global rate of convergence exhibited by PG algorithms such as ISTA; see e.g., [34,98,99] and references therein. Since for large-scale problems first-order (gradient) methods are in many cases the only admissible alternative, the goal of these works has been to retain the computational simplicity of ISTA while markedly enhancing its global rate of convergence.

Remarkable results in [99] assert that convergence speedups can be obtained through the so-termed *accelerated* (A)PG algorithm. Going back to the derivations in the beginning of Section 2.4.1, APG algorithms generate the following sequence of iterates [cf. (2.22) and (2.23)]

$$\mathbf{V}[k] = \arg\min_{\mathbf{V}} Q(\mathbf{V}, \mathbf{U}[k-1]) = \text{prox}_{g/L_f}(\mathbf{G}(\mathbf{U}[k-1]))$$

where

$$\mathbf{U}[k] := \mathbf{V}[k-1] + \left(\frac{c[k-1]-1}{c[k]}\right)(\mathbf{V}[k-1] - \mathbf{V}[k-2]) \tag{2.30}$$

$$c[k] = \frac{1+\sqrt{4c^2[k-1]+1}}{2}. \tag{2.31}$$

In words, instead of minimizing a quadratic approximation to the cost evaluated at $\mathbf{V}[k-1]$ as in ISTA [cf. (2.22)], the accelerated PG algorithm [a.k.a. fast (F)ISTA] utilizes a linear combination of the previous two iterates $\{\mathbf{V}[k-1], \mathbf{V}[k-2]\}$. The iteration-dependent combination weights are function of the scalar sequence (2.31). FISTA offers quantifiable iteration complexity, namely

a (worst-case) convergence rate guarantee of $\mathcal{O}(1/\sqrt{\epsilon})$ iterations to return an ϵ-optimal solution measured by its objective value (ISTA instead offers $\mathcal{O}(1/\epsilon)$) [34, 99]. Even for general (non-)smooth optimization, APG algorithms have been shown to be optimal within the class of first-order (gradient) methods, in the sense that the aforementioned worst-case convergence rate cannot be improved [98, 99].

The FISTA solver for (2.5) entails the following steps [cf. (2.24)-(2.27)]

$$\tilde{\mathbf{a}}_{-i}[k] := \mathbf{a}_{-i}[k] + \left(\frac{c[k-1]-1}{c[k]}\right)(\mathbf{a}_{-i}[k] - \mathbf{a}_{-i}[k-1]) \tag{2.32}$$

$$\tilde{b}_{ii}[k] := b_{ii}[k] + \left(\frac{c[k-1]-1}{c[k]}\right)(b_{ii}[k] - b_{ii}[k-1]) \tag{2.33}$$

$$\nabla_{\mathbf{a}_{-i}} f[k] = \mathbf{\Sigma}^t_{-i}\tilde{\mathbf{a}}_{-i}[k] + \bar{\mathbf{Y}}^t_{-i}\mathbf{x}_i\tilde{b}_{ii}[k] - \boldsymbol{\sigma}^t_{-i} \tag{2.34}$$

$$\nabla_{b_{ii}} f[k] = \tilde{\mathbf{a}}^\top_{-i}[k]\bar{\mathbf{Y}}^t_{-i}\mathbf{x}_i + \frac{(1-\beta^t)}{1-\beta}\tilde{b}_{ii}[k]\|\mathbf{x}_i\|_2^2 - (\bar{\mathbf{y}}^t_i)^\top\mathbf{x}_i \tag{2.35}$$

$$\mathbf{a}_{-i}[k+1] = \mathcal{S}_{\lambda_t/L_f}\left(\tilde{\mathbf{a}}_{-i}[k] - (1/L_f)\nabla_{\tilde{\mathbf{a}}_{-i}} f[k]\right) \tag{2.36}$$

$$b_{ii}[k+1] = \tilde{b}_{ii}[k] - (1/L_f)\nabla_{\tilde{b}_{ii}} f[k] \tag{2.37}$$

where $c[k]$ is updated as in (2.31). The overall (pseudo) real-time FISTA for tracking the network topology is tabulated under Algorithm 3. As desired, the computational complexity of Algorithms 2 and 3 is roughly the same. Relative to Algorithm 2, the memory requirements are essentially doubled since one now has to store the two prior estimates of $\mathbf{A}$ and $\mathbf{B}$, which are nevertheless sparse and diagonal matrices, respectively. Numerical tests in Section 2.5 suggest that Algorithm 3 exhibits the best performance when compared to Algorithms 1 and 2, especially when modified to accommodate real-time processing requirements – the subject dealt with next.

2.4.2 Inexact (F)ISTA for time-sensitive operation

Additional challenges arise with real-time data collection, where analytics must often be performed "on-the-fly" as well as without an opportunity to revisit past entries. Online operation in delay-sensitive applications may not tolerate running multiple inner (F)ISTA iterations per time interval, so that convergence is attained for each t as required by Algorithms 2 and 3. This section touches upon an interesting tradeoff that emerges with time-constrained data-intensive problems, where a

Algorithm 3 Pseudo real-time FISTA for topology tracking

Require: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, β.

Initialize $\hat{\mathbf{A}}^0 = \mathbf{0}_{N\times N}, \hat{\mathbf{B}}^0 = \mathbf{\Sigma}^0 = \mathbf{I}_N, \bar{\mathbf{Y}}^0 = \mathbf{0}_{N\times C}, \lambda_0$.
for $t = 1, \ldots, T$ **do**
  Update λ_t, L_f and $\mathbf{\Sigma}^t$, $\bar{\mathbf{Y}}^t$ via (2.28)-(2.29).
  Initialize $\mathbf{A}[0] = \mathbf{A}[-1] = \hat{\mathbf{A}}^{t-1}, \mathbf{B}[0] = \mathbf{B}[-1] = \hat{\mathbf{B}}^{t-1}$, $c[0] = c[-1] = 1$, and set $k = 0$.
  while not converged **do**
    for $i = 1 \ldots N$ (in parallel) **do**
      Compute $\mathbf{\Sigma}^t_{-i}$ and $\bar{\mathbf{Y}}^t_{-i}$.
      Update $\tilde{\mathbf{a}}_{-i}[k]$ and $\tilde{b}_{ii}[k]$ via (2.32)-(2.33).
      Form gradients at $\tilde{\mathbf{a}}_{-i}[k]$ and $\tilde{b}_{ii}[k]$ via (2.34)-(2.35).
      Update $\mathbf{a}_{-i}[k+1]$ via (2.36).
      Update $b_{ii}[k+1]$ via (2.37).
      Update $\mathbf{a}_i[k+1]$ via (2.13).
    end for
    $k = k + 1$.
    Update $c[k]$ via (2.31).
  end while
  return $\hat{\mathbf{A}}^t = \mathbf{A}[k]$, $\hat{\mathbf{B}}^t = \mathbf{B}[k]$.
end for

high-quality answer that is obtained slowly can be less useful than a medium-quality answer that is obtained quickly.

Consider for the sake of exposition a scenario where the underlying network processes are stationary, or just piecewise stationary with sufficiently long coherence time for that matter. The rationale behind the proposed real-time algorithm hinges upon the fact that the solution of (2.5) for each $t = 1, \ldots, T$ does not need to be super accurate in the aforementioned stationary setting, since it is just an intermediate step in the outer loop matched to the time-instants of data acquisition. This motivates stopping earlier the inner iteration which solves (2.5) (cf. the **while** loop in Algorithms

2 and 3), possibly even after a single soft-thresholding step, as detailed in the real-time Algorithm 4. Note that in this case the inner-iteration index k coincides with the time index t. A similar adjustment can be made to the ISTA variant (Algorithm 2), and one can in general adopt a less aggressive approach by allowing a few (not just one) inner-iterations per t.

A convergence proof of Algorithm 4 in a stationary network setting will not be provided here, and is left as a future research direction. For the infinite-memory case [cf. $\beta = 1$ in (2.5)] and the simpler ISTA counterpart of Algorithm 4 obtained when $c[t] = 1,\ \forall t$, it appears possible to adapt the arguments in [91, 94] to establish that the resulting iterations converge to a minimizer of the batch problem (2.4). In the dynamic setting where the network is time-varying, then convergence is not expected to occur because of the continuous network fluctuations. Still, as with adaptive signal processing algorithms [121] one would like to establish that the tracking error attains a bounded steady-state. These interesting and challenging problems are subject of ongoing investigation and will be reported elsewhere.

For synthetically-generated data according to the setup described in Section 2.5.1, Fig. 2.2 shows the time evolution of Algorithm 2's mean-square error (MSE) estimation performance (note that MSE $= \sum_{i,j}(\hat{a}_{ij}^t - a_{ij}^t)^2/N^2$). For each time interval t, (2.5) is solved "inexactly" after running only $k = 1, 5, 10$ and 15 inner iterations. Note that the case $k = 1$ corresponds to Algorithm 4. Certainly $k = 10$ iterations suffice for the FISTA algorithm to converge to the minimizer of (2.5); the curve for $k = 15$ is identical. Even with $k = 5$ the obtained performance is satisfactory for all practical purposes, especially after a short transient where the warm-restarts offer increasingly better initializations. While Algorithm 4 shows a desirable convergent behavior, it seems that this example's network coherence time of $t = 250$ time intervals is too short to be tracked effectively. Still, if the network changes are sufficiently smooth as it occurs at $t = 750$, then the real-time algorithm is able to estimate the network reliably.

A convergence proof of Algorithm 4 in a stationary network setting will not be provided here, and is left as a future research direction. Still, convergence will be demonstrated next with the aid of computer simulations. For the infinite-memory case [cf. $\beta = 1$ in (2.5)] and the simpler ISTA counterpart of Algorithm 4 obtained when $c[t] = 1,\ \forall t$, it appears possible to adapt the arguments in [91, 94] to establish that the resulting iterations converge to a minimizer of the batch problem

Algorithm 4 Real-time inexact FISTA for topology tracking

Require: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, β.

Initialize $\mathbf{A}[1]=\mathbf{A}[0]=\mathbf{0}_{N\times N}, \mathbf{B}[1]=\mathbf{B}[0]=\mathbf{\Sigma}^0=\mathbf{I}_N, \bar{\mathbf{Y}}^0=\mathbf{0}_{N\times C}, c[1]=c[0]=1, \lambda_0$.
for $t=1,\ldots,T$ **do**
  Update λ_t, L_f and $\mathbf{\Sigma}^t$, $\bar{\mathbf{Y}}^t$ via (2.28)-(2.29).
  for $i=1\ldots N$ (in parallel) **do**
    Compute $\mathbf{\Sigma}^t_{-i}$ and $\bar{\mathbf{Y}}^t_{-i}$.
    Update $\tilde{\mathbf{a}}_{-i}[t]$ and $\tilde{b}_{ii}[t]$ via (2.32)-(2.33).
    Form gradients at $\tilde{\mathbf{a}}_{-i}[t]$ and $\tilde{b}_{ii}[t]$ via (2.34)-(2.35).
    Update $\mathbf{a}_{-i}[t+1]$ via (2.36).
    Update $b_{ii}[t+1]$ via (2.37).
    Update $\mathbf{a}_i[t+1]$ via (2.13).
  end for
  Update $c[t+1]$ via (2.31).
  return $\hat{\mathbf{A}}^t=\mathbf{A}[t+1], \hat{\mathbf{B}}^t=\mathbf{B}[t+1]$.
end for

(2.4). In the dynamic setting where the network is time-varying, then convergence is not expected to occur because of the continuous network fluctuations. Still, as with adaptive signal processing algorithms [121] one would like to establish that the tracking error attains a bounded steady-state. These interesting and challenging problems are subject of ongoing investigation and will be reported elsewhere.

For synthetically-generated data according to the setup described in Section 2.5.1, Fig. 2.2 shows the time evolution of Algorithm 2's mean-square error (MSE) estimation performance. For each time interval t, (2.5) is solved "inexactly" after running only $k=1,5,10$ and 15 inner iterations. Note that the case $k=1$ corresponds to Algorithm 4. Certainly $k=10$ iterations suffice for the FISTA algorithm to converge to the minimizer of (2.5); the curve for $k=15$ is identical. Even with $k=5$ the obtained performance is satisfactory for all practical purposes, especially after a short transient where the warm-restarts offer increasingly better initializations. While Algorithm

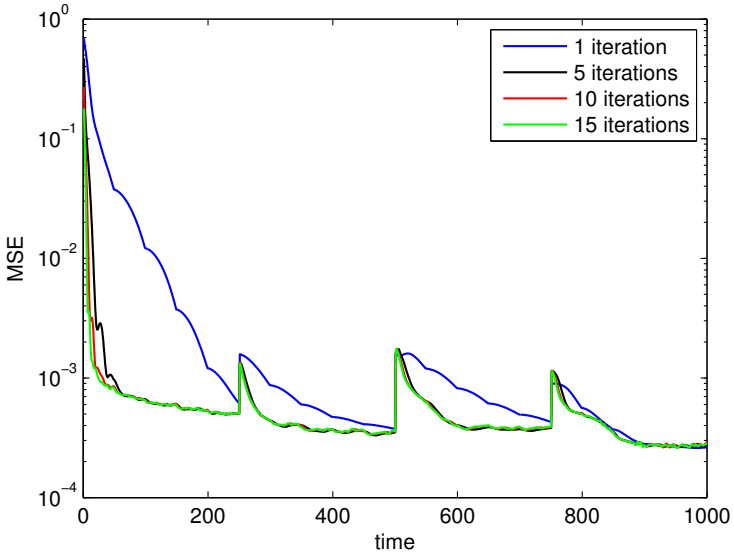


Figure 2.2: MSE performance of Algorithm 3 for $k = 1$ (Algorithm 4), 5, 10, and 15 inner iterations plotted against time.

4 shows a desirable convergent behavior, it seems that this example's network coherence time of $t = 250$ time intervals is too short to be tracked effectively. Still, if the network changes are sufficiently smooth as it occurs at $t = 750$, then the real-time algorithm is able to estimate the network reliably.

2.4.3 Stochastic-gradient descent algorithm

A stochastic gradient descent (SGD) algorithm is developed in this section, which operates in real time and can track the (slowly-varying) underlying network topology. Among all algorithms developed so far, the SGD iterations incur the least computational cost.

Towards obtaining the SGD algorithm, consider $\beta = 0$ in (2.5). The resulting cost function can be expressed as $f_t(\mathbf{V}) + g(\mathbf{V})$, where $\mathbf{V} := [\mathbf{A}\ \mathbf{B}]$ and $f_t(\mathbf{V}) := (1/2)\|\mathbf{Y}^t - \mathbf{A}\mathbf{Y}^t - \mathbf{B}\mathbf{X}\|_F^2$ only accounts for the data acquired at time interval t. Motivated by computational simplicity, the

Algorithm 5 SGD algorithm for topology tracking

Require: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, η.

Initialize $\mathbf{A}[1]=\mathbf{0}_{N\times N}, \mathbf{B}[1]=\mathbf{I}_N, \lambda_1$.
for $t=1,\ldots,T$ **do**
  Update λ_t.
  for $i=1\ldots N$ (in parallel) **do**
    Form gradients at $\mathbf{a}_{-i}[t]$ and $b_{ii}[t]$ via (2.38)-(2.39).
    Update $\mathbf{a}_{-i}[t+1]$ via (2.40).
    Update $b_{ii}[t+1]$ via (2.41).
    Update $\mathbf{a}_i[t+1]$ via (2.13).
  end for
  return $\hat{\mathbf{A}}^t=\mathbf{A}[t+1], \hat{\mathbf{B}}^t=\mathbf{B}[t+1]$.
end for

"inexact" gradient descent plus soft-thresholding ISTA iterations yield the following updates

$$\nabla_{\mathbf{a}_{-i}} f_t[t]=\mathbf{Y}^t_{-i}\left((\mathbf{Y}^t_{-i})^\top \mathbf{a}_{-i}[t]+\mathbf{x}_i b_{ii}[t]-\mathbf{y}^t_i\right) \tag{2.38}$$

$$\nabla_{b_{ii}} f_t[t]=\mathbf{a}^\top_{-i}[t]\mathbf{Y}^t_{-i}\mathbf{x}_i+b_{ii}[t]\|\mathbf{x}_i\|^2-(\mathbf{y}^t_i)^\top\mathbf{x}_i \tag{2.39}$$

$$\mathbf{a}_{-i}[t+1]=\mathcal{S}_{\lambda_t/\eta}\left(\mathbf{a}_{-i}[t]-\eta\nabla_{\mathbf{a}_{-i}} f_t[t]\right) \tag{2.40}$$

$$b_{ii}[t+1]=b_{ii}[t]-\eta\nabla_{b_{ii}} f_t[t]. \tag{2.41}$$

Compared to the parallel ISTA iterations in Algorithm 2 [cf. (2.24)-(2.26)], three main differences are noteworthy: (i) iterations k are merged with the time intervals t of data acquisition; (ii) the stochastic gradients $\nabla_{\mathbf{a}_{-i}} f_t[t]$ and $\nabla_{b_{ii}} f_t[t]$ involve the (noisy) data $\{\mathbf{Y}^t(\mathbf{Y}^t)^\top, \mathbf{Y}^t\}$ instead of their time-averaged counterparts $\{\mathbf{\Sigma}^t, \bar{\mathbf{Y}}^t\}$; and (iii) a generic constant step-size η is utilized for the gradient descent steps.

The overall SGD algorithm is tabulated under Algorithm 5. Forming the gradients in (2.38)-(2.39) requires one matrix-vector mutiplication by $(\mathbf{Y}^t_{-i})^\top$ and two by $\mathbf{Y}^t_{-i}$. These multiplications dominate the per-iteration computational complexity of Algorithm 5, justifying its promised simplicity. Accelerated versions could be developed as well, at the expense of marginal increase in

computational complexity and doubling the memory requirements.

To gain further intuition on the SGD algorithm developed, consider the online learning paradigm under which the network topology inference problem is to minimize the expected cost $E[f_t(\mathbf{V}) + g(\mathbf{V})]$ (subject to the usual constraints on $\mathbf{V} = [\mathbf{A}\ \mathbf{B}]$). The expectation is taken w.r.t. the *unknown* probability distribution of the data. In lieu of the expectation, the approach taken throughout this paper is to minimize the empirical cost $C^T(\mathbf{V}) := (1/T)[\sum_{t=1}^T f_t(\mathbf{V}) + g(\mathbf{V})]$. Note that for $\beta = 1$, the minimizers of $C^T(\mathbf{V})$ coincide with (2.4) since the scaling by $1/T$ does not affect the optimal solution. For $\beta < 1$, the cost $C_\beta^T(\mathbf{V}) := \sum_{t=1}^T \beta^{T-t} f_t(\mathbf{V}) + g(\mathbf{V})$ implements an EWMA which "forgets" past data and allows tracking. In all cases, the rationale is that by virtue of the law of large numbers, if data $\{\mathbf{Y}^t\}_{t=1}^T$ are stationary, solving $\lim_{T\to\infty} \min_{\mathbf{V}} C^T(\mathbf{V})$ yields the desired solution to the expected cost.

A different approach to achieve this same goal – typically with reduced computational complexity – is to drop the expectation (or the sample averaging operator for that matter), and update the estimates via a stochastic (sub)gradient iteration $\mathbf{V}(t) = \mathbf{V}(t-1) - \eta \partial\{f_t(\mathbf{V}) + g(\mathbf{V})\}|_{\mathbf{V}=\mathbf{V}[t-1]}$. The subgradients with respect to $\mathbf{a}_{-i}$ are

$$\partial_{\mathbf{a}_{-i}} f_t[t] = \mathbf{Y}_{-i}^t \left((\mathbf{Y}_{-i}^t)^\top \mathbf{a}_{-i}[t] + \mathbf{x}_i b_{ii}[t] - \mathbf{y}_i^t \right) + \lambda_t \text{sign}(\mathbf{a}_{-i}[t]) \tag{2.42}$$

so the resulting algorithm has the drawback of (in general) not providing sparse solutions per iteration; see also [46] for a sparse least-mean squares (LMS) algorithm. For that reason, the approach here is to adopt the proximal gradient (ISTA) formalism to tackle the minimization of the instantaneous costs $f_t(\mathbf{V}) + g(\mathbf{V})$, and yield sparsity-inducing soft-thresholded updates (2.40). Also acknowledging the limitation of subgradient methods to yield sparse solutions, related "truncated gradient" updates were advocated for sparse online learning in [81].

2.4.4 Choice of algorithm

In order to track the topology of a network from cascade data, one is faced with making a choice among the developed algorithms. Algorithm 3 enjoys a theoretically-proven faster convergence than Algorithm 2, and it is recommended in situations where real-time operation and computational cost are not critical e.g., when C, N are in the range of hundreds or thousands. Algorithm 4 is a

more practical alternative to Algorithm 3 in real-time topology tracking applications. Algorithm 5 is the most lightweight and scalable option for real-time settings, provided a slightly degraded error performance is affordable. Matlab implementation of the algorithms tested in the following section assumes that the network data can be stored in main memory, and does not exploit the parallel structure of the update recursions.

2.5 Numerical Experiments

Performance of the proposed algorithms is assessed in this section via computer simulations using both synthetically-generated network data, and real traces of information cascades collected from the web [86].

2.5.1 Tests on synthetic data

Data generation. Numerical tests on synthetic network data are conducted here to evaluate the tracking ability and compare Algorithms 2-5. From a "seed graph" with adjacency matrix

$$\mathbf{M} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

a Kronecker graph of size $N = 64$ nodes was generated as described in [84]. The resulting nonzero edge weights of $\mathbf{A}^t$ were allowed to vary over $T = 1,000$ intervals under 3 settings: i) i.i.d. Bernoulli(0.5) random variables; ii) random selection of the edge-evolution pattern uniformly from a set of four smooth functions: $a_{ij}(t) = 0.5 + 0.5\text{sin}(0.1t)$, $a_{ij}(t) = 0.5 + 0.5\text{cos}(0.1t)$, $a_{ij}(t) = e^{-0.01t}$, and $a_{ij}(t) = 0$; and iii) random selection of the edge-evolution pattern uniformly from a set of four nonsmooth functions shown in Fig. 2.3.

The number of contagions was set to $C = 80$, and $\mathbf{X}$ was formed with i.i.d. entries uniformly distributed over $[0, 3]$. Matrix $\mathbf{B}^t$ was set to diag$(\mathbf{b}^t)$, where $\mathbf{b}^t \in \mathbb{R}^N$ is a standard Gaussian random vector. During time interval t, infection times were generated synthetically as $\mathbf{Y}^t = (\mathbf{I}_N - \mathbf{A}^t)^{-1}(\mathbf{B}^t\mathbf{X} + \mathbf{E}^t)$, where $\mathbf{E}^t$ is a standard Gaussian random matrix.

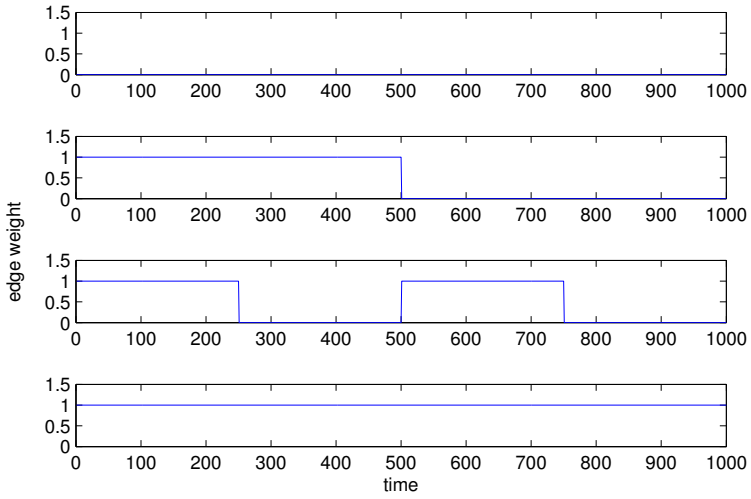


Figure 2.3: Nonsmooth variation of synthetically-generated edge weights of the time-varying network.

Performance evaluation. With $\beta = 0.98$, Algorithm 2 was run after initializing the relevant variables as described in the algorithm table (cf. Section 2.3.3), and setting $\lambda_0 = 25$. In addition, $\lambda_t = \lambda_0$ for $t = 1, \ldots, T$ as discussed in Remark 2. Fig. 2.4 shows the evolution of the mean-square error (MSE), $\sum_{i,j}(\hat{a}_{ij}^t - a_{ij}^t)^2/N^2$. As expected, the best performance was obtained when the temporal evolution of edges followed smooth functions. Even though the Bernoulli evolution of edges resulted in the highest MSE, Algorithm 1 still tracked the underlying topology with reasonable accuracy as depicted in the heat maps of the inferred adjacency matrices; see Fig. 2.5.

Selection of a number of parameters is critical to the performance of the developed algorithms. In order to evaluate the effect of each parameter on the network estimates, several tests were conducted by tracking the non-smooth network evolution using Algorithm 3 with varying parameter values. To illustrate the importance of leveraging sparsity of the edge weights, Fig. 2.6 depicts heatmaps of the adjacency matrices inferred at $t = 900$, with λ set to $0, 50$, and 100 for all time intervals. Comparisons with the actual adjacency matrix reveal that increasing λ progressively refines the network estimates by driving erroneously detected nonzero edge weights to 0. Indeed, the value $\lambda = 100$ in this case appears to be just about right, while smaller values markedly overestimate the support set associated with the edges present in the actual network.

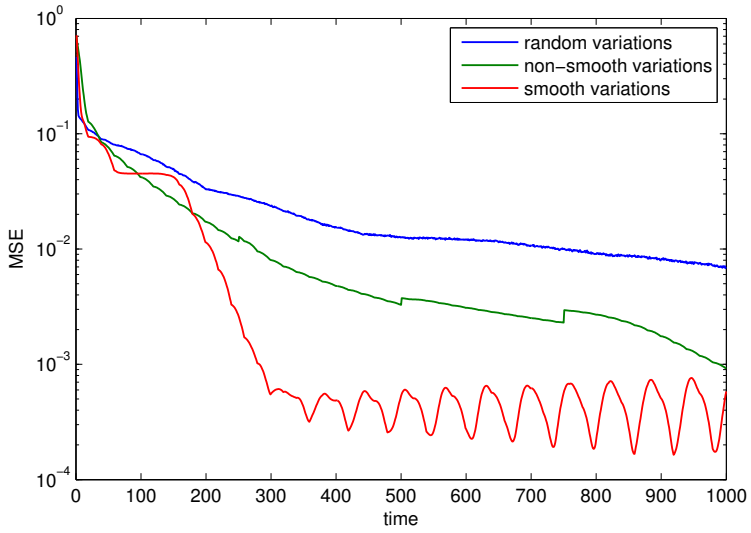


Figure 2.4: MSE versus time obtained using pseudo real-time ISTA (Algorithm 2), for different edge evolution patterns.

Fig. 2.7 compares the MSE performance of Algorithm 3 for $\beta \in \{0.999, 0.990, 0.900, 0.750\}$. As expected, the MSE associated with values of β approaching 1 degrades more dramatically when changes occur within the network (at time intervals $t = 250$, $t = 500$, and $t = 750$ in this case; see Fig. 2.3). The MSE spikes observed when $\beta \in \{0.999, 0.990\}$ are a manifestation of the slower rate of adaptation of the algorithm for these values of the forgetting factor. In this experiment, $\beta = 0.990$ outperformed the rest for $t > 500$. In addition, comparisons of the MSE performance in the presence of increasing noise variance are depicted in Figure 2.8. Although the MSE values are comparable during the initial stages of the topology inference process, as expected higher noise levels lead to MSE performance degradation in the long run.

Finally, a comparison of the real-time version of the different algorithms was carried out when tracking the synthetic time-varying network with non-smooth edge variations. Specifically, the real-time (inexact) counterparts of ISTA, FISTA (cf. Algorithm 4), SGD (cf. Algorithm 5), and a suitably modified version of Algorithm 1 were run as suggested in Section 2.4.2, i.e., eliminating the inner **while** loop in Algorithms 2 and 3 so that a single iteration is run per time interval. Fig. 2.9 compares the resulting MSE curves as the error evolves with time, showing that the inexact online FISTA algorithm achieves the best error performance. The MSE performance degradation of

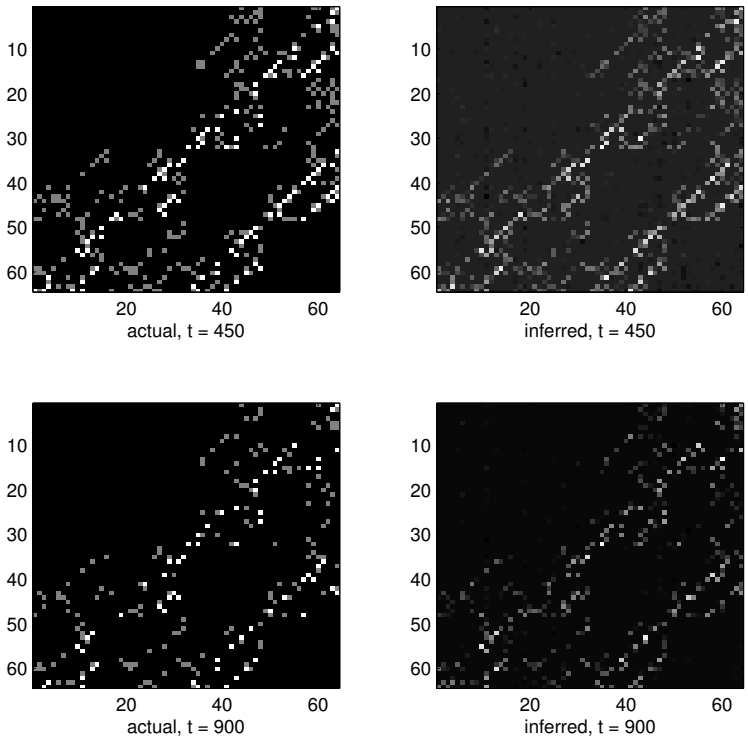


Figure 2.5: Actual adjacency matrix $\mathbf{A}^t$ its estimate $\hat{\mathbf{A}}^t$ obtained using pseudo real-time ISTA (Algorithm 2), at time intervals $t = 450$ and $t = 900$.

Algorithm 4 relative to its (exact) counterpart Algorithm 3 is depicted in Fig. 2.2, as a function of the number of inner iterations k.

2.5.2 Tests on real data

Dataset description. The real data used was collected during a prior study by monitoring blog posts and news articles for memes (popular textual phrases) appearing within a set of over 3.3 million websites [112]. Traces of information cascades were recorded over a period of one year, from March 2011 till February 2012; the data is publicly available from [86]. The time when each website mentioned a specific news item was recorded as a Unix timestamp in hours (i.e., the number of hours since midnight on January 1, 1970). Specific globally-popular topics during this period were identified and cascade data for the top $5,000$ websites that mentioned memes associated

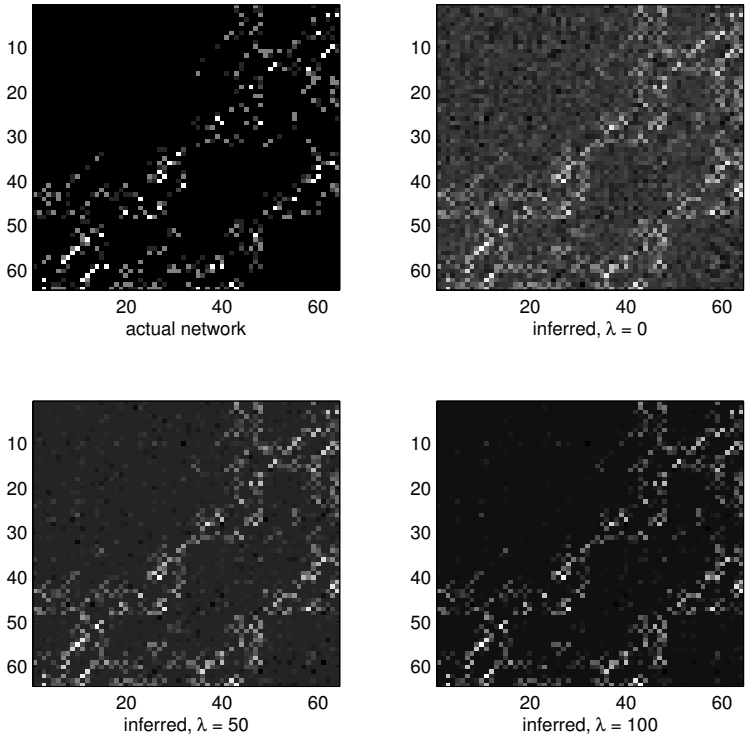


Figure 2.6: Actual adjacency matrix at $t = 900$ compared with the inferred adjacency matrices using pseudo real-time FISTA (Alg. 3), with $\lambda_t = \lambda$ for all t and $\lambda = 0$, $\lambda = 50$, and $\lambda = 100$.

with them were retained.

The real-data tests that follow focus on two keywords: i) "Kim Jong-un" the current leader of North Korea whose popularity rose after the death of his father and predecessor, during the observation period; and ii) "Reid Hoffman" the founder of the professional online social network *LinkedIn*, that went public during the observation period. Each keyword is associated with several phrases mentioned on the web by blogs and news websites that covered the two individuals. Each phrase is assigned a list of tuples in the form (website id, timestamp) capturing the time when a website mentioned the phrase. Defining a cascade as any list with at least 7 tuples, the dataset was significantly reduced to the 360 websites over which 466 cascades related to "Kim Jong-un" propagated during a 45 week period. Similarly, 125 websites were retained for propagation of 85 cascades related to "Reid Hoffman" over 41 weeks. Each time interval was set to one week and the

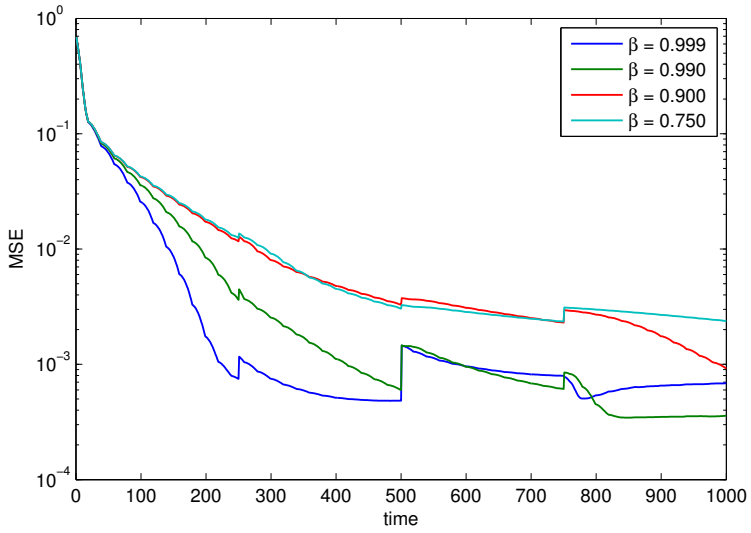


Figure 2.7: MSE performance of the pseudo real-time FISTA (Alg. 3) versus time, for different values of β.

observation time-scale was adjusted to start at the beginning of the earliest cascades.

In both cases, matrix $\mathbf{Y}^t$ was constructed by setting y_{ic}^t to the time when website i mentioned phrase c if this occurred during the span of week t. Otherwise y_{ic}^t was set to a large number, $100t_{\text{max}}$, where t_{max} denotes the largest timestamp in the dataset. Typically, the entries of matrix $\mathbf{X}$ capture prior knowledge about the susceptibility of each node to each contagion. For instance, the entry $\mathbf{x}_{ic}$ could denote the online search rank of website i for a search keyword associated with contagion c. In the absence of such real data, the entries of $\mathbf{X}$ were set to zero.

Experimental results. Algorithm 3 was run on real data with $\beta = 0.9$ and $\lambda_t = 100$. The choice of Algorithm 3 was based on its faster convergence properties when compared to the other alternatives, and that this is not a delay-sensitive application. Fig. 2.10 depicts visualizations of the inferred network at $t = 10$ and $t = 40$ weeks. Little was known about Kim Jong-un during the first 10 weeks of the observation period. However, speculation about the possible successor of the dying North Korean ruler, Kim Jong-il, rose until his death on December 17, 2011 (week 38). He was succeeded by Kim Jong-un on December 30, 2011 (week 40). The network visualizations show an increasing number of edges over the 45 weeks, illustrating the growing interest of international news websites and blogs in the new ruler. Unfortunately, the observation horizon does not go beyond

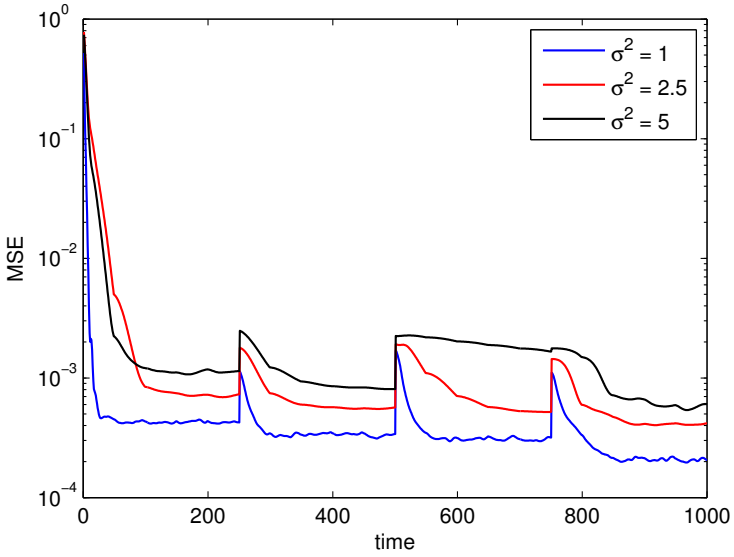


Figure 2.8: MSE performance of the pseudo real-time FISTA (Alg. 3) versus time, for different noise variance values.

$T = 45$ weeks. A longer span of data would have been useful to investigate at what rate did the global news coverage on the topic eventually subside.

Fig. 2.11 depicts the time evolution of the total number of edges in the inferred dynamic network. Of particular interest are the weeks during which: i) Kim Jong-un was appointed as the vice chairman of the North Korean military commission; ii) Kim Jong-il died; and iii) Kim Jong-un became the ruler of North Korea. These events were the topics of many online news articles and political blogs, an observation that is reinforced by the experimental results shown in the plot.

The results of running Algorithm 3 on the second dataset are shown in Figures 2.12 and 2.13. Although Reid Hoffman was already popular in technology media coverage, his visibility in popular news and blogs increased tremendously following the highly successful initial public offering (IPO) of LinkedIn on May 19, 2011; see Fig. 2.13. Towards the end of 2011, a number of other successful technology companies like Groupon and Zynga went public, possibly stabilizing the amount of media coverage on Reid Hoffman. In fact, the drop in the number of edges towards week 41 could be attributed to the captivation of media attention by the IPOs that occurred later in the year.

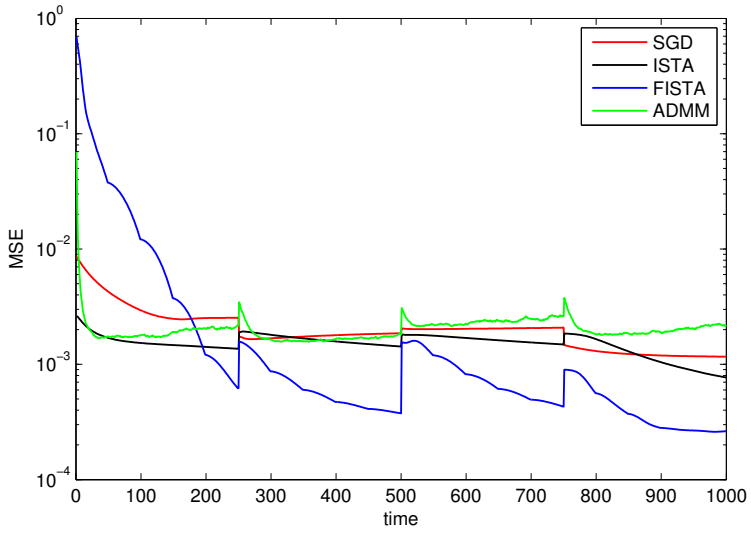
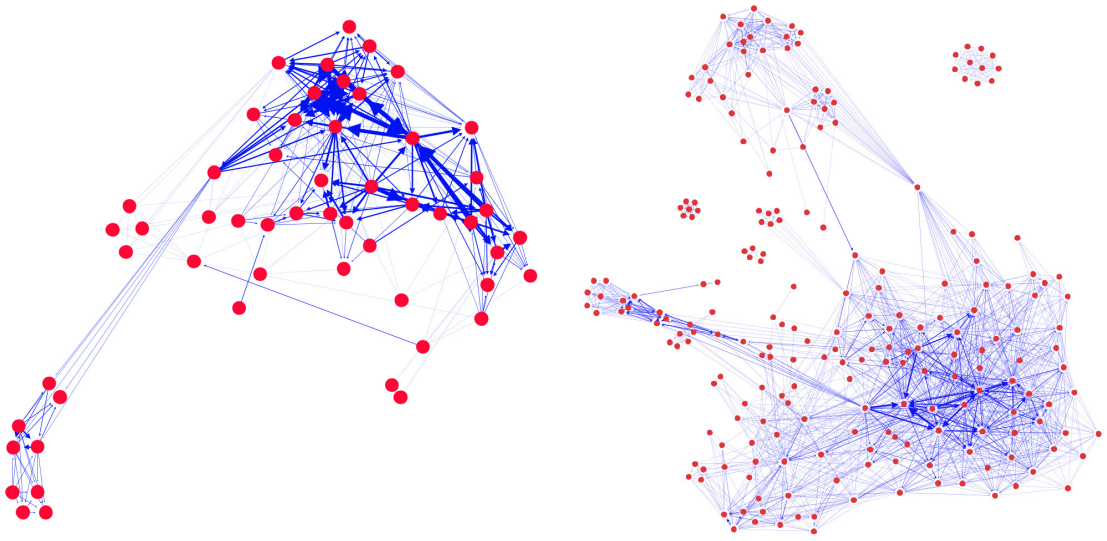


Figure 2.9: MSE performance comparison of the real-time algorithms versus time.

2.6 Chapter Summary

A dynamic SEM was proposed in this chapter for network topology inference, using timestamp data for propagation of contagions typically observed in social networks. The model explicitly captures both topological influences and external sources of information diffusion over the unknown network. Exploiting the inherent edge sparsity typical of large networks, algorithms based on ADMM and PG approaches were developed to minimize a suitable sparsity-regularized exponentially-weighted LS estimator. Focusing on the merits of PG methods for low-cost processing in big data settings, several algorithmic enhancements were proposed, pertaining to accelerating PG convergence and performing the network topology inference task in real time. In addition, a reduced-complexity stochastic-gradient descent variant was outlined and shown to attain worthwhile performance.

A number of experiments conducted on synthetically-generated data demonstrated the effectiveness of the proposed algorithms in tracking dynamic and sparse networks. Experimental results on real datasets focused on two popular personalities that made headlines during the observation period successfully showed changes in edge connectivity between media websites corresponding to increased media frenzy following specific events centered on them.



(a) $t = 10$ (b) $t = 40$

Figure 2.10: Visualization of estimated networks obtained by tracking information cascades related to "Kim Jong-un".

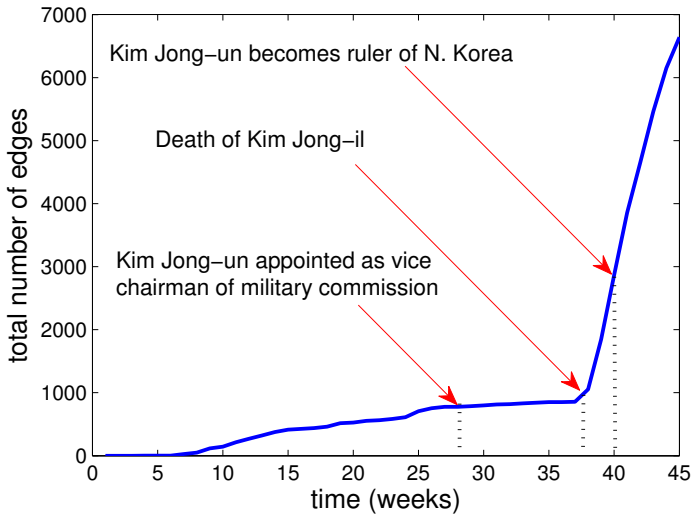
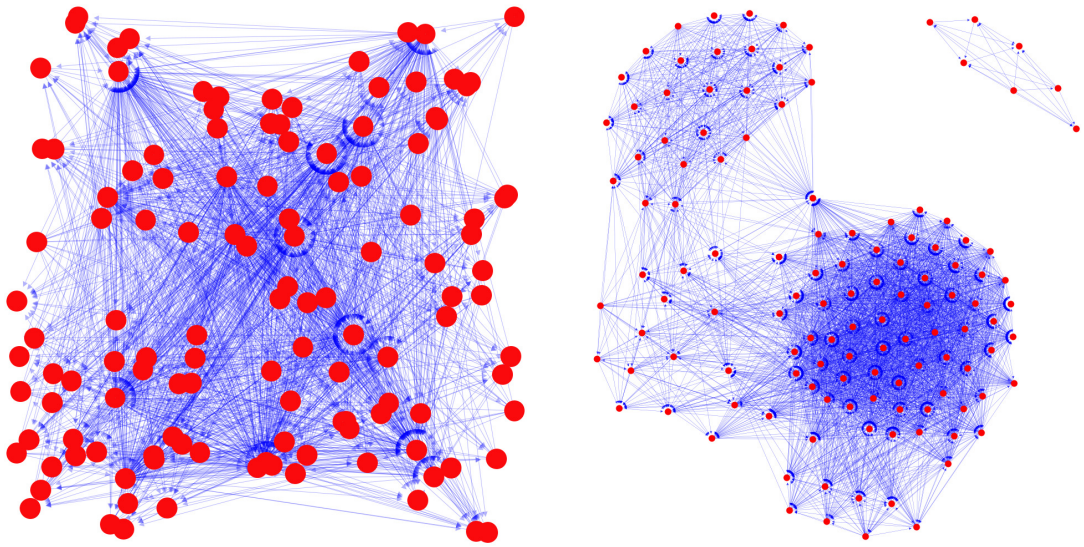


Figure 2.11: Plot depicting the evolution of the number of inferred edges per week for "Kim Jong-un" cascades.



(a) $t = 5$ (b) $t = 30$

Figure 2.12: Estimated networks obtained from "Reid Hoffman" cascades at $t = 5$ and 30 weeks.

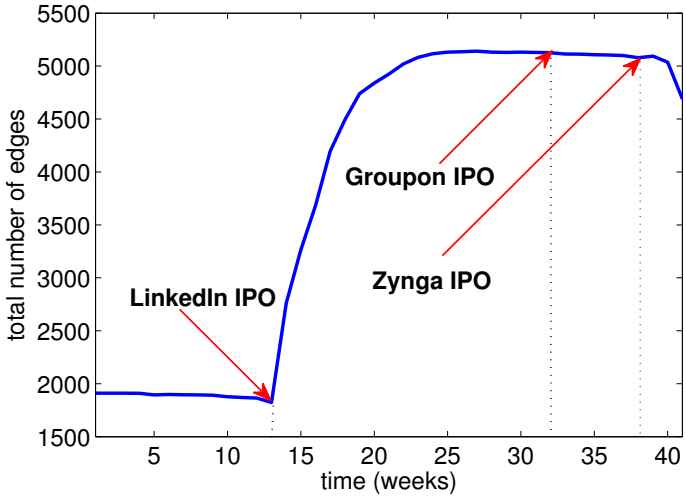


Figure 2.13: Plot depicting the evolution of the number of inferred edges per week for "Reid Hoffman" cascades.

Chapter 3

Tracking Switched Network Topologies

3.1 Introduction

Network topologies may sometimes "jump" between a finite number of discrete states, as manifested by sudden changes in cascade behavior. For example, an e-mail network may switch topologies from predominantly work-based connections during the week, to friend-based connections over the weekend. Connection dynamics between bloggers may switch suddenly at the peak of sports events (e.g., "Superbowl"), or presidential elections. In such settings, contemporary approaches assuming that network dynamics arise as a result of slow topology variations may yield unpredictable results. This chapter capitalizes on this prior knowledge, and puts forth a novel *switched* dynamic SEM to account for propagation of information cascades in such scenarios. The proposed approach builds upon the dynamic SEM advocated in the last chapter, where it is tacitly assumed that node infection times depend on both the switching topologies and exogenous influences e.g., external information sources, or prior predisposition to certain cascades.

This chapter draws connections to identification of hybrid systems, whose behavior is driven by interaction between continuous and discrete dynamics; see e.g., [107] and references therein. Switched linear models have emerged as a useful framework to capture piecewise linear input-output relations in control systems [23, 114, 126]. The merits of these well-grounded approaches are extended to temporal network inference from state-driven cascade dynamics. Although the evolution of the unknown network state sequence may be controlled by structured hidden dynamics

(e.g., *hidden Markov models*), this work advocates a more general framework in which such prior knowledge is not assumed.

Within the context of prior works on dynamic network inference, the contributions of this chapter are three-fold. First, a novel switched dynamic SEM that captures sudden topology changes within a finite state-space is put forth in Section 3.2. Second, identifiability results for the proposed switched model are established under reasonable assumptions in Section 3.3. Finally, an efficient sparsity-promoting proximal-gradient (PG) algorithm is developed (Sections 3.4 and 3.5) to jointly track the evolving state sequence, and the unknown network topologies. Numerical tests on both synthetic and real cascades in Section 3.6 corroborate the efficacy of the novel approach. Interestingly, experiments on real-world web cascade data reveal that media influence is dominated by major news outlets (e.g., *cnn.com* and *bbc.com*), as well as well-known web-based news aggregators (e.g., *news.yahoo.com* and *news.google.com*).

3.2 Model and Problem Statement

Consider a dynamic network with N nodes observed over time intervals $t = 1, \ldots, T$, captured by a graph whose topology may arbitrarily switch between S discrete states. Suppose the network topology during interval t is described by an unknown, weighted adjacency matrix $\mathbf{A}^{\sigma(t)} \in \mathbb{R}^{N \times N}$, with $\sigma(t) \in \mathcal{S} := \{1, \ldots, S\}$. Entry (i, j) of $\mathbf{A}^{\sigma(t)}$ (henceforth denoted by $a_{ij}^{\sigma(t)}$) is nonzero only if a directed edge connects nodes i and j (pointing from j to i) during interval t. In general $a_{ij}^{\sigma(t)} \neq a_{ji}^{\sigma(t)}$, i.e., matrix $\mathbf{A}^{\sigma(t)}$ is generally non-symmetric, which is suitable to model directed networks. The model tacitly assumes that the network topology remains fixed during any given time interval t, but can change across time intervals.

Suppose C contagions propagating over the network are observed, and the time of first infection per node i by contagion c is denoted by y_{ic}^t. In online media, y_{ic}^t can be obtained by recording the timestamp when blog i first mentioned news item c. If a node remains uninfected during t, y_{ic}^t is set to an arbitrarily large number. Assume that the susceptibility x_{ic} of node i to external (non-topological) infection by contagion c is known and time invariant. In the web context, x_{ic} can be set to the search engine rank of website i with respect to (w.r.t.) keywords associated with c.

The infection time of node i during interval t is modeled according to the following *switched*

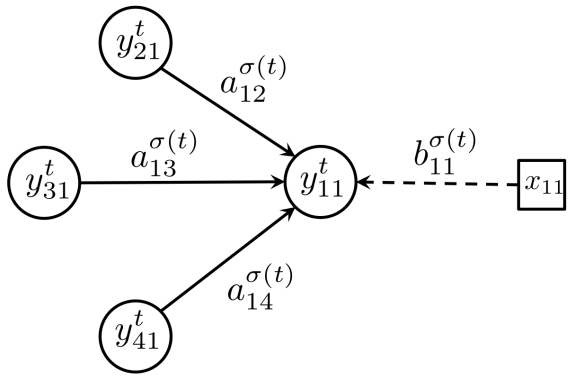


Figure 3.1: The cascade infection time for node 1 during interval t is a linear combination of cascade infection times of its single-hop neighbors, and the exogenous influence x_{11}. Unknown edge weights are captured through the coefficients $a_{12}^{\sigma(t)}, a_{12}^{\sigma(t)}$ and $a_{12}^{\sigma(t)}$.

dynamic structural equation model (SEM)

$$y_{ic}^t = \sum_{j \neq i} a_{ij}^{\sigma(t)} y_{jc}^t + b_{ii}^{\sigma(t)} x_{ic} + e_{ic}^t \tag{3.1}$$

where $b_{ii}^{\sigma(t)}$ captures the state-dependent level of influence of external sources, and e_{ic}^t accounts for measurement errors and unmodeled dynamics. It follows from (3.1) that if $a_{ij}^{\sigma(t)} \neq 0$, then y_{ic}^t is affected by the values of $\{y_{jc}^t\}_{j \neq i}$ (see Figure 3.1). With diagonal $\mathbf{B}^{\sigma(t)} := \text{diag}(b_{11}^{\sigma(t)}, \ldots, b_{NN}^{\sigma(t)})$, collecting observations for the entire network and all C contagions yields the dynamic matrix SEM

$$\mathbf{Y}_t = \mathbf{A}^{\sigma(t)} \mathbf{Y}_t + \mathbf{B}^{\sigma(t)} \mathbf{X} + \mathbf{E}_t \tag{3.2}$$

where $\mathbf{Y}_t := \left[y_{ic}^t\right]$, $\mathbf{X} := [x_{ic}]$, and $\mathbf{E}_t := \left[e_{ic}^t\right]$ are all $N \times C$ matrices. A single network topology $\mathbf{A}^{\sigma(t)}$ is adopted for all contagions, which is suitable e.g., when information cascades are formed around a common meme or trending (news) topic over the Internet. Matrix $\mathbf{A}^{\sigma(t)}$ has an all-zero diagonal to ensure that the underlying network is free of self loops, i.e., $a_{ii}^{\sigma(t)} = 0 \ \ \forall i, \forall t$.

Problem statement. Given the sequence $\{\mathbf{Y}_t\}_{t=1}^T$ and $\mathbf{X}$ adhering to (3.2), the goal is to identify the unknown state matrices $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$, and the switching sequence $\{\sigma(t) \in \mathcal{S}\}_{t=1}^T$.

3.3 Model Identifiability

This section explores the conditions under which one can uniquely recover the unknown state matrices $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$. First, (3.2) can be written as

$$\mathbf{Y}_t = \left(\sum_{s=1}^{S} \chi_{ts}\mathbf{A}^s\right)\mathbf{Y}_t + \left(\sum_{s=1}^{S} \chi_{ts}\mathbf{B}^s\right)\mathbf{X} + \mathbf{E}_t \tag{3.3}$$

where the binary indicator variable $\chi_{ts} = 1$ if $\sigma(t) = s$, otherwise $\chi_{ts} = 0$. Note that (3.3) is generally underdetermined when S is unknown, and it may not be possible to uniquely identify $\{\mathbf{A}^s, \mathbf{B}^s, \{\chi_{ts}\}_{t=1}^T\}_{s=1}^S$. Even in the worst-case scenario where $S = T$, complete identification of all states is impossible if there exists t and t' ($t \neq t'$) such that $\sigma(t) = \sigma(t')$. In order to establish model identifiability, it will be assumed that the following hold:

as1. The dynamic SEM in (3.2) is noise-free ($\mathbf{E}_t = \mathbf{0}$); that is,

$$\mathbf{Y}_t = \mathbf{A}^{\sigma(t)}\mathbf{Y}_t + \mathbf{B}^{\sigma(t)}\mathbf{X}. \tag{3.4}$$

as2. All cascades $\{\mathbf{Y}_t\}_{t=1}^T$ are generated by some pair $\{\mathbf{A}^s, \mathbf{B}^s\}$, where $s \in \{1, \ldots, S\}$, and S is known. This is a *realizability assumption* which guarantees the existence of a submodel responsible for the observed cascades.

as3. No two states can be active during a given time interval, i.e., $\|\mathbf{Y}_t - \mathbf{A}^s\mathbf{Y}_t - \mathbf{B}^s\mathbf{X}\|_F = \|\mathbf{Y}_t - \mathbf{A}^{s'}\mathbf{Y}_t - \mathbf{B}^{s'}\mathbf{X}\|_F = 0$ implies that $s = s'$.

Under (as1)-(as3), the following proposition holds.

Proposition 1. *Suppose data matrices $\mathbf{Y}_t$ and $\mathbf{X}$ adhere to (3.4) with $a_{ii}^{\sigma(t)} = 0$, $b_{ii}^{\sigma(t)} \neq 0, \ \forall\, i, t$, and $b_{ij}^{\sigma(t)} = 0 \ \ \forall\, i \neq j, t$. If $N \leq C$ and $\mathbf{X}$ has full row rank, then $\mathbf{A}^{\sigma(t)}$ and $\mathbf{B}^{\sigma(t)}$ are uniquely expressible in terms of $\mathbf{X}$ and $\mathbf{Y}_t$ as $\mathbf{B}^{\sigma(t)} = \left(Diag\left[\left(\mathbf{Y}_t\mathbf{X}^\dagger\right)\right]\right)^{-1}$, and $\mathbf{A}^{\sigma(t)} = \mathbf{I} - \mathbf{B}^{\sigma(t)}\left(\mathbf{Y}_t\mathbf{X}^\dagger\right)^{-1}$. If $p_s := Pr\left(\chi_{ts} = 1\right) > 0 \ \forall s$ denotes the activation probability of state s, with $\sum_{s=1}^S p_s = 1$, and (as1)-(as3) hold, then $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$ can be uniquely identified with probability one as $T \to \infty$.*

Proposition 1 establishes a two-step identifiability result for the dynamic SEM model in (3.4). First, it establishes that if the exogenous data matrix $\mathbf{X}$ is sufficiently rich (i.e., $\mathbf{X}$ has full row rank), then the per-interval network topology captured by $\mathbf{B}^{\sigma(t)}$ can be uniquely identified per t. Second, if the state activation probabilities are all strictly positive, then the S switching network topologies will

all be recovered as $T \to \infty$. In fact, if cascade data are acquired in an infinitely streaming fashion ($T = \infty$), then one is guaranteed to uniquely recover all states.

Proof of Proposition 1: Equation (3.4) can be written as $(\mathbf{I} - \mathbf{A}^{\sigma(t)})\mathbf{Y}_t = \mathbf{B}^{\sigma(t)}\mathbf{X}$, which implies that rank $\left((\mathbf{I} - \mathbf{A}^{\sigma(t)})\mathbf{Y}_t\right) = $ rank $\left(\mathbf{B}^{\sigma(t)}\mathbf{X}\right)$. With $\mathbf{X}$ and $\mathbf{B}^{\sigma(t)}$ both having full row rank (recall that $b_{ii}^{\sigma(t)} > 0$), it follows that $(\mathbf{I} - \mathbf{A}^{\sigma(t)})\mathbf{Y}_t$ has full row rank, and $\mathbf{I} - \mathbf{A}^{\sigma(t)}$ is invertible. Consequently, the linear system of equations $\mathbf{Y}_t = \mathbf{\Phi}\mathbf{X}$ is solved by $\mathbf{\Phi}^* = \left(\mathbf{I} - \mathbf{A}^{\sigma(t)}\right)^{-1}\mathbf{B}^{\sigma(t)}$.

On the other hand, note that $\mathbf{Y}_t = \mathbf{\Phi}\mathbf{X}$ admits the solution

$$\mathbf{\Phi}^* = \mathbf{Y}_t\mathbf{X}^\dagger \tag{3.5}$$

which is unique since $\mathbf{X}$ is full row rank. Since $a_{ii}^{\sigma(t)} = 0$ and $\left(\mathbf{B}^{\sigma(t)}\right)^{-1}$ is a diagonal matrix, the diagonal entries of $(\mathbf{\Phi}^*)^{-1} = \left(\mathbf{B}^{\sigma(t)}\right)^{-1}\left(\mathbf{I} - \mathbf{A}^{\sigma(t)}\right)$ coincide with those of $\left(\mathbf{B}^{\sigma(t)}\right)^{-1}$; hence, $\mathbf{B}^{\sigma(t)} = \left(\text{Diag}\left[(\mathbf{\Phi}^*)^{-1}\right]\right)^{-1}$, which leads to

$$\mathbf{B}^{\sigma(t)} = \left(\text{Diag}\left[\left(\mathbf{Y}_t\mathbf{X}^\dagger\right)^{-1}\right]\right)^{-1} \tag{3.6}$$

upon substitution of (3.5). Furthermore, note that $\mathbf{B}^{\sigma(t)}\left(\mathbf{Y}_t\mathbf{X}^\dagger\right)^{-1} = \mathbf{I} - \mathbf{A}^{\sigma(t)}$, and thus

$$\mathbf{A}^{\sigma(t)} = \mathbf{I} - \left[\text{Diag}\left(\left(\mathbf{Y}_t\mathbf{X}^\dagger\right)^{-1}\right)\right]^{-1}\left(\mathbf{Y}_t\mathbf{X}^\dagger\right)^{-1} \tag{3.7}$$

which concludes the first part of the proof.

Since $\mathbf{A}^{\sigma(t)} = \sum_{s=1}^{S}\chi_{ts}\mathbf{A}^s$ and $\mathbf{B}^{\sigma(t)} = \sum_{s=1}^{S}\chi_{ts}\mathbf{B}^s$, the results from the unique solutions (3.6) and (3.7) coincide with a specific pair in the set $\{(\mathbf{A}^s, \mathbf{B}^s)\}_{s=1}^S$ per t. Intuitively, complete recovery of all state matrices is tantamount to identification of S unique pairs $(\mathbf{A}^s, \mathbf{B}^s)$ as more data are sequentially acquired. In order to guarantee identifiability of all states in the long run, it suffices to prove that the probability of activation of any state at least once tends to 1 as $T \to \infty$. Since χ_{ts} are Bernoulli random variables with $p_s = \Pr(\chi_{ts} = 1)$ per t, the number of times that state s is activated over T time intervals follows a binomial distribution. Letting $T_s := \sum_{t=1}^{T}\chi_{ts}$ denote the total number of activations of state s over $t = 1, \ldots, T$, then

$$\begin{aligned}\Pr(T_s \geq 1) &= 1 - \Pr(T_s = 0) \\ &= 1 - (1 - p_s)^T. \end{aligned} \tag{3.8}$$

Since $\lim_{T\to\infty} \Pr(T_s \geq 1) = 1$ only if $p_s > 0$, it follows that all states can be uniquely identified with probability 1 as $T \to \infty$ when $p_s > 0 \ \ \forall s$, which completes the second part of the proof.

3.3.1 Topology tracking by clustering when $\mathbf{E}_t \neq \mathbf{0}$

In general, even when $\mathbf{X}$ is full row rank, $\mathbf{E}_t \neq \mathbf{0}$ in order to compensate for measurement errors and unmodeled dynamics. Under noisy conditions, it will turn out that $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$ can be interpreted as cluster centroids (cf. (3.3)), and one can leverage traditional clustering approaches (e.g., k-means), coupled with the closed-form solutions in (3.6) and (3.7) to identify the unknown state matrices. Indeed, with $\mathbf{b}^{\sigma(t)} := \text{diag}\left(\mathbf{B}^{\sigma(t)}\right)$, and $\boldsymbol{\theta}^t := \text{vec}\left(\left[\mathbf{A}^{\sigma(t)} \ \mathbf{b}^{\sigma(t)}\right]\right)$, it follows readily that

$$\boldsymbol{\theta}^t = \sum_{s=1}^{S} \chi_{ts} \boldsymbol{\theta}^s \ \ t = 1, \ldots, T \tag{3.9}$$

where $\boldsymbol{\theta}^s := \text{vec}\left([\mathbf{A}^s \ \mathbf{b}^s]\right)$, and $\mathbf{b}^s := \text{diag}(\mathbf{B}^s)$.

Clearly, $\{\boldsymbol{\theta}^s\}_{s=1}^S$ in (3.9) can be viewed as S cluster centers, and χ_{ts} as unknown binary cluster-assignment variables. Consequently, the sequence $\{\boldsymbol{\theta}^t\}_{t=1}^{T_{\text{cluster}}}$ can be directly computed per t via (3.6) and (3.7), where $T_{\text{cluster}} < T$ denotes the number of training samples. Identification of $\{\chi_{ts}\}$ and $\{\boldsymbol{\theta}^s\}$ can then be accomplished by batch clustering $\{\boldsymbol{\theta}^t\}$ into S clusters. The subsequent operational phase ($t > T_{\text{cluster}}$) then boils down to computing $\boldsymbol{\theta}^t$, followed by finding the centroid $\boldsymbol{\theta}^s$ to which it is closest in Euclidean distance, that is

$$\hat{s} = \arg\min_{s} \ \|\boldsymbol{\theta}^t - \boldsymbol{\theta}^s\|_2 \tag{3.10}$$

and $\hat{\chi}_{ts} = 1$ if $s = \hat{s}$, otherwise $\hat{\chi}_{ts} = 0$. Algorithm 6 summarizes this cluster-based state identification scheme, with a training phase ($t \leq T_{\text{cluster}}$), and an operational phase ($t > T_{\text{cluster}}$). The sub-procedure cluster($\{\boldsymbol{\theta}^t\}_{t=1}^{T_{\text{train}}}, S$) calls an off-the-shelf clustering algorithm with the training set $\{\boldsymbol{\theta}^t\}_{t=1}^{T_{\text{train}}}$ and the number of clusters S as inputs.

Remark 4. Algorithm 6 identifies the unknown state-dependent matrices by resorting to "hard clustering," which entails deterministic assignment of $\boldsymbol{\theta}^t$ to one of the S centroids. In principle, the algorithm can be readily modified to adopt "soft clustering" approaches, with probabilistic assignments to the cluster centroids.

Algorithm 6 Switched topology identification by clustering

Require: $\{\mathbf{Y}_t\}_{t=1}^{T}$, $\mathbf{X}$, S, $T_{\text{cluster}} < T$

1: **for** $t = 1, \ldots, T_{\text{cluster}}$ **do**

2: $\quad \mathbf{b}^{\sigma(t)} = \text{diag}[(\text{Diag}[(\mathbf{Y}_t\mathbf{X}^\dagger)^{-1}])^{-1}]$

3: $\quad \mathbf{A}^{\sigma(t)} = \mathbf{I} - [\text{Diag}((\mathbf{Y}_t\mathbf{X}^\dagger)^{-1})]^{-1}(\mathbf{Y}_t\mathbf{X}^\dagger)^{-1}$

4: $\quad \boldsymbol{\theta}^t := \text{vec}\left(\left[\mathbf{A}^{\sigma(t)}\ \mathbf{b}^{\sigma(t)}\right]\right)$

5: **end for**

6: $\{\boldsymbol{\theta}^s, \{\chi_{ts}\}_{t=1}^{T_{\text{cluster}}}\}_{s=1}^{S} = \text{cluster}(\{\boldsymbol{\theta}^t\}_{t=1}^{T_{\text{cluster}}}, S)$

7: Set $\hat{\sigma}(t) = s$ if $\chi_{ts} = s$ for $t = 1, \ldots, T_{\text{cluster}}$

8: Extract $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^{S}$ from $\{\boldsymbol{\theta}^s\}_{s=1}^{S}$

9: **for** $t > T_{\text{cluster}}$ **do**

10: $\quad$ Compute $\boldsymbol{\theta}^t$ using lines $(2) - (4)$

11: $\quad \hat{\sigma}(t) = \arg\min_{s} \ \|\boldsymbol{\theta}^t - \boldsymbol{\theta}^s\|_2$

12: **end for**

13: **return** $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^{S}, \{\hat{\sigma}(t)\}_{t=1}^{T}$

3.4 Exploiting edge sparsity

The fundamental premise established by Proposition 1 is that $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^{S}$ are uniquely identifiable provided $\mathbf{X}$ is sufficiently rich. However, requiring that $\mathbf{X}$ has full row rank is a restrictive condition, tantamount to requiring that at least as many cascades are observed as the number of nodes ($C \geq N$). This is especially prohibitive in large-scale networks such as the world-wide web, with billions of nodes. It is therefore of interest to measure as few cascades as possible while ensuring that $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^{S}$ are uniquely recovered. To this end, one is motivated to leverage prior knowledge about the unknowns in order to markedly reduce the amount of data required to guarantee model identifiability. For example, each node in the network is connected only to a small number of nodes out of the $N - 1$ possible connections. As a result, most practical networks exhibit edge sparsity, a property that can be exploited to ensure that the rank condition on $\mathbf{X}$ can be relaxed, as shown in the sequel. In addition to (as1)-(as3), consider the following.

as4. Each row $\mathbf{a}_n^s$ of $\mathbf{A}^s$ has at most K nonzero entries; i.e., $\|\mathbf{a}_n^s\|_0 \leq K \ \ \forall n, s$.

as5. The nonzero entries of $\mathbf{A}^s$ for all $s = 1, \ldots, S$ are drawn from a continuous distribution.

as6. The Kruskal rank of $\mathbf{X}^\top$ satisifies kr $\left(\mathbf{X}^\top\right) \geq 2K+1$, where kr $(\mathbf{Z})$ is defined as the maximum number k such that any combination of k columns of $\mathbf{Z}$ constitute a full column rank submatrix.

Proposition 2. *Suppose data matrices $\mathbf{Y}_t$ and $\mathbf{X}$ adhere to (3.4) with $a_{ii}^{\sigma(t)} = 0$, $b_{ii}^{\sigma(t)} \neq 0, \ \forall\, i, t$, and $b_{ij}^{\sigma(t)} = 0 \ \ \forall\, i \neq j, t$. If (as1)-(as6) hold, and $p_s > 0 \ \ \forall s$, then $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$ can be uniquely identified with probability one as $T \to \infty$.*

The proof of Proposition 2 is rather involved, and is deferred to Appendix B. Unlike the matrix rank which only requires existence of a subset of linearly independent columns, the Kruskal rank requires that every possible combination of a given number of columns be linearly independent. Moreover, computing kr $\left(\mathbf{X}^\top\right)$ is markedly more challenging, as it entails a combinatorial search over the rows of $\mathbf{X}$. Admittedly, requiring kr $\left(\mathbf{X}^\top\right) \geq 2K+1$ is more restrictive than rank $\left(\mathbf{X}^\top\right) \geq 2K+1$. Nevertheless, in settings where $2K+1 \ll N$, (as6) may be satisfied even if rank $(\mathbf{X}) < N$. In such cases, Proposition 2 asserts that one can uniquely identify $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$ even when $C \leq N$.

3.4.1 Sparsity-promoting estimator

The remainder of this chapter leverages inherent edge sparsity to develop an efficient algorithm to track switching network topologies from noisy cascade traces. Assuming S is known a priori, one is motivated to solve the following regularized LS batch estimator

$$\text{(P0)} \quad \underset{\substack{\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S \\ \{\{\chi_{ts}\}_{s=1}^S\}_{t=1}^T}}{\arg\min} \quad (1/2)\sum_{t=1}^{T}\sum_{s=1}^{S} \chi_{ts} \|\mathbf{Y}_t - \mathbf{A}^s\mathbf{Y}_t - \mathbf{B}^s\mathbf{X}\|_F^2 + \sum_{s=1}^{S} \lambda_s \|\mathbf{A}^s\|_1$$

$$\text{s.t.} \quad \sum_{s=1}^{S} \chi_{ts} = 1 \ \forall t, \ \ \chi_{ts} \in \{0,1\} \ \forall s, t$$

$$a_{ii}^s = 0, \ b_{ij}^s = 0, \ \forall s, i \neq j \tag{3.11}$$

where the constraint $\sum_{s=1}^S \chi_{ts} = 1$ enforces a realizability condition, ensuring that only one state can account for the system behavior at any time. With $\|\mathbf{A}^s\|_1 := \sum_{ij} |a_{ij}^s|$, the regularization term promotes edge sparsity that is inherent to most real networks. The sparsity level of $\hat{\mathbf{A}}^s$ is controlled by $\lambda_s > 0$. Absence of a self-loop at node i is enforced by the constraint $a_{ii}^s = 0$, while having $b_{ij}^s = 0, \ \forall i \neq j$, ensures that $\hat{\mathbf{B}}^s$ is diagonal as in (3.2).

Note that (P0) is an NP-hard mixed integer program that is unsuitable for large-scale and potentially real-time operation. Moreover, entries of per-state matrices $\{\mathbf{A}^s, \mathbf{B}^s\}$ may evolve slowly over reasonably long observation periods, motivating algorithms that not only track the switching sequence, but also the slow drifts occurring in per-state network topologies.

Suppose data are sequentially acquired, rendering batch estimators impractical. If it is assumed that instantaneous and past estimates of $\{\{\hat{\chi}_{\tau s}\}_{s=1}^{S}\}_{\tau=1}^{t}$ are known during interval t, (P0) decouples over $\mathcal{S}$, and is tantamount to solving the following subproblem per t and s

$$\text{(P1)} \quad \underset{\mathbf{A}^s, \mathbf{B}^s}{\arg\min} \quad \frac{1}{2}\sum_{\tau=1}^{t} \hat{\chi}_{\tau s} \|\mathbf{Y}_\tau - \mathbf{A}^s\mathbf{Y}_\tau - \mathbf{B}^s\mathbf{X}\|_F^2 + \lambda_s \|\mathbf{A}^s\|_1$$
$$\text{s.t.} \quad a_{ii}^s = 0,\ b_{ij}^s = 0,\ \forall i \neq j. \tag{3.12}$$

Note that the LS term in the penalized cost only aggregates residuals when $\hat{\chi}_{\tau s} = 1$ ($\hat{\sigma}(t) = s$). In principle, during interval t, (P1) updates the estimates of $\mathbf{A}^s$ and $\mathbf{B}^s$ only if $\mathbf{Y}_t$ has been generated by submodel s. Critical to efficiently tracking the hidden network topologies is the need for a reliable approach to estimate $\chi_{\tau s}$. Before developing an efficient tracking algorithm that will be suitable to solve (P1), the rest of this section puts forth criteria for estimating $\sigma(t)$.

Estimation of $\boldsymbol{\sigma(t)}$. Given the most recent estimates $\{\hat{\mathbf{A}}^s, \hat{\mathbf{B}}^s\}_{s=1}^{S}$ prior to acquisition of $\mathbf{Y}_t$, one can estimate $\sigma(t)$ by minimizing the a priori error over $\mathcal{S}$ i.e.,

$$\hat{\sigma}(t) = \arg\min_{s\in\mathcal{S}} \ \|\mathbf{Y}_t - \hat{\mathbf{A}}^s\mathbf{Y}_t - \hat{\mathbf{B}}^s\mathbf{X}\|_F \tag{3.13}$$

followed by solving (P1) with $s = \hat{\sigma}(t)$. If network dynamics arise from switching between static or slowly-varying per-state topologies, (3.13) yields a reliable estimate of the most likely state sequence over time.

Alternatively, one may resort to a criterion that depends on minimizing the a posteriori error. This entails first solving (P1) for all states $s \in \mathcal{S}$ upon acquisition of $\mathbf{Y}^t$, and then selecting $s = \hat{\sigma}(t)$ so that

$$\hat{\sigma}(t) = \arg\min_{s\in\mathcal{S}} \ \|\mathbf{Y}_t - \hat{\mathbf{A}}^{(s|\sigma(t)=s)}\mathbf{Y}_t - \hat{\mathbf{B}}^{(s|\sigma(t)=s)}\mathbf{X}\|_F \tag{3.14}$$

where $\hat{\mathbf{A}}^{(s|\sigma(t)=s)}$ (resp. $\hat{\mathbf{B}}^{(s|\sigma(t)=s)}$) denotes the estimate of $\mathbf{A}^s$ (resp. $\mathbf{B}^s$) given that $\sigma(t) = s$. In the context of big data and online streaming, (3.14) is prohibitive, and (3.13) is more desirable for its markedly lower computational overhead. The tracking algorithm developed next adopts (3.13) for state sequence estimation.

3.5 Topology Tracking Algorithm

In order to solve (P1), this section resorts to proximal gradient (PG) approaches, which have been popularized for ℓ_1-norm regularized problems, through the class of iterative shrinkage-thresholding algorithms (ISTA); see e.g., [50] and [108]. Unlike off-the-shelf interior point methods, ISTA is computationally simple, with iterations entailing matrix-vector multiplications, followed by a soft-thresholding operation [64, p. 93]. Motivated by its well-documented merits, an ISTA algorithm is developed in the sequel for recursively solving (P1) per t. Memory storage and computational costs incurred by the algorithm per acquired sample $\mathbf{Y}_t$ do not grow with t.

Solving (P1) for a single time interval t. Introducing the optimization variable $\mathbf{V}^s := [\mathbf{A}^s \ \ \mathbf{B}^s]$, it follows that the gradient of $f(\mathbf{V}^s) := (1/2)\sum_{\tau=1}^{t} \hat{\chi}_{\tau s}\|\mathbf{Y}_\tau - \mathbf{A}^s\mathbf{Y}_\tau - \mathbf{B}^s\mathbf{X}\|_F^2$ is Lipschitz continuous, meaning there exists a constant L_f so that $\|\nabla f(\mathbf{V}_1^s) - \nabla f(\mathbf{V}_2^s)\| \leq L_f\|\mathbf{V}_1^s - \mathbf{V}_2^s\|$, $\forall \mathbf{V}_1^s, \mathbf{V}_2^s$ in the domain of f. Instead of directly optimizing the cost in (P1), PG algorithms minimize a sequence of overestimators evaluated at the current iterate, or a linear combination of the two previous iterates.

Letting $k = 1, 2, \ldots$ denote the iteration index, and $g(\mathbf{V}^s) := \lambda_s\|\mathbf{A}^s\|_1$, PG algorithms iterate

$$\mathbf{V}^s[k] := \arg\min_{\mathbf{V}} \left\{ \frac{L_f}{2}\|\mathbf{V} - \mathbf{G}(\mathbf{V}^s[k-1])\|_F^2 + g(\mathbf{V}) \right\} \tag{3.15}$$

where $\mathbf{G}(\mathbf{V}^s[k-1])$:=$\mathbf{V}^s[k-1] - (1/L_f)\nabla f(\mathbf{V}^s[k-1])$ corresponds to a gradient-descent step taken from $\mathbf{V}^s[k-1]$, with step-size equal to $1/L_f$. The optimization problem (3.15) is denoted by $\text{prox}_{g/L_f}(\mathbf{G}(\mathbf{V}^s[k-1]))$, and is known as the proximal operator of the function g/L_f evaluated at $\mathbf{G}(\mathbf{V}^s[k-1])$. With $\mathbf{G}^s[k-1] := \mathbf{G}(\mathbf{V}^s[k-1])$ for convenience, the PG iterations can be compactly rewritten as $\mathbf{V}[k] = \text{prox}_{g/L_f}(\mathbf{G}^s[k-1])$.

The success of PG algorithms hinges upon efficient evaluation of the proximal operator (cf. (3.15)). Focusing on (P1), note that (3.15) decomposes into

$$\begin{aligned} \mathbf{A}^s[k] &:= \arg\min_{\mathbf{A}^s} \left\{ \frac{L_f}{2}\|\mathbf{A}^s - \mathbf{G}_A^s[k-1]\|_F^2 + \lambda_s\|\mathbf{A}^s\|_1 \right\} \\ &= \text{soft}_{\lambda_s/L_f}(\mathbf{G}_A^s[k-1]) \end{aligned} \tag{3.16}$$

$$\mathbf{B}^s[k] := \arg\min_{\mathbf{B}^s} \left\{ \|\mathbf{B}^s - \mathbf{G}_B^s[k-1]\|_F^2 \right\} = \mathbf{G}_B^s[k-1] \tag{3.17}$$

subject to the constraints in (P1) which so far have been left implicit, and $\mathbf{G}^s:=[\mathbf{G}_A^s\ \mathbf{G}_B^s]$. Letting $\text{soft}_\mu(\mathbf{M})$ with (i,j)-th entry given by $\text{sign}(m_{ij})\max(|m_{ij}|-\mu,0)$ denote the soft-thresholding operator, it follows that $\text{prox}_{\lambda_s\|\cdot\|_1/L_f}(\cdot)=\text{soft}_{\lambda_s/L_f}(\cdot)$, see e.g., [50,64]. Since there is no regularization on $\mathbf{B}^s$, (3.17) boils-down to a simple gradient-descent step.

Specification of $\mathbf{G}_A^s$ and $\mathbf{G}_B^s$ only requires expressions for the gradient of $f(\mathbf{V}^s)$ with respect to $\mathbf{A}^s$ and $\mathbf{B}^s$. Note that by incorporating the constraints $a_{ii}^s=0$ and $b_{ij}^s=0,\ \forall j\neq i, i=1,\ldots N$, one can express $f(\mathbf{V}^s)$ as

$$f(\mathbf{V}^s):=\frac{1}{2}\sum_{\tau=1}^{t}\sum_{i=1}^{N}\hat{\chi}_{\tau s}\|\mathbf{y}_{i,\tau}^\top-(\mathbf{a}_{-i}^s)^\top\mathbf{Y}_{-i,\tau}-b_{ii}^s\mathbf{x}_i^\top\|_2^2 \tag{3.18}$$

where $\mathbf{y}_{i,\tau}^\top$ and $\mathbf{x}_i^\top$ denote the i-th row of $\mathbf{Y}_\tau$ and $\mathbf{X}$, respectively; while $(\mathbf{a}_{-i}^s)^\top$ denotes the $1\times(N-1)$ vector obtained by removing entry i from the i-th row of $\mathbf{A}^s$, and likewise $\mathbf{Y}_{-i,\tau}$ is the $(N-1)\times C$ matrix obtained by removing row i from $\mathbf{Y}_\tau$. It is apparent from (3.18) that $f(\mathbf{V}^s)$ is separable across the trimmed row vectors $(\mathbf{a}_{-i}^s)^\top$, and the diagonal entries b_{ii}^s, $i=1,\ldots,N$. The sought gradients are

$$\nabla_{\mathbf{a}_{-i}^s}f(\mathbf{V}^s)=\boldsymbol{\Omega}_{-i,t}^s\mathbf{a}_{-i}^s+\bar{\mathbf{Y}}_{-i,t}^s\mathbf{x}_ib_{ii}^s-\boldsymbol{\omega}_{-i,t}^s \tag{3.19}$$

$$\nabla_{b_{ii}^s}f(\mathbf{V}^s)=(\mathbf{a}_{-i}^s)^\top\bar{\mathbf{Y}}_{-i,t}^s\mathbf{x}_i+\alpha_t^sb_{ii}^s\|\mathbf{x}_i\|_2^2-(\bar{\mathbf{y}}_{i,t}^s)^\top\mathbf{x}_i \tag{3.20}$$

where $(\bar{\mathbf{y}}_{i,t}^s)^\top$ denotes the i-th row of $\bar{\mathbf{Y}}_t^s:=\sum_{\tau=1}^t\hat{\chi}_{\tau s}\mathbf{Y}_\tau$, $\bar{\mathbf{Y}}_{-i,t}^s:=\sum_{\tau=1}^t\hat{\chi}_{\tau s}\mathbf{Y}_{-i,\tau}$, and $\alpha_t^s:=\sum_{\tau=1}^t\hat{\chi}_{\tau s}$. Similarly, $\boldsymbol{\omega}_{-i,t}^s:=\sum_{\tau=1}^t\hat{\chi}_{\tau s}\mathbf{Y}_{-i,\tau}\mathbf{y}_{i,\tau}$ and $\boldsymbol{\Omega}_{-i,t}^s$ is obtained by removing the i-th row and i-th column from $\boldsymbol{\Omega}_t^s:=\sum_{\tau=1}^t\hat{\chi}_{\tau s}\mathbf{Y}_\tau\mathbf{Y}_\tau^\top$. From (3.16)-(3.17) and (3.19)-(3.20), the parallel ISTA iterations

$$\nabla_{\mathbf{a}_{-i}^s}f[k]=\boldsymbol{\Omega}_{-i,t}^s\mathbf{a}_{-i}^s[k]+\bar{\mathbf{Y}}_{-i,t}^s\mathbf{x}_ib_{ii}^s[k]-\boldsymbol{\omega}_{-i,t}^s \tag{3.21}$$

$$\nabla_{b_{ii}^s}f[k]=(\mathbf{a}_{-i}^s)^\top[k]\bar{\mathbf{Y}}_{-i,t}^s\mathbf{x}_i+\alpha_t^sb_{ii}^s[k]\|\mathbf{x}_i\|_2^2-(\bar{\mathbf{y}}_{i,t}^s)^\top\mathbf{x}_i \tag{3.22}$$

$$\mathbf{a}_{-i}^s[k+1]=\text{soft}_{\lambda_s/L_f}\left(\mathbf{a}_{-i}^s[k]-(1/L_f)\nabla_{\mathbf{a}_{-i}^s}f[k]\right) \tag{3.23}$$

$$b_{ii}^s[k+1]=b_{ii}^s[k]-(1/L_f)\nabla_{b_{ii}^s}f[k] \tag{3.24}$$

are provably convergent to the globally optimal solution $\{\hat{\mathbf{A}}^s,\hat{\mathbf{B}}^s\}$ of (P1), as per general convergence results for PG methods [50, 108].

Note that iterations (3.23)-(3.24) incur a low computational overhead, involving at most matrix-vector multiplication complexity for the gradient evaluations. A final step entails zero-padding the updated $\mathbf{a}^s_{-i}[k]$ by setting

$$(\mathbf{a}^s_i)^\top[k] = [a^s_{-i,1}[k] \dots a^s_{-i,i-1}[k]\ 0\ a^s_{-i,i}[k] \dots a^s_{-i,N}[k]]. \tag{3.25}$$

The desired SEM parameter estimates are subsequently obtained as $\hat{\mathbf{A}}^s = [(\mathbf{a}^s_1)^\top[k], \dots, (\mathbf{a}^s_N)^\top[k]]^\top$ and $\hat{\mathbf{B}}^s = \text{diag}(b^s_{11}[k], \dots, b^s_{NN}[k])$, for k large enough so that convergence has been attained.

Solving (P1) over the entire time horizon. Tracking the switching sequence and the state parameters entails sequentially alternating between two operations per datum arrival. First, $\hat{\sigma}(t)$ is estimated (via solving (3.13) or (3.14)), and the corresponding values $\{\hat{\chi}_{ts}\}_{s=1}^S$ are accordingly obtained. The iteration steps (3.21)-(3.24) are then run until convergence is attained. Note that these iterations only depend on past data through recursively updated moving averages, that is,

$$\mathbf{\Omega}^s_t = \mathbf{\Omega}^s_{t-1} + \hat{\chi}_{ts}\mathbf{Y}_t\mathbf{Y}_t^\top, \quad \bar{\mathbf{Y}}^s_t = \bar{\mathbf{Y}}^s_{t-1} + \hat{\chi}_{ts}\mathbf{Y}_t. \tag{3.26}$$

Similar recursive expressions can be readily derived for $\bar{\mathbf{Y}}^s_{-i,t}, \alpha^s_t$, and $\boldsymbol{\omega}^s_{-i,t}$. The complexity in evaluating the Gram matrix $\mathbf{Y}_t\mathbf{Y}_t^\top$ dominates the per-iteration computational cost of the algorithm. The recursive updates in (3.26) are conducted only for a single state $s = \hat{\sigma}(t)$ per interval t, with the remainder $\{\mathbf{\Omega}^s_t, \bar{\mathbf{Y}}^s_t\}_{s\in\mathcal{S}\setminus\hat{\sigma}(t)}$ set to $\{\mathbf{\Omega}^s_{t-1}, \bar{\mathbf{Y}}^s_{t-1}\}_{s\in\mathcal{S}\setminus\hat{\sigma}(t)}$. Similarly, iterations (3.21)-(3.24) are run only for $s = \hat{\sigma}(t)$, while $\{\hat{\mathbf{A}}^s, \hat{\mathbf{B}}^s\}_{s\in\mathcal{S}\setminus\hat{\sigma}(t)}$ are not updated during interval t. Furthermore, the need to recompute L_f per t can be circumvented by selecting an appropriate step-size for (3.23)-(3.24) by line search [108].

Algorithm 7 summarizes the developed state-dependent topology tracking scheme. Numerical tests indicate that $3-5$ inner ISTA iterations suffice to track the evolving topology remarkably well.

3.5.1 Initialization of Algorithm 7

In order to run Algorithm 7, one needs initial state estimates $\{\mathbf{A}^s_0, \mathbf{B}^s_0\}_{s=1}^S$. If $\mathbf{X}$ is known to have full row rank, Algorithm 6 can be used to initialize the state matrices as the S cluster centroids. However, this is quite restrictive, and a more general initialization scheme can be obtained with less

Algorithm 7 Switched topology tracking algorithm

Require: $\{\mathbf{Y}_t\}_{t=1}^T$, $\mathbf{X}$, S, $\{\lambda_s\}_{s=1}^S$

Initialize $\{\hat{\mathbf{A}}_0^s,\ \hat{\mathbf{B}}_0^s,\ \bar{\mathbf{Y}}_0^s = \mathbf{0}_{N\times C}\}_{s=1}^S$
for $t = 1, \ldots, T$ **do**
  $\hat{\sigma}(t) = \arg\min_{s\in\mathcal{S}} \|\mathbf{Y}_t - \hat{\mathbf{A}}_{t-1}^s \mathbf{Y}_t - \hat{\mathbf{B}}_{t-1}^s \mathbf{X}\|_F$
  Set $\hat{\chi}_{t\hat{\sigma}(t)} = 1$ and $\hat{\chi}_{ts} = 0$ for $s \in \mathcal{S} \setminus \hat{\sigma}(t)$
  for $s = 1, \ldots, S$ **do**
    if $s = \hat{\sigma}(t)$ **then**
      Update $\mathbf{\Omega}_t^s, \bar{\mathbf{Y}}_t^s, \alpha_t^s, L_f$
      Set $\mathbf{A}^s[0] = \hat{\mathbf{A}}_{t-1}^s$, $\mathbf{B}^s[0] = \hat{\mathbf{B}}_{t-1}^s$, $k = 0$
      while not converged **do**
        for $i = 1 \ldots N$ (in parallel) **do**
          $\mathbf{z}^s[k] = \mathbf{a}_{-i}^s[k] - (1/L_f)\nabla_{\mathbf{a}_{-i}^s} f[k]$
          $\mathbf{a}_{-i}^s[k+1] = \text{soft}_{\lambda_s/L_f}(\mathbf{z}^s[k])$
          $b_{ii}^s[k+1] = b_{ii}^s[k] - (1/L_f)\nabla_{b_{ii}^s} f[k]$
          Update $\mathbf{a}_i^s[k+1]$ via (3.25)
        end for
        $k = k + 1$
      end while
      $\hat{\mathbf{A}}_t^s = \mathbf{A}^s[k], \hat{\mathbf{B}}_t^s = \mathbf{B}^s[k]$
    else
      $\hat{\mathbf{A}}_t^s = \hat{\mathbf{A}}_{t-1}^s, \hat{\mathbf{B}}_t^s = \hat{\mathbf{B}}_{t-1}^s$
      $\alpha_t^s = \alpha_{t-1}^s, \mathbf{\Omega}_t^s = \mathbf{\Omega}_{t-1}^s, \bar{\mathbf{Y}}_t^s = \bar{\mathbf{Y}}_{t-1}^s$
    end if
  end for
end for
return $\{\hat{\mathbf{A}}_t^s, \hat{\mathbf{B}}_t^s\}_{s=1}^S, \{\hat{\sigma}(t)\}_{t=1}^T$

stringent restrictions on $\mathbf{X}$. For example, the following regularized LS estimator with $\mu > 0$

$$\underset{\{\mathbf{A},\mathbf{B}:a_{ii}=0,b_{ij}=0\}}{\arg\min} \quad (1/2)\|\mathbf{Y}_t - \mathbf{A}\mathbf{Y}_t - \mathbf{B}\mathbf{X}\|_F^2 + \mu\|\mathbf{A}\|_F^2 \tag{3.27}$$

yields estimates $\{\hat{\mathbf{A}}_t, \hat{\mathbf{B}}_t\}_{t=1}^{T_{\text{train}}}$ per t for a designated training interval $t = 1, \ldots, T_{\text{train}}$ ($T_{\text{train}} \ll T$). The initializations $\{\mathbf{A}_0^s, \mathbf{B}_0^s\}_{s=1}^S$ are then obtained as the S cluster centroids of $\{\hat{\mathbf{A}}_t, \hat{\mathbf{B}}_t\}_{t=1}^{T_{\text{train}}}$. Note that (3.27) decouples across nodes, and amounts to solving

$$\underset{\mathbf{a}_{-i},b_{ii}}{\arg\min} \quad (1/2)\|\mathbf{y}_{i,t} - \mathbf{Y}_{-i,t}^\top \mathbf{a}_{-i} - b_{ii}\mathbf{x}_i\|_2^2 + \mu\|\mathbf{a}_{-i}\|_2^2 \tag{3.28}$$

for $i = 1, \ldots, N$. Indeed, (3.28) admits the following per-variable closed-form solutions

$$\mathbf{a}_{-i} = \left(\mathbf{Y}_{-i,t}\mathbf{Y}_{-i,t}^\top + 2\mu\mathbf{I}\right)^{-1} \mathbf{Y}_{-i,t}\left(\mathbf{y}_{i,t} - b_{ii}\mathbf{x}_i\right) \tag{3.29}$$

and

$$b_{ii} = \frac{\left(\mathbf{a}_{-i}^\top \mathbf{Y}_{-i,t} - \mathbf{y}_{i,t}^\top\right)\mathbf{x}_i}{\|\mathbf{x}_i\|_2^2} \tag{3.30}$$

for $i = 1, \ldots, N$. Starting with an initial value for b_{ii}, one can compute (3.29) and (3.30) in an alternating fashion by fixing one variable and updating the other, until convergence is attained.

3.5.2 Tracking slowly-changing state matrices

It has tacitly been assumed that state matrices $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$ are static, and that temporal dynamics only arise from the switching sequence $\{\sigma(t)\}_{t=1}^T$. Nevertheless, entries of each of the state matrices may drift slowly over time, motivating algorithms that down-weigh the influence of past data. To this end, (P1) can be modified as follows

$$\begin{aligned} \text{(P2)} \quad \underset{\mathbf{A}^s,\mathbf{B}^s}{\arg\min} \quad & \frac{1}{2}\sum_{\tau=1}^{t} \beta^{t-\tau}\hat{\chi}_{\tau s}\|\mathbf{Y}_\tau - \mathbf{A}^s\mathbf{Y}_\tau - \mathbf{B}^s\mathbf{X}\|_F^2 + \lambda_s\|\mathbf{A}^s\|_1 \\ \text{s.t.} \quad & a_{ii}^s = 0,\ b_{ij}^s = 0,\ \forall i \neq j \end{aligned} \tag{3.31}$$

where $\beta \in (0, 1]$ is a forgetting factor that exponentially down-weighs past data whenever $\beta < 1$. In order to solve P2, Algorithm 7 can be readily modified in a reasonably straight-forward manner, and details are omitted here for brevity.

3.6 Numerical Tests

3.6.1 Synthetic data

Data generation. To assess the performance of the developed algorithms, the first set of experiments were conducted on synthetic cascade data. Four Kronecker graphs were generated from the following seed matrices [84]

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \mathbf{H}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{H}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \mathbf{H}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The resulting graph adjacency matrices $\{\mathbf{A}^s \in \mathbb{R}^{64\times 64}\}_{s=1}^4$, each encoding a network of $N = 64$ nodes, were obtained by repeated Kronecker products $\mathbf{A}^s = \mathbf{H}_s \otimes \mathbf{H}_s \otimes \mathbf{H}_s$ for $s = 1, \ldots, 4$. Each diagonal entry of $\mathbf{A}^s$ was set to zero, while the number of cascades was set to $C = 80$, and time intervals to $T = 1,000$. Furthermore, $\mathbf{X} \in \mathbb{R}^{N\times C}$ was constructed with entries sampled from a uniform distribution as $[\mathbf{X}]_{ij} \sim \mathcal{U}[0, 3]$. Similarly, the diagonal matrices $\{\mathbf{B}^s \in \mathbb{R}^{N\times N}\}_{s=1}^4$ were constructed by sampling entries from a uniform distribution as $[\mathbf{B}]_{ii} \sim \mathcal{U}[0, 1]$. Synthetic cascade data were then generated as $\mathbf{Y}_t = (\mathbf{I}_N - \mathbf{A}^{\sigma(t)})^{-1}(\mathbf{B}^{\sigma(t)}\mathbf{X} + \mathbf{E}_t)$, with $\sigma(t)$ sampled uniformly at random from $\mathcal{S} = \{1, 2, 3, 4\}$ per t, and $[\mathbf{E}_t]_{ij} \sim \mathcal{N}(0, 0.01)$, for $t = 1, \ldots, T$. Figure 3.2 depicts the ground-truth adjacency matrices $\{\mathbf{A}^s\}_{s=1}^4$ used to generate the synthetic cascade data.

Experimental results. First, Algorithm 6 was run with $T_{\text{train}} = 200$, and k-means as the clustering algorithm of choice. Figure 3.3 depicts heatmaps of the resulting adjacency matrices obtained as cluster centroids. Note that Algorithm 6 is devoid of a data-driven thresholding scheme, hence most entries in the recovered adjacency matrix are nonzero. Nevertheless, visual inspection of the heatmaps reveals that larger entries generally correspond to non-zero edge weights as shown in

Figure 3.2. Since the synthetic cascade data are noisy, it is not surprising that Algorithm 6 exhibits suboptimal topology identification. In fact as demonstrated next, Algorithm 7 is markedly superior to Algorithm 6 with respect to coping with noise, as well as shrinking non-edge entries of $\{\mathbf{A}^s\}_{s=1}^4$ to zero.

Using the initial 50 cascade samples $\{\mathbf{Y}_t\}_{t=1}^{50}$, per-interval state-agnostic batch estimates $\{\hat{\mathbf{A}}_t, \hat{\mathbf{B}}_t\}_{t=1}^{50}$ were obtained by solving (3.27) with $\mu = 0.01$. The estimates $\{\hat{\mathbf{A}}_t, \hat{\mathbf{B}}_t\}_{t=1}^{50}$ were then clustered via the k-means algorithm, with $k = 4$ clusters, and the corresponding cluster centers were used as the initial values $\{\hat{\mathbf{A}}_0^s, \hat{\mathbf{B}}_0^s\}_{s=1}^4$. Algorithm 7 was then run for $t = 51, \ldots, 1,000$, with $\lambda_s = 0.95$, and $s \in \mathcal{S}$. Figure 3.4 depicts heatmaps of the recovered state-dependent adjacency matrices at $T = 1,000$. Thanks to sparsity regularization, Algorithm 7 unveils the non-zero support structures of the state matrices with remarkable success. Figure 3.5 plots the actual and estimated switching sequences from $t = 901$ to $t = 1,000$, clearly demonstrating the remarkable success in tracking the underlying states.

Next, the advocated approach was compared with earlier work which models temporal information cascades using dynamic SEMs, with slow topology variations [20]. A stochastic gradient descent (SGD) algorithm developed in [20] was run with batch initialization on the time-series of cascade data. Per interval t, the resulting relative estimation error, $(\|\mathbf{A}^t - \hat{\mathbf{A}}^t\|_F + \|\mathbf{B}^t - \hat{\mathbf{B}}^t\|_F)/(\|\hat{\mathbf{A}}^t\|_F + \|\hat{\mathbf{B}}^t\|_F)$, was evaluated for both Algorithm 7 and the SGD tracker from [20]. Note that in computing the estimation error for Algorithm 7, $\mathbf{A}^t = \mathbf{A}^{\sigma(t)}, \mathbf{B}^t = \mathbf{B}^{\sigma(t)}$ (similarly for $\hat{\mathbf{A}}^t$ and $\hat{\mathbf{B}}^t$). Figure 3.6 (a) compares the per-interval relative errors of the two approaches. It is clear from the plot that exploiting the prior knowledge that network dynamics arise due to random switching between $S = 4$ states yields remarkably superior error performance than the alternative. In fact, the final adjacency matrix $\hat{\mathbf{A}}^T$ does not correspond to a specific state, but is rather an average of the underlying state matrices. However, this is not surprising since the framework advocated in [20] exploits slow topology variations, and it is not expected to outperform algorithms developed in this chapter, when topologies potentially jump suddenly between discrete states.

To this end, a new slowly-varying state sequence was used to control the evolving network topologies, and a new time series of synthetic cascades was generated. Specifically, the following

scheme was used to generate a piecewise-constant sequence $\{\sigma(t)\}_{t=1}^{T}$:

$$\sigma(t) = \begin{cases} 1, & t \in \{\{1, \ldots, 24\} \cup \{200, \ldots, 299\}\} \\ 2, & t \in \{\{25, \ldots, 49\} \cup \{300, \ldots, 699\}\} \\ 3, & t \in \{\{50, \ldots, 74\} \cup \{700, \ldots, 899\}\} \\ 4, & t \in \{\{75, \ldots, 199\} \cup \{900, \ldots, 1,000\}\} \end{cases}$$

while the ground-truth matrices $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^{S}$ and $\{\mathbf{E}_t\}_{t=1}^{T}$ remained unchanged. Figure 3.6 (a) compares per-interval relative errors resulting from running Algorithm 7, and the SGD algorithm developed in [20]. In this case, there are fewer sudden transitions between network states. Nevertheless, leveraging the prior information about the state-based network evolution is beneficial as shown by the plot. Indeed, modeling cascade propagation by the proposed switched dynamic SEM framework leads to better error performance than adopting a state-incognizant approach that exploits the underlying slow variations.

3.6.2 Real cascade data

Dataset description. This section presents results of experimental tests conducted on real cascades observed on the web between March 2011 and February 2012. Blog posts and news articles for memes (popular textual phrases) appearing within on approximately 3.3 million websites were monitored during the observation period. The time when a given website first mentioned a story related to a specific set of memes was recorded, and the data were availed to the public from [86]. Cascade infection times were recorded as Unix timestamps in hours (i.e., the number of hours since midnight on January 1, 1970). Several trending news topics during this period were identified, and cascade data for the top $5,000$ websites that mentioned memes associated with them were retained. From this dataset, cascades related to 10 broad news topics were extracted for this chapter's experiments. Table 3.1 lists the names of these topics, with a correspondingly brief description.

For each broad topic in Table 3.1, many news stories appeared on blogs and mainstream news websites over the observation period. Each story was assigned a list of tuples in the form (website id, timestamp) capturing the time when it was first mentioned on a particular website. For this chapter,

	Broad news topic	**Brief description**
1	Fukushima	Nuclear accident at Japan's Fukushima nuclear power plant in March 2011
2	Kate Middleton	English royal whose wedding took place in April 2011
3	Kim Jong-un	Leader of North Korea who rose to prominence upon the death of his father in December 2011
4	Osama bin Laden	Infamous terrorist leader who was killed in May 2011
5	Amy Winehouse	Famous English singer who died of drug overdose in July 2011
6	Rupert Murdoch	Businessman whose media company was involved in a phone hacking scandal revealed in May 2011
7	Steve Jobs	Technology entrepreneur whose death in October 2011 was followed by many news headlines
8	Arab spring	Wide-spread politically-charged protests among several Arab nations starting in December 2010
9	Strauss Kahn	International Monetary Fund (IMF) director who resigned in May 2011 due to sex assault allegations
10	Reid Hoffman	Founder of LinkedIn whose stock started trading in May 2011

Table 3.1: The 10 broad news topics whose memes constituted the cascades tracked for topology inference.

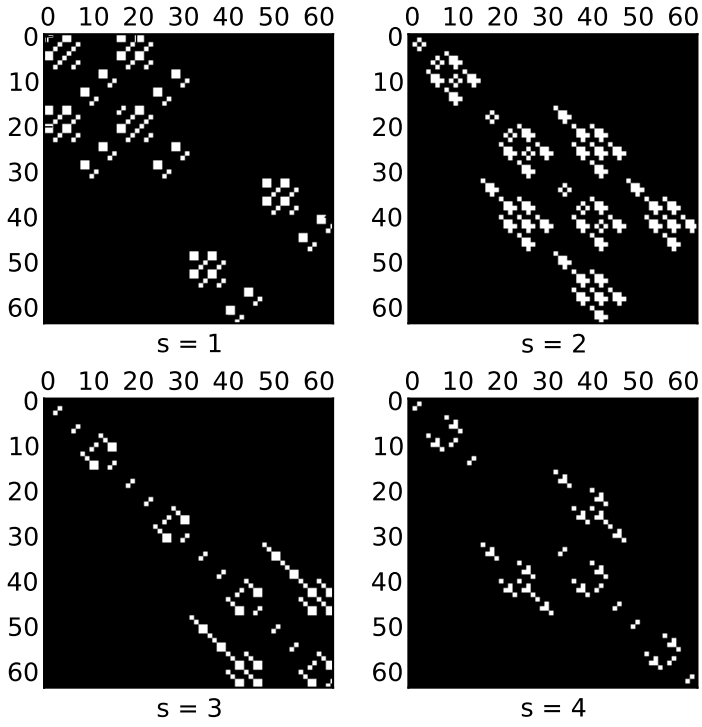


Figure 3.2: Adjacency matrices corresponding to actual switching networks used for the synthetic dataset.

a popular news story was characterized as a meme if it propagated to at least 100 websites, due to repeated mentions. Based on this definition, all news stories that did not qualify as memes were discarded, and a total of $C = 625$ cascades constituting the selected memes, and their infection times were retained. Due to this thresholding, the total number of of websites reduced to $N = 1,131$. The observation period was split into $T = 180$ time intervals $\{\mathcal{T}_t\}_{t=1}^{180}$, each corresponding to approximately 2 days. Letting u_{ic} denote the Unix infection time of node i by cascade c, datum y_{ic}^t was computed as follows:

$$y_{ic}^t = \log_{10}\left(u_{ic} - \min_{u_{ic}\in\mathcal{T}_t} u_{ic}\right) \tag{3.32}$$

only if $u_{ic} \in \mathcal{T}_t$, otherwise y_{ic}^t is set to a very large value, namely $y_{ic}^t = 2 + \log_{10}(\max_{i,c} u_{ic})$, as a surrogate for infinity.

Entries of $\mathbf{X}$ capture prior knowledge about the susceptibility of each node to each contagion. For instance, the entry x_{ic} could denote the online search rank of website i for a search keyword

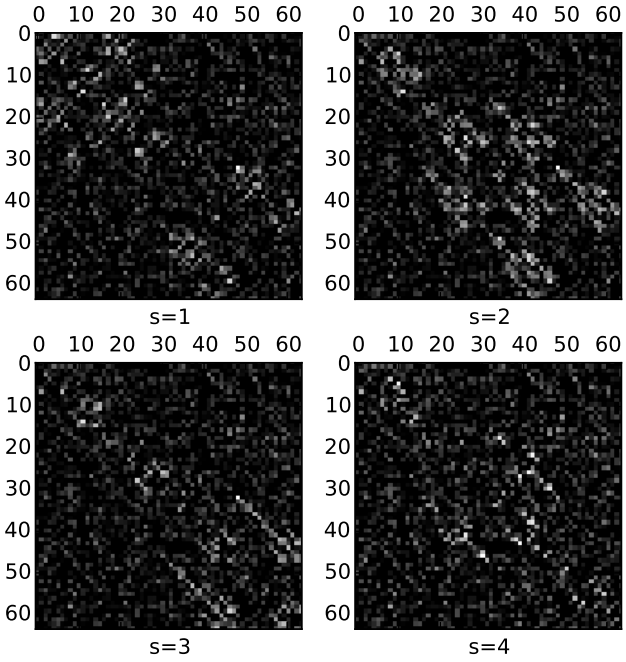


Figure 3.3: Adjacency matrices corresponding to state-dependent network topologies inferred by Algorithm 6.

associated with contagion c. In the absence of such website ranking information, this chapter resorted to an alternative approach involving assignment of susceptibilities from the entire corpus of the cascade data. First, five broad categories of memes were identified as *politics*, *entertainment*, *sports*, *business*, and *technology*, and indexed by $k = 1, \ldots, 5$. Each group of news topics was manually labeled using these five prescribed categories. Next, for each website i, γ_{ik} was computed as follows

$$\gamma_{ik} := \frac{\text{number of category } k \text{ cascades that infected node } i}{\text{total number of cascades that infected node } i}$$

for all k categories. Entry x_{ic} was then set to γ_{ik} if cascade c belonged to category k.

Experimental results. Since S is unknown for the real dataset, the first part of this experimental test entailed approximating its value from the data. Setting $\mu = 0.15$, the initialization scheme in (3.27) was run in batch mode over 60 intervals. The resulting estimates of $\{\hat{\mathbf{A}}_t, \hat{\mathbf{B}}_t\}_{t=1}^{60}$ were then clustered using S-means, with $S = 1, \ldots, 10$. For each S, the total intra-cluster distance was

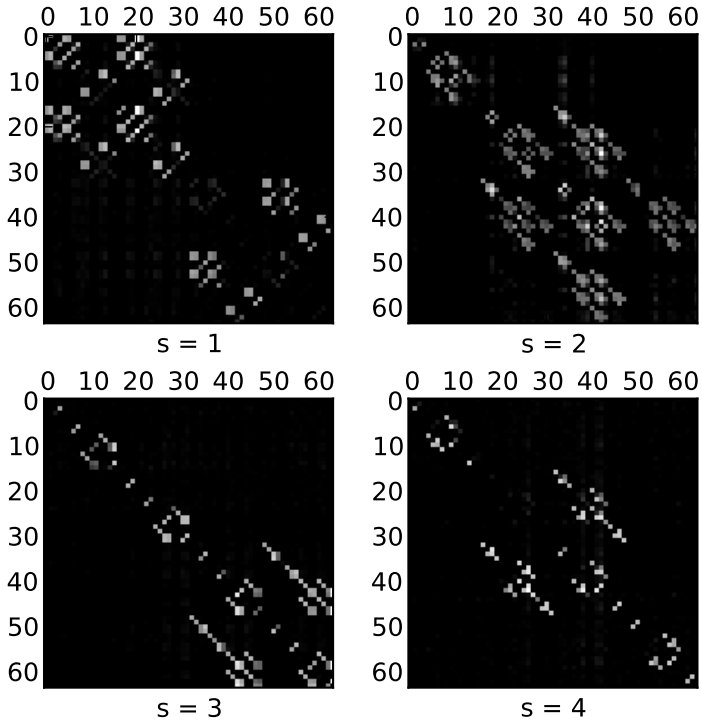


Figure 3.4: Adjacency matrices corresponding to state-dependent network topologies inferred by Algorithm 7.

evaluated upon convergence as

$$\delta(S) := \log_{10}\left\{\sum_{s=1}^{S}\sum_{t=1}^{60}\hat{\chi}_{ts}\|\hat{\boldsymbol{\theta}}^t - \hat{\boldsymbol{\theta}}^s\|_2^2\right\} \tag{3.33}$$

on a logarithmic scale, with the estimates $\hat{\chi}_{ts}, \hat{\boldsymbol{\theta}}^t$, and $\hat{\boldsymbol{\theta}}^s$ defined earlier. Figure 3.8 plots $\delta(S)$ for $S = 1, \ldots, 10$, depicting a substantial decrease from $S = 2$ to $S = 3$. Subsequent values of S do not markedly reduce $\delta(S)$. From this plot, $S = 3$ was selected as the underlying number of network states.

Next, Algorithm 7 was run with the pre-processed cascade data as inputs. Note that Algorithm 7 requires the sparsity-promoting regularization parameters $\{\lambda_s\}_{s=1}^S$ as inputs. In this experiment, a uniform value was adopted for all states, i.e., $\lambda_1 = \lambda_2 = \lambda_3 = \lambda$. In the absence of ground-truth state topologies, selection of λ is rather challenging, and it remains an open question for future work. Nevertheless, a heuristic approach inspired by typical properties of real-world large-scale networks was adopted for this chapter. Over the last $10 - 15$ years, several studies in network

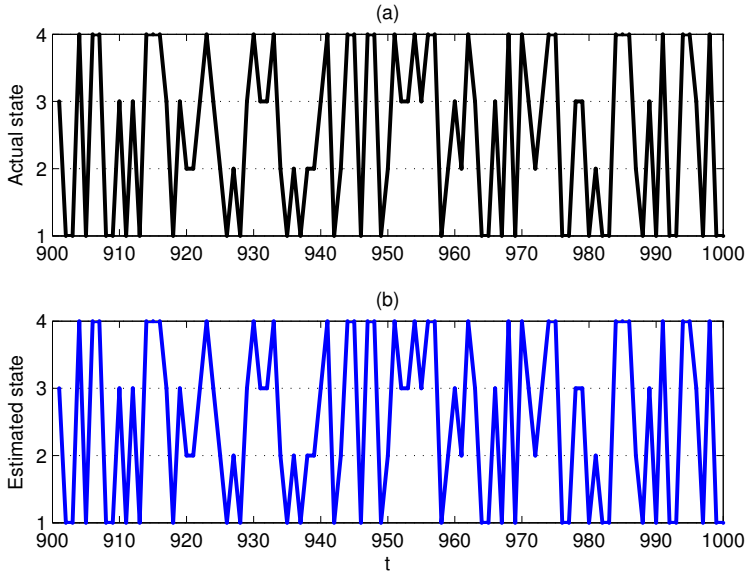
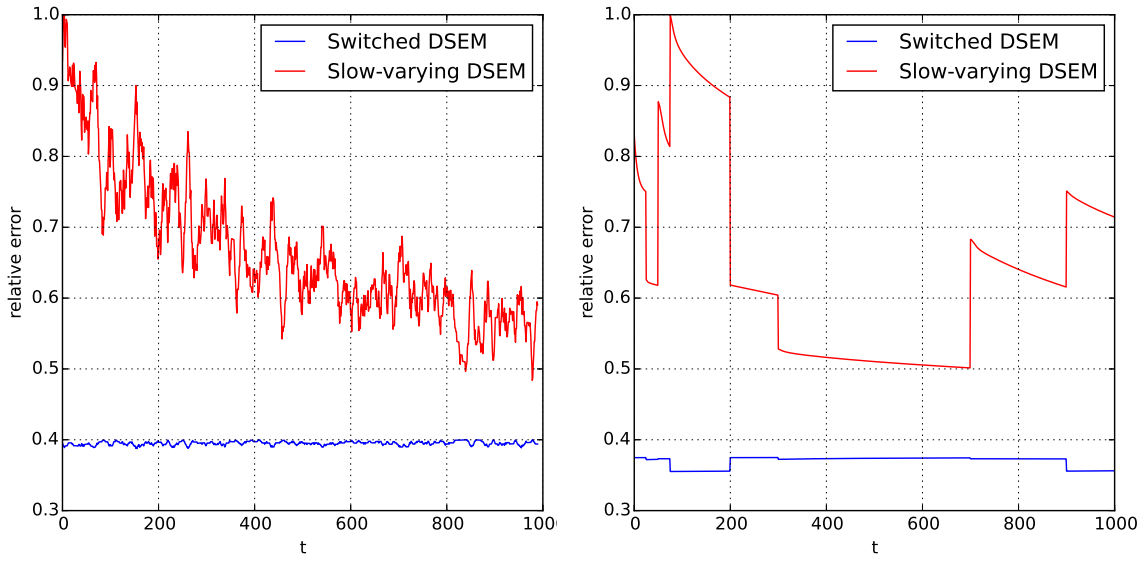


Figure 3.5: Actual (a) and estimated (b) switching sequences plotted from $t = 900$ to $t = 1,000$.



(a) (b)

Figure 3.6: Comparison of relative errors resulting from tracking the unknown topologies as a switched sequence versus slowly-varying changes: (a) rapidly switching state sequence; and (b) piecewise-constant state sequence.

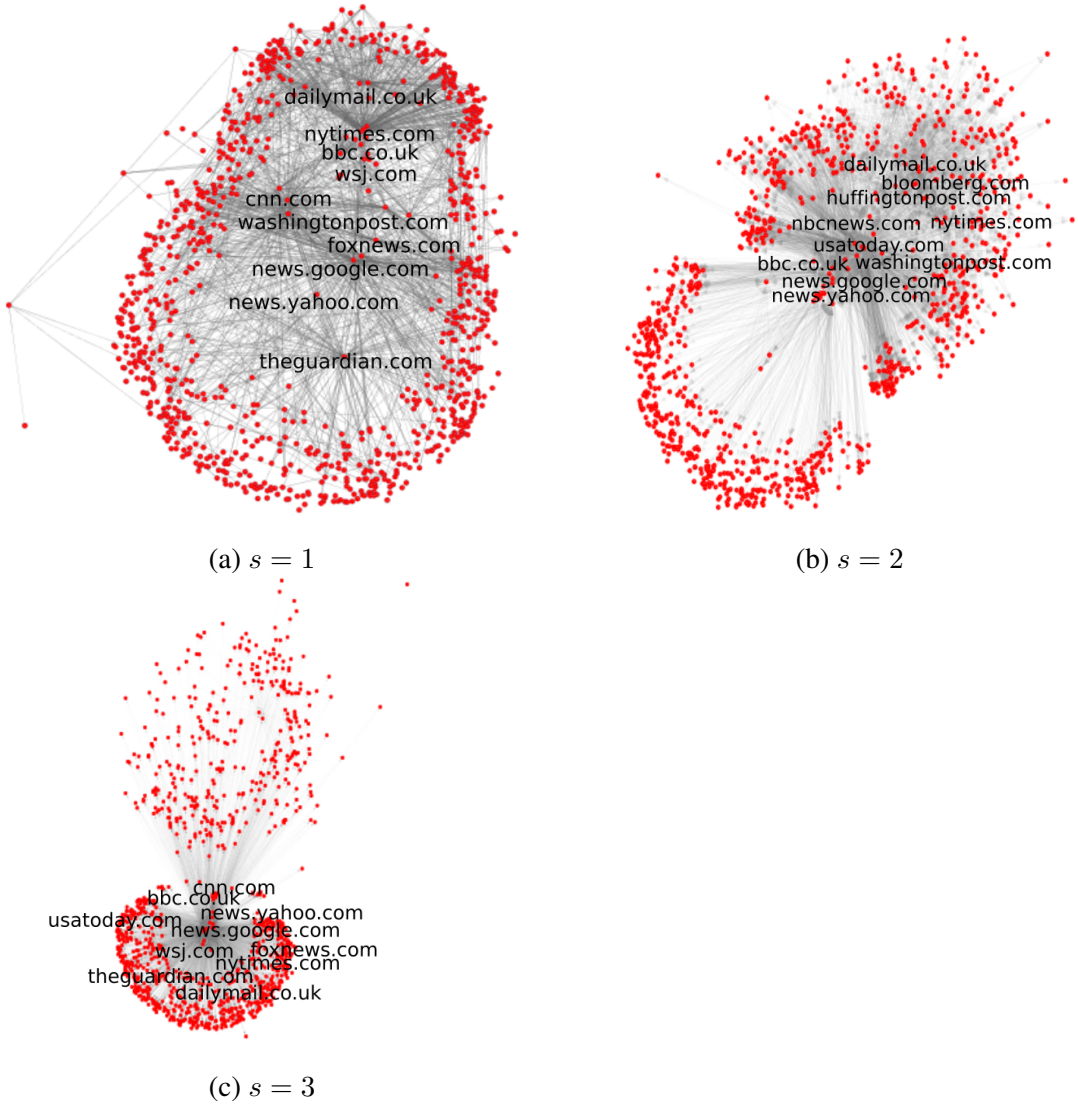
Network state	**1**	**2**	**3**
Average clustering coefficient	0.367	0.216	0.218
Network diameter	18	13	8
Average number of neighbors	134.164	81.132	25.738
Average shortest path length	9.34	7.66	4.51

Table 3.2: Simple summary statistics extracted from the inferred network state topologies with $\lambda = 10$.

science have unveiled remarkable universal properties that seem to underlie most real networks. For example, networks generally exhibit the so-termed "small-world" property, and their degree distributions often follow power laws [129].

Acknowledging these as universal laws inherent to most networks, it is possible to constrain the search space of the most likely network topologies uncovered by Algorithm 7. In this experiment, λ was selected in such a way that the average shortest path length between any pair of nodes is $\mathcal{O}(\log N)$, which is consistent with the "small-world" phenomenon. With $N = 1,131$, the goal was to select λ so that the resulting average shortest path length was within the interval $[6, 10]$. For each λ, Algorithm 7 was run and the average shortest path lengths were computed per network state, and then averaged over all states. Figure 3.9 plots the average path length as λ is varied between 0 and 100. The plot also depicts the corresponding average network diameters, which can also be used as a validation metric. It turns out that setting $\lambda = 10$ for $s \in \{1, 2, 3\}$ led to an average shortest path length of approximately 7, which is indicative of an underlying "small-world" property. Table 3.2 summarizes the per-state average clustering coefficient, network diameter, average number of neighbors, and the average shortest path length when $\lambda = 10$.

Figure 3.7 depicts visualizations of the state-dependent network topologies inferred from the cascade data using Algorithm 7. Each drawing is annotated with the top 10 websites when ranked in order of decreasing out-degree. It is interesting to note that this list is dominated by websites corresponding to the most influential news outlets in the English-speaking world e.g., *nytimes.com*, *cnn.com*, *bbc.co.uk*, and *theguardian.com*. Furthermore, it turns out that popular news aggregators, such as *news.yahoo.com* and *news.google.com*, are competitive with traditional media powerhouses in terms of influencing the propagation of news over the web. Although this analysis only considers



(a) $s = 1$ (b) $s = 2$

(c) $s = 3$

Figure 3.7: State-dependent network topologies inferred from real cascades annotated by the top 10 ranked websites, in order of decreasing out-degree.

English-based websites, these experimental results on real-world cascades are strongly corroborated by prior expectations about the most influential enablers of information propagation over the world-wide web. Finally, Figure 3.10 depicts the switching sequence for the network topologies over the entire observation period. It turns out that evolution of the network topology adheres to piecewise constant state variations.

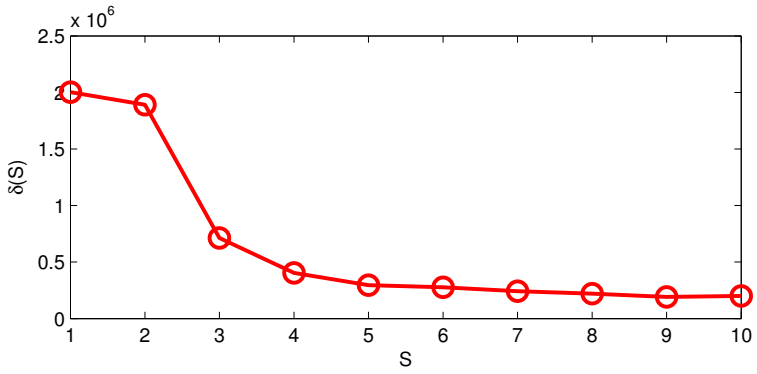


Figure 3.8: Total intra-cluster distances plotted for different values of S, with $S = 3$ selected as the optimal size of the state space in the real web cascades dataset.

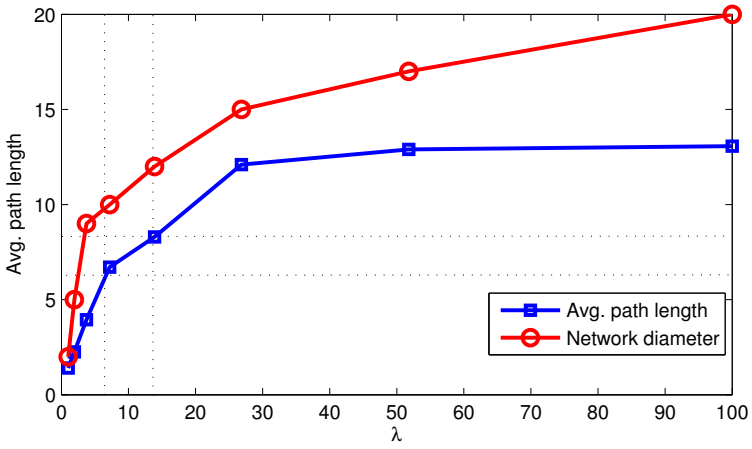


Figure 3.9: Empirical selection of λ guided by the resulting average shortest path lengths in the inferred networks. With $\lambda = 10$, the inferred network topologies exhibit the "small-world" property with average path lengths between $6 - 10$ hops.

3.7 Chapter Summary

A switched dynamic SEM was proposed for tracking the evolution of dynamic networks from information cascades when topologies arbitrarily switch between discrete states. It was shown that presence of exogenous influences is critical to guarantee model identifiability, and one can even identify the underlying network topologies with fewer cascade measurements by leveraging edge sparsity. Recognizing that identification of the unknown network states and the switching sequence

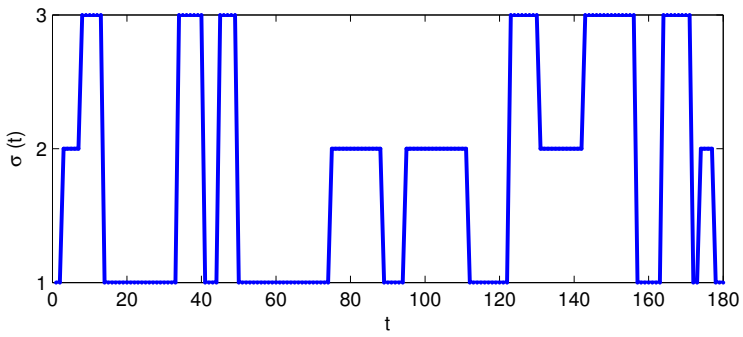


Figure 3.10: Inferred switching sequence from the real-world web cascade data. It turns out that network dynamics are governed by piecewise-constant state variations.

is computationally challenging, this chapter assumed that the number of states are known a priori, and advocated an exponentially-weighted LS estimator capitalizing on edge sparsity. A recursive two-step topology tracking algorithm leveraging advances in proximal gradient optimization was developed. Per interval, the first step estimates the active state by minimizing the a priori prediction error, while the second step recursively updates only the matrices corresponding to the estimated state.

Experiments on synthetically-generated cascades demonstrated the effectiveness of the advocated approach in jointly tracking the switching sequence, while identifying the state-dependent network topologies. Numerical tests were also conducted on real cascades of popular news over the web, collected over one year. The goal was to identify the underlying topological structure of causal influences between news websites and blogs. Interestingly, websites exhibiting the highest influence (as measured by the out-degree) turned out to be recognizable news outlets and popular news aggregators. Future directions include investigation of more efficient approaches for tracking the switching sequence, identification of the number of states, and selection of sparsity-promoting parameters, especially when ground-truth network topologies are unknown.

Chapter 4

Tracking Dynamic Communities and Anomalies

4.1 Introduction

Community identification is a well-studied task in modern network analysis [55, 59]. A community is generally characterized by existence of many edges interconnecting nodes within the community, and far fewer connecting to the rest of the network. Standard "work-horse" community detection approaches e.g., spectral clustering [90], and modularity maximization [103], are inherently limited because they assume networks are static and communities do not overlap. Furthermore, true communities are often distorted by aberrant nodes exhibiting "anomalous" behavior, often manifested as unusually strong, uni-directional edge connectivity across communities; see e.g., Figure 4.1. Examples of such anomalies include e-mail spammers, or phishing accounts in online social networks. Cognizant of the temporal nature of most networks, several recent works have focused on tracking time-varying communities [60, 87, 92, 123, 134].

Recent approaches for overlapping community discovery mostly leverage matrix factor models, and associate each node with a per-community affiliation strength a.k.a., soft clustering [92, 115, 135]. A non-negative matrix factorization (NMF) model was advocated for batch recovery of overlapping communities in time-varying networks in [92]. In general, these contemporary approaches are incognizant of distortive anomalies, which may hurt the accuracy of community assignments.

The main contribution of this chapter is a novel approach for *jointly* tracking temporal, overlapping communities, while compensating for distortive anomalies. Motivated by contemporary NMF approaches, a community affiliation model in which edge weights are generated by mutual community participation between node pairs is adopted in Section 4.2. Unlike prior methods, the proposed approach explicitly compensates for anomalies, and capitalizes on their sparsity, while leveraging the low rank property induced by the presence of a few communities with respect to the number of nodes. Furthermore, the network time series is assumed to evolve slowly. Under these conditions, a novel, sparsity-promoting, rank-regularized, exponentially-weighted least-squares estimator is put forth in Section 4.3. Leveraging advances in proximal splitting approaches (see e.g., [34]), computationally efficient community tracking algorithms, based on alternating minimization are developed.

In order to appeal to big data contexts, within which most social networks of interest arise, a number of algorithmic modifications are considered. In such settings, batch approaches are impractical due to prohibitive memory costs. Moreover, approximate fast solutions are preferrable to slow exact ones. Towards facilitating real-time, memory-efficient operation in streaming data settings, an online algorithm leveraging stochastic gradient descent iterations is developed in Section 4.4.2. Moreover, certain practical settings entail storage of network data across many clusters of computing nodes, possibly geographically located at different sites. In such scenarios, tracking communities in a decentralized fashion is well-motivated. To this end, a tracking algorithm that leverages the alternating direction method of multipliers (ADMM) is developed in Section 4.5. Numerical tests with synthetically-generated networks demonstrate the effectiveness of the developed algorithms in tracking communities and anomalies (Section 4.6.1). Further experiments in Section 4.6.2 are conducted on real data, extracted from global trade flows among nations between 1870 and 2009.

4.2 Model and Problem Statement

4.2.1 Community affiliation model

Consider a dynamic directed N-node network whose time-varying topology is captured by the time-series of adjacency matrices $\{\mathbf{A}^t \in \mathbb{R}^{N\times N}\}_{t=1}^{T}$. Entry (i,j) of $\mathbf{A}^t$ (hereafter denoted by a_{ij}^t) is

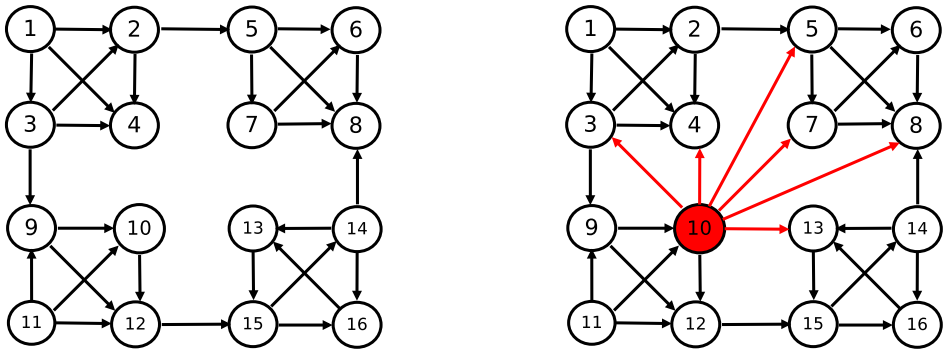


Figure 4.1: An illustration of the distortive effect of anomalous nodes: a) A 16-node directed network with four easily-discernible communities; and b) The same network with node 10 exhibiting an unusually high number of outgoing edges. Identifying the underlying communities is more challenging, as a result of the distortion caused by node 10.

nonzero only if an edge originating from node i is connected to node j during interval t. It is assumed that edge weights are nonnegative, namely $a_{ij}^t \geq 0$. Suppose that the network consists of C unknown communities which are allowed to overlap, that is a node can belong to one or more communities simultaneously. This is motivated by practical settings where e.g., a Facebook user may be associated with multiple communities consisting of her work colleagues, former schoolmates, or friends from the local sports club. It can be reasoned that the likelihood of two people becoming friends is directly related to the number of communities to which they mutually belong. For example, if two work colleagues happen to have attended the same high school, then chances are high that they will become friends. This a fortiori argument based upon a reasonably natural observation in social settings lies at the foundation for several recent community affiliation models for edge generation [92, 135].

Suppose $\mathbf{V}^t := [\mathbf{v}_1^t \dots \mathbf{v}_C^t] \in \mathbb{R}^{N \times C}$ denotes a temporal basis matrix whose columns span a linear subspace of dimension C during observation interval t. Associating each basis vector with one of C communities, the edge vector associated with each node i can be expressed as a linear combination of $\{\mathbf{v}_c^t\}_{c=1}^C$, i.e.,

$$\mathbf{a}_i^t = \mathbf{V}^t \mathbf{u}_i^t + \mathbf{e}_i^t, \quad i = 1, \dots, N \tag{4.1}$$

where $\left(\mathbf{a}_i^t\right)^\top$ denotes row i of $\mathbf{A}^t$, and $\mathbf{e}_i^t$ captures unmodeled dynamics. Entries of $\mathbf{u}_i^t :=$

$[u_{i1}^t \dots u_{iC}^t]^\top \in \mathbb{R}^C$ in (4.1) assign community affiliation strengths, with $u_{ic}^t = 0$ only if node i does not belong to community c during interval t. Since $a_{ij}^t \geq 0$, entries of both $\mathbf{V}^t$ and $\mathbf{u}_i^t$ will be constrained to nonnegative values. Collecting edge vectors for all nodes, (4.1) can be expressed equivalently as the following canonical NMF model

$$\mathbf{A}^t = \mathbf{U}^t(\mathbf{V}^t)^\top + \mathbf{E}^t \tag{4.2}$$

where $(\mathbf{U}^t)^\top := [\mathbf{u}_1^t \dots \mathbf{u}_N^t]$, and $(\mathbf{E}^t)^\top := [\mathbf{e}_1^t \dots \mathbf{e}_N^t]$. The asymmetry inherent to (4.2) generalizes traditional approaches (e.g., spectral clustering), rendering them capable of capturing communities in weighted, directed, and even bipartite graphs. The special case in which edges are undirected (i.e., $a_{ij}^t = a_{ji}^t$) can be readily realized $\mathbf{U}^t \equiv \mathbf{V}^t$.

Contemporary community detection approaches overwhelmingly focus on unipartite networks, whereby edges are allowed to exist between any pair of nodes. On the other hand, directional edges in bipartite networks only connect nodes belonging to two distinct classes. Examples include buyer-seller networks existing in online applications like E-bay, or recommender networks with edges capturing ratings of products by customers. With matrices $\mathbf{U}^t \in \mathbb{R}^{N\times C}$, and $\mathbf{V}^t \in \mathbb{R}^{M\times C}$, communities in an NM-node bipartite network (N nodes belonging to one class, and M to the other) can be readily captured through the affiliation model (4.2).

Unfortunately, (4.1) does not effectively capture anomalous nodes that exhibit unusually strong affiliation to one or more communities. This aberrant behavior has been observed in several real-world networks, and it often arises due to any of several reasons. For example, the presence of "fake" user accounts in online social networks created for phishing purposes from unsuspecting peers may lead to abnormally high numbers of outgoing links. In addition, fraudulent reviewers in web-based rating applications may exhibit abnormal affiliation while unfairly promoting their services to specific communities. Regardless of the underlying reason, detection of such nodes is envisioned as a source of strategic information for network operators. Moreover, identification of anomalies is critical for improved community detection accuracy; see Figure 4.1 for an example where an anomalous node distorts the underlying community structure.

4.2.2 Outlier-aware community affiliation model

Suppose node i is considered anomalous, exhibiting an abnormally strong level of affiliation in one or more of the C communities. In order to preserve the estimation accuracy of community discovery algorithms, one is motivated to modify (4.1) so that such outliers are accounted for. the present chapter postulates the following robust edge generation model

$$\mathbf{a}_i^t = \mathbf{V}^t(\mathbf{u}_i^t + \mathbf{o}_i^t) + \mathbf{e}_i^t, \;\; i = 1, \ldots, N \tag{4.3}$$

where $\mathbf{o}_i^t := [o_{i1}^t \ldots o_{iC}^t]^\top$, and $o_{ic}^t \neq 0$ only if node i exhibits anomalous affiliation in community c. Intuitively, (4.3) reasonably suggests that one can investigate whether any node is an anomaly or not by introducing more variables that compensate for the effect of outliers on the edge generation model. Since outliers are rare by definition, vectors $\{\mathbf{o}_i^t\}_{i=1}^N$ are generally sparse, and this prior knowledge can be exploited to recover the unknowns.

Letting $(\mathbf{O}^t)^\top := [\mathbf{o}_1^t \ldots \mathbf{o}_N^t]$, the outlier-aware community affiliation model in (4.2) can be written as

$$\mathbf{A}^t = (\mathbf{U}^t + \mathbf{O}^t)(\mathbf{V}^t)^\top + \mathbf{E}^t, \;\; t = 1, 2, \ldots. \tag{4.4}$$

where $\mathbf{O}^t$ is sparse. In static scenarios with $\mathbf{A} = (\mathbf{U} + \mathbf{O})\mathbf{V}^\top$, setting $\mathbf{O} = \mathbf{0}$ yields the canonical NMF model for community discovery. Given $\{\mathbf{A}^t\}_{t=1}^T$, the goal of this chapter is to track the community affiliation matrices $\{\mathbf{U}^t, \mathbf{V}^t\}_{t=1}^T$, as well as outliers captured through matrices $\{\mathbf{O}^t\}_{t=1}^T$.

It is worth reiterating that (4.4) is a heavily under-determined model, and the only hope to recover $\{\mathbf{U}^t, \mathbf{O}^t, \mathbf{V}^t\}_{t=1}^T$ lies in exploiting prior information about the structure of the unknowns. Indeed, the estimator advocated in the sequel will capitalize on sparsity, low rank, and the slow evolution of networks. Introducing extra variables to capture outliers has been used in different contexts; see e.g., [58] and references therein.

Remark 5 (Measurement outliers)**.** Model 4.4 is motivated by anomalous nodes, whose presence leads to distorted community structures in e.g., social networks. A slight variation of this problem arises in cases where one is interested in identifying which edges are anomalous. This is well-motivated in settings where edge weights are directly measured, and encode valuable information e.g., star ratings in online review systems. The outliers in such cases are the result of faulty measurements, bad data (e.g., skewed user ratings), or data corruption. To detect anomalous edge weights,

one can postulate that [cf. (4.4)]

$$\mathbf{A}^t = \mathbf{U}^t(\mathbf{V}^t)^\top + \mathbf{O}^t + \mathbf{E}^t, \;\; t = 1, 2, \ldots. \tag{4.5}$$

where $\mathbf{O}^t$ is sparse, and can be effectively recovered by leveraging sparsity-promoting estimation approaches; see e.g., [58] and references therein.

4.3 Community Tracking Algorithm

This section assumes that the following hold: a1) $\mathbf{O}^t$ is sparse; a2) $\mathbf{U}^t(\mathbf{V}^t)^\top$ is low rank; and a3) $\{\mathbf{A}^t\}_{t=1}^T$ evolve slowly over time, that is, the sequence of matrices $\{\mathbf{A}^t - \mathbf{A}^{t-1}\}_{t=1}^T$ are sparse. In order to justify a1, note that the set of anomalous nodes is much smaller than that of "ordinary" nodes. On the other hand, a2 results from requiring that $\text{rank}(\mathbf{U}^t(\mathbf{V}^t)^\top) \leq C \ll N$, while a3 is motivated by observations of the evolution of most real-world networks. In the sequel, a sequential estimator that exploits a1-a3 is put forth.

4.3.1 Exponentially-weighted least-squares estimator

Suppose data are acquired sequentially over time, and storage memory is limited; thus, it is impractical to aim for batch estimation. Under the aforementioned assumptions on the community affiliation model (4.4), the following sparsity-promoting, rank-regularized, and exponentially-weighted least-squares (EWLS) estimator can track the unknown matrices

$$\begin{aligned}\{\hat{\mathbf{U}}^t, \hat{\mathbf{V}}^t, \hat{\mathbf{O}}^t\} \;\; = \;\; & \underset{\{\mathbf{U},\mathbf{V},\mathbf{O}\}\in\mathbb{R}_+^{N\times C}}{\arg\min} \;\; \sum_{\tau=1}^{t} \beta^{t-\tau} \left\| \mathbf{A}^\tau - (\mathbf{U}+\mathbf{O})\mathbf{V}^\top \right\|_F^2 \\ & + \lambda_t \|\mathbf{U}\mathbf{V}^\top\|_* + \mu_t \|\mathbf{O}\|_0 \end{aligned} \tag{4.6}$$

where the nuclear norm $\|\mathbf{U}\mathbf{V}^\top\|_* := \sum_n \sigma_n(\mathbf{U}\mathbf{V}^\top)$ sums the singular values of $\mathbf{U}\mathbf{V}^\top$, while $\|\mathbf{O}\|_0 := \sum_{ik} \mathbb{1}_{\{o_{ik}\neq 0\}}$ counts the non-zero entries in $\mathbf{O}$. Regularization parameters $\lambda_t \geq 0$ and $\mu_t \geq 0$ control the low rank of $\mathbf{U}\mathbf{V}^\top$ and sparsity in $\mathbf{O}$, respectively. Finally, $\beta^{t-\tau}$ is a "forgetting" factor with $\beta \in (0, 1]$, which facilitates tracking slow variations by down-weighing past data when $\beta < 1$.

Problem (4.6) is non-convex and NP-hard to solve. Nevertheless, this can be circumvented by resorting to tight convex relaxation. Specifically, $\|\mathbf{O}\|_0$ can be surrogated with $\|\mathbf{O}\|_1 := \sum_{ic} |o_{ic}|$, and one can leverage the following characterization of the nuclear norm; see e.g., [93]

$$\|\mathbf{Z}\|_* := \min_{\mathbf{U},\mathbf{V}\in\mathbb{R}^{N\times C}} \frac{1}{2}\Big\{\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2\Big\} \;\text{ s.t. }\; \mathbf{Z} = \mathbf{U}\mathbf{V}^\top \tag{4.7}$$

which leads to the following optimization problem

$$\begin{aligned}\{\hat{\mathbf{U}}^t, \hat{\mathbf{V}}^t, \hat{\mathbf{O}}^t\} \;=\; \underset{\{\mathbf{U},\mathbf{V},\mathbf{O}\}\in\mathbb{R}_+^{N\times C}}{\arg\min} \;& \sum_{\tau=1}^{t} \beta^{t-\tau}\Big\|\mathbf{A}^\tau - (\mathbf{U}+\mathbf{O})\mathbf{V}^\top\Big\|_F^2 \\ & + \frac{\lambda_t}{2}\Big\{\|\mathbf{U}\Big\|_F^2 + \|\mathbf{V}\|_F^2\Big\} + \mu_t\|\mathbf{O}\|_1 \end{aligned} \tag{4.8}$$

whose separability renders it amenable to alternating minimization (AM) strategies, as discussed next.

Remark 6 (Sparsity and low rank)**.** The estimator (4.8) capitalizes on sparsity and low rank properties inherent to the matrix decomposition in (4.4). One of the contributions of the present chapter is to recognize this structure in evolving community affiliations in dynamic networks. Nevertheless, a number of prior studies have leveraged sparsity and low rank in a variety of other contexts; see e.g., [45, 94, 137].

4.3.2 Alternating minimization

Focusing on the first term of the cost in (4.8), note that

$$\begin{aligned}\sum_{\tau=1}^{t} \beta^{t-\tau}\Big\|\mathbf{A}^\tau - (\mathbf{U}+\mathbf{O})\mathbf{V}^\top\Big\|_F^2 = s^t\mathrm{Tr}\Big\{\mathbf{V}(\mathbf{U}^\top\mathbf{U}+\mathbf{O}^\top\mathbf{O})\mathbf{V}^\top + 2\mathbf{V}\mathbf{U}^\top\mathbf{O}\mathbf{V}^\top\Big\} \\ - 2\mathrm{Tr}\Big\{(\mathbf{S}^t)^\top(\mathbf{U}+\mathbf{O})\mathbf{V}^\top\Big\}\end{aligned} \tag{4.9}$$

where $s^t := (1-\beta^t)/(1-\beta)$ and $\mathbf{S}^t := \mathbf{A}^t + \beta\mathbf{S}^{t-1}$ recursively accumulate past data with minimal storage requirements. Since (4.8) is separable across the optimization variables, one can resort to iterative AM, by alternately solving for each variable while holding the others fixed. With

$$\Phi\left(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{S}^t,s^t\right) := \sum_{\tau=1}^{t} \beta^{t-\tau}\Big\|\mathbf{A}^\tau - (\mathbf{U}+\mathbf{O})\mathbf{V}^\top\Big\|_F^2 \tag{4.10}$$

AM iterations amount to the following per-interval updates

$$\mathbf{U}[k] = \underset{\mathbf{U}\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \Phi\left(\mathbf{U}, \mathbf{V}[k-1], \mathbf{O}[k-1], \mathbf{S}^t, s^t\right) + (\lambda_t/2)\,\|\mathbf{U}\|_F^2 \tag{4.11a}$$

$$\mathbf{V}[k] = \underset{\mathbf{V}\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \Phi\left(\mathbf{U}[k], \mathbf{V}, \mathbf{O}[k-1], \mathbf{S}^t, s^t\right) + (\lambda_t/2)\,\|\mathbf{V}\|_F^2 \tag{4.11b}$$

$$\mathbf{O}[k] = \underset{\mathbf{O}\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \Phi\left(\mathbf{U}[k], \mathbf{V}[k], \mathbf{O}, \mathbf{S}^t, s^t\right) + \mu_t\|\mathbf{O}\|_1 \tag{4.11c}$$

over iterations indexed by k, until convergence is achieved.

The constrained subproblems in (4.11a) and (4.11b) are convex, and can be readily solved via projected gradient (PG) iterations. Since the gradients of their cost functions are available, and the projection operator onto the nonnegative orthant is well defined, PG iterations are guaranteed to eventually converge to the optimal solution [35, p. 223]. Per iteration k, PG updates amount to setting

$$\mathbf{U}[k] = \Big[\mathbf{U}[k-1] - \alpha_{u,k}\nabla_{\mathbf{U}}\Phi\Big(\mathbf{U}[k-1], \mathbf{V}[k-1], \mathbf{O}[k-1], \mathbf{S}^t, s^t\Big)\Big]_+ \tag{4.12}$$

$$\mathbf{V}[k] = \Big[\mathbf{V}[k-1] - \alpha_{v,k}\nabla_{\mathbf{V}}\Phi\Big(\mathbf{U}[k], \mathbf{V}[k-1], \mathbf{O}[k-1], \mathbf{S}^t, s^t\Big)\Big]_+ \tag{4.13}$$

where $\alpha_{u,k}$ and $\alpha_{v,k}$ denote (possibly) iteration-dependent step sizes. In addition, $\nabla_{\mathbf{U}}\Phi(.)$ ($\nabla_{\mathbf{V}}\Phi(.)$) denotes the gradient of $\Phi(.)$ with respect to $\mathbf{U}$ ($\mathbf{V}$). Expressions for the gradients can be readily obtained, but are omitted here for brevity.

The cost function in (4.11c) is convex with both smooth and non-smooth terms. Recent advances in proximal algorithms have led to efficient, provably-convergent iterative schemes for solving such optimization problems. We will resort to the *fast iterative shrinkage thresholding algorithm (FISTA)* whose accelerated convergence rate renders it attractive for sequential learning [34].

4.3.3 FISTA for outlier updates

Note that (4.11c) does not admit a closed-form solution, and the proposed strategy will entail a number of inner iterations (indexed by r), per AM iteration k. FISTA solves for $\mathbf{O}$ in (4.11c) through a two-step update involving gradient descent on $\Phi(.)$, evaluated at a linear combination of the two most recent iterates, followed by a closed-form soft-thresholding step per iteration r. Setting

Algorithm 8 Alternating minimization

Input: $\{\mathbf{A}^t\}_{t=1}^T$, β, C

Initialize $\mathbf{U}[0]$, $\mathbf{V}[0]$, $\mathbf{O}[0]$

Set $\mathbf{S}^0 = \mathbf{0}$

for $t = 1 \dots$ **do**

  Set $s^t = (1 - \beta^t)/(1 - \beta)$

  Update $\mathbf{S}^t = \mathbf{A}^t + \beta \mathbf{S}^{t-1}$

  Initialize $k = 0$

  repeat

    $k = k + 1$

    Set $\{\alpha_{u,k}, \alpha_{v,k}\}$

    Compute $\mathbf{U}[k]$ via (4.12)

    Compute $\mathbf{V}[k]$ via (4.13)

    $r = 0, \mathbf{W}_r[k] = \mathbf{O}_r[k] = \mathbf{O}[k-1], \theta_r[k] = 1$

    repeat

      Update $\mathbf{X}_r[k]$ via (4.15)

      $\mathbf{O}_r[k] = \left[\text{shrink}_{\mu_t/L_\Psi}\left(\mathbf{X}_r[k]\right)\right]_+$

      $\theta_{r+1}[k] = (1 + \sqrt{1 + 4\theta_r^2[k]})/2$

      Update $\mathbf{W}_{r+1}[k]$ via (4.16)

      $r = r + 1$

    until $\mathbf{O}_r[k]$ converges

    $\mathbf{O}[k] = \mathbf{O}_r[k]$

  until $\{\mathbf{U}[k], \mathbf{V}[k], \mathbf{O}[k]\}$ converge

  $\hat{\mathbf{U}}^t = \mathbf{U}[k], \hat{\mathbf{V}}^t = \mathbf{V}[k], \hat{\mathbf{O}}^t = \mathbf{O}[k]$

end for

$\theta_0[k] = 1$ and $\mathbf{W}_1[k] = \mathbf{O}_{k-1}$, it turns out that the updates can be written as [34]

$$\mathbf{O}_r[k] = \underset{\mathbf{O}\in\mathbb{R}_+^{N\times C}}{\arg\min} \quad (L_\Phi/2)\left\|\mathbf{O} - \mathbf{X}_r[k]\right\|_F^2 + \mu_t\|\mathbf{O}\|_1 \tag{4.14}$$

where

$$\mathbf{X}_r[k] = (\mathbf{W}_r[k] - (1/L_\Phi)\nabla_{\mathbf{O}}\Phi(\mathbf{W}_r[k], \mathbf{U}[k], \mathbf{V}[k], \mathbf{S}^t, s^t)) \tag{4.15}$$

with

$$\theta_{r+1}[k] \;=\; (1 \;+\; \sqrt{1 + 4\theta_r^2[k]})/2.$$

Furthermore,

$$\mathbf{W}_{r+1}[k] = \mathbf{O}_r[k] + \left(\frac{\theta_r[k] - 1}{\theta_{r+1}[k]}\right)(\mathbf{O}_r[k] - \mathbf{O}_{r-1}[k]) \tag{4.16}$$

where L_Φ denotes a Lipschitz constant of $\nabla_{\mathbf{O}}\Phi(.)$. Note that (4.14) is similar to the least-absolute shrinkage and selection operator (Lasso) with a closed-form solution, namely $[\mathbf{O}_r[k]]_{ij} = \left[\text{shrink}_{\mu_t/L_\Phi}\left([\mathbf{X}_r[k]]_{ij}\right)\right]_+$, where the thresholding operator is defined entry-wise as [64]

$$\text{shrink}_\mu(x) := (|x| - \mu)_+\text{sign}(x).$$

Computation of $\mathbf{O}[k]$ entails solving (4.14) over several iterations indexed by r until convergence. Algorithm 8 summarizes the details of the developed community tracking scheme, with β and C assumed to be given as algorithm inputs.

4.4 Delay-sensitive operation

Algorithm 8 relies upon convergence of the unknown variables per time interval. Unfortunately, this mode of operation is not suitable for delay-sensitive applications, where decisions must be made "on the fly." In fact, one may be willing to trade off solution accuracy for real-time operation in certain application domains. This section puts forth a couple of algorithmic enhancements that will facilitate real-time tracking, namely by premature termination of PG iterations, and leveraging the stochastic gradient descent framework.

4.4.1 Premature termination

For networks that generally evolve slowly over time, it is not necessary to run the tracking algorithm until convergence per time interval. Since a compromise can be struck between an accurate solution computed slowly and a less accurate solution computed very fast, one can judiciously truncate the number of inner iterations to $k_{\max}$. This premature termination is well motivated when the network topology is piecewise stationary with sufficiently long coherence time, with respect to the number of time intervals. It is then unnecessary to seek convergence per time interval since it can be argued that a "good" solution will be attained across time intervals before the topology changes. If the network topology varies in accordance with a stationary distribution, it can be demonstrated that convergence will eventually be attained, even when $k_{\max} = 1$ i.e., running a single iteration per time interval. Algorithm 9 summarizes the steps involved in this inexact tracking scheme under the special case with $k_{\max} = 1$.

Algorithm 9 Inexact alternating minimization

Input: $\{\mathbf{A}^t\}_{t=1}^T$, β, C
Initialize $\mathbf{U}^0$, $\mathbf{V}^0$, $\mathbf{O}^0$
Set $\mathbf{S}^0 = \mathbf{0}$, $\mathbf{W}^0 = \mathbf{O}^0, \theta_0 = 1$
for $t = 1 \dots T$ **do**
  Set $s^t = (1 - \beta^t)/(1 - \beta)$
  Update $\mathbf{S}^t = \mathbf{A}^t + \beta \mathbf{S}^{t-1}$
  Set $\{\alpha_{u,t}, \alpha_{v,t}\}$
  $\mathbf{U}^t = \left[\mathbf{U}^{t-1} - \alpha_{u,t} \nabla_{\mathbf{U}} \Phi(\mathbf{U}^{t-1}, \mathbf{V}^{t-1}, \mathbf{O}^{t-1}, \mathbf{S}^t, s^t)\right]_+$
  $\mathbf{V}^t = \left[\mathbf{V}^{t-1} - \alpha_{v,t} \nabla_{\mathbf{V}} \Phi(\mathbf{V}^{t-1}, \mathbf{U}^t, \mathbf{O}^{t-1}, \mathbf{S}^t, s^t)\right]_+$
  $\mathbf{X}^t = (\mathbf{W}^t - \frac{1}{L_\Phi} \nabla_{\mathbf{O}} \Phi(\mathbf{W}^t, \mathbf{U}^t, \mathbf{V}^t, \mathbf{S}^t, s^t))$
  $\mathbf{O}^t = \left[\text{shrink}_{\mu_t / L_\Phi}\left(\mathbf{X}^t\right)\right]_+$
  $\theta_t = (1 + \sqrt{1 + 4\theta_{t-1}^2})/2$
  $\mathbf{W}^t = \mathbf{O}^t + ((\theta_{t-1} - 1)/\theta_t)\,(\mathbf{O}^t - \mathbf{O}^{t-1})$
end for

4.4.2 Stochastic gradient descent

Instead of premature termination, real-time operation can be facilitated by resorting to stochastic gradient descent (SGD) iterations. Let

$$\zeta_\tau\left(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{A}^\tau\right) := \left\|\mathbf{A}^\tau - (\mathbf{U}+\mathbf{O})\mathbf{V}^\top\right\|_F^2 \tag{4.17}$$

and

$$\zeta_{\lambda_\tau,\mu_\tau}\left(\mathbf{U},\mathbf{V},\mathbf{O}\right) := \frac{\lambda_\tau}{2}\left\{\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2\right\} + \mu_\tau\|\mathbf{O}\|_1 \tag{4.18}$$

and consider the online learning setup, where the goal is to minimize the expected cost $E\{\zeta_\tau\left(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{A}^\tau\right) + \zeta_{\lambda_\tau,\mu_\tau}\left(\mathbf{U},\mathbf{V},\mathbf{O}\right)\}$ (with respect to the unknown probability distribution of the data). In this subsection, an alternative online learning strategy is pursued, in which the expected cost is replaced with the empirical cost function $(1/t)\sum_{\tau=1}^{t}\left[\zeta_\tau\left(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{A}^\tau\right) + \zeta_{\lambda_\tau,\mu_\tau}\left(\mathbf{U},\mathbf{V},\mathbf{O}\right)\right]$ as a surrogate. Generally, SGD is applicable to separable sum-minimization settings, in which the τ-th summand is a function of the τ-th datum. To incur the least computational and memory storage costs, the SGD approach advocated here discards all past data, and solves

$$\underset{\mathbf{U},\mathbf{V},\mathbf{O}\in\mathbb{R}_+^{N\times C}}{\arg\min}\quad \zeta_t\left(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{A}^t\right) + \zeta_{\lambda_t,\mu_t}\left(\mathbf{U},\mathbf{V},\mathbf{O}\right) \tag{4.19}$$

per interval t, which is tantamount to solving the EWLSE upon setting $\beta = 0$. This tracking scheme is reminiscent of the popular *least mean-squares (LMS)* algorithm, that has been well-studied within the context of adaptive estimation; see e.g., [121]. With $\mathbf{Y} := [\mathbf{U}\ \mathbf{V}\ \mathbf{O}]$, a common approach to solving (4.19) involves the projected iteration (see e.g., [46])

$$\mathbf{Y}^t = \left[\mathbf{Y}^{t-1} - \eta\partial\left(\zeta_t(\mathbf{Y}) + \zeta_{\lambda_t,\mu_t}(\mathbf{Y})\right)\Big|_{\mathbf{Y}=\mathbf{Y}^{t-1}}\right]_+ \tag{4.20}$$

where $\partial(.)$ denotes the subgradient operator, and $\eta \geq 0$ is a small constant step-size. A major limitation associated with (4.20) is that the resulting per-iteration solutions $\mathbf{O}^t$ are generally not sparse. Instead of subgradients, an AM procedure with FISTA updates adopted for $\mathbf{O}^t$ is followed in a manner similar to Algorithms 8 and 9. The main differences stem from eliminating the recursive updates, and adopting constant step sizes to facilitate tracking.

To this end, the following subproblems are solved during interval t,

$$\mathbf{U}^t = \underset{\mathbf{U}\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \zeta_t\left(\mathbf{U},\mathbf{V}^{t-1},\mathbf{O}^{t-1},\mathbf{A}^t\right)+\zeta_{\lambda_t,\mu_t}\left(\mathbf{U},\mathbf{V}^{t-1},\mathbf{O}^{t-1}\right) \tag{4.21}$$

$$\mathbf{V}^t = \underset{\mathbf{V}\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \zeta_t\left(\mathbf{U}^t,\mathbf{V},\mathbf{O}^{t-1},\mathbf{A}^t\right)+\zeta_{\lambda_t,\mu_t}\left(\mathbf{U}^t,\mathbf{V},\mathbf{O}^{t-1}\right) \tag{4.22}$$

$$\mathbf{O}^t = \underset{\mathbf{O}\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \zeta_t\left(\mathbf{U}^t,\mathbf{V}^t,\mathbf{O},\mathbf{A}^t\right)+\zeta_{\lambda_t,\mu_t}\left(\mathbf{U}^t,\mathbf{V}^t,\mathbf{O}\right). \tag{4.23}$$

In order to operate in real-time, $\mathbf{U}$ and $\mathbf{V}$ can be updated by a single gradient descent step per time slot. Similarly, minimization of (4.23) across time slots entails a single FISTA update that linearly combines $\mathbf{O}^{t-1}$ and $\mathbf{O}^{t-2}$. Exact algorithmic details of the SGD-based community tracking algorithm are tabulated in Algorithm 10, with

$$\Upsilon_t(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{A}^t) := \zeta_t\left(\mathbf{U},\mathbf{V},\mathbf{O},\mathbf{A}^t\right)+\zeta_{\lambda_t,\mu_t}\left(\mathbf{U},\mathbf{V},\mathbf{O}\right).$$

Algorithm 10 SGD tracking algorithm

Input: $\{\mathbf{A}^t\}_{t=1}^T$, β, C, α_u, α_v
Initialize $\mathbf{U}^0$, $\mathbf{V}^0$, $\mathbf{O}^0$
Set $\mathbf{W}^0=\mathbf{O}^0, \theta_0=1$
for $t=1\ldots T$ **do**
  $\mathbf{U}^t = \left[\mathbf{U}^{t-1}-\alpha_u\nabla_{\mathbf{U}}\Upsilon_t(\mathbf{U},\mathbf{V}^{t-1},\mathbf{O}^{t-1},\mathbf{A}^t)\right]_+$
  $\mathbf{V}^t = \left[\mathbf{V}^{t-1}-\alpha_v\nabla_{\mathbf{V}}\Upsilon_t(\mathbf{U}^t,\mathbf{V},\mathbf{O}^{t-1},\mathbf{A}^t)\right]_+$
  $\mathbf{X}^t = (\mathbf{W}^t-\frac{1}{L_\Phi}\nabla_{\mathbf{O}}\Phi(\mathbf{W}^t,\mathbf{U}^t,\mathbf{V}^t,\mathbf{A}^t,1))$
  $\mathbf{O}^t = \left[\text{shrink}_{\mu_t/L_\Phi}\left(\mathbf{X}^t\right)\right]_+$
  $\theta_t = (1+\sqrt{1+4\theta_{t-1}^2})/2$
  $\mathbf{W}^t = \mathbf{O}^t+((\theta_{t-1}-1)/\theta_t)\left(\mathbf{O}^t-\mathbf{O}^{t-1}\right)$
end for

4.5 Decentralized Implementation

The tracking algorithms developed so far have assumed that connectivity data (i.e., $\{\mathbf{A}^t\}$) are acquired and processed in a centralized fashion. This may turn out to be infeasible, since for example,

certain applications store large graphs over distributed file storage system hosted across a large network of computers. In fact, graphs capturing the web link structure, and online social networks are typically stored as "chunks" of files that are both distributed across computing nodes, and spatially over several geographical sites. In addition to the inherent computational bottlenecks, soaring data communication costs would render centralized approaches infeasible in such scenarios.

In lieu of these computational constraints, this section puts forth a decentralized algorithm that jointly tracks temporal communities and anomalous nodes. The alternating-direction method of multipliers (ADMM) has recently emerged as powerful tool for decentralized optimization problems [117], and it will be adopted here for the community tracking task. A connected network of computing agents is deployed, with links representing direct communication paths between nodes. The key idea is that each node iteratively solves the problem using only a subset of the input data, while exchanging intermediate solutions with single-hop neighbors until consensus is achieved.

4.5.1 Consensus constraints

Consider an undirected graph $\mathcal{G} = (\mathcal{M}, \mathcal{L})$ whose vertices $\mathcal{M} := \{1, \ldots, M\}$ are M spatially-distributed computing agents, and whose edges $\mathcal{L} := \{1, \ldots, L\}$ are representative of direct communication links between pairs of agents. It is assumed that the processing network abstracted by $\mathcal{G}$ is connected, so that (multi-hop) communication is possible between any pair of agents. Suppose the temporal adjacency matrix is partitioned as follows $\mathbf{A}^t := [(\mathbf{A}_1^t)^\top, \ldots, (\mathbf{A}_M^t)^\top]^\top$, and it is distributed across the processing network. During time interval t, agent m receives the submatrix $\mathbf{A}_m^t$. To minimize communication costs, each agent is only allowed to send and receive data from its single-hop neighborhood $\mathcal{N}_m$. Let $\mathbf{U} := [\mathbf{U}_1^\top, \ldots, \mathbf{U}_M^\top]^\top$ and $\mathbf{O} := [\mathbf{O}_1^\top, \ldots, \mathbf{O}_M^\top]^\top$, where $\mathbf{U}_m \in \mathbb{R}^{N_m \times C}$, $\mathbf{O}_m \in \mathbb{R}^{N_m \times C}$, and $\sum_{m=1}^M N_m = N$. In terms of the per-agent submatrices, (4.8) can now be written as follows

$$\begin{Bmatrix} \text{arg min} \\ \{\mathbf{U}_m, \mathbf{O}_m \in \mathbb{R}_+^{N_m \times C}\}_{m=1}^M \\ \mathbf{V} \in \mathbb{R}_+^{N \times C} \end{Bmatrix} \sum_{m=1}^M \Bigg[\sum_{\tau=1}^t \beta^{t-\tau} \Big\| \mathbf{A}_m^\tau - (\mathbf{U}_m + \mathbf{O}_m)\mathbf{V}^\top \Big\|_F^2 + \frac{\lambda_t}{2}\|\mathbf{U}_m\|_F^2 + \frac{\lambda_t}{2M}\|\mathbf{V}\|_F^2 + \mu_t \|\mathbf{O}_m\|_1 \Bigg] \quad (4.24)$$

for $t = 1, \ldots, T$. Clearly $\mathbf{U}$ and $\mathbf{O}$ decouple across computing agents, whereas $\mathbf{V}$ does not. A viable approach entails allowing each agent to solve for its corresponding unknowns $\mathbf{U}_m$ and $\mathbf{O}_m$ in parallel, followed by communication of the estimates to a central processing node that solves for $\mathbf{V}$. The downside to this approach is that it involves a significant communication and storage overhead as the central node must receive intermediate values of $\{\mathbf{U}_m, \mathbf{O}_m\}_{m=1}^M$, and then broadcast its computed value of $\mathbf{V}$ per iteration. In addition, this introduces the risk of a single point of failure at the central node.

To operate in a truly decentralized manner, each agent will solve for $\mathbf{V}$ independently under *consensus* constraints requiring equality of the solution to those computed by single-hop neighbors. Since $\mathcal{G}$ is connected, it can be readily shown that consensus on the solution for $\mathbf{V}$ will be reached upon convergence of the algorithm [117]. Incorporating consensus constraints, each time interval entails solving the following fully decoupled minimization problem

$$\begin{aligned} \underset{\{\mathbf{U}_m, \mathbf{O}_m, \mathbf{V}_m\}_{m=1}^M}{\arg\min} \quad & \sum_{m=1}^{M} \left[\Phi\left(\mathbf{U}_m, \mathbf{V}_m, \mathbf{O}_m, \mathbf{S}_m^t, s^t\right) + \frac{\lambda_t}{2}\|\mathbf{U}_m\|_F^2 + \frac{\lambda_t}{2M}\|\mathbf{V}_m\|_F^2 + \mu_t\|\mathbf{O}_m\|_1 \right] \\ \text{s. to} \quad & \{\mathbf{U}_m, \mathbf{O}_m\} \in \mathbb{R}_+^{N_m \times C}, \mathbf{V}_m \in \mathbb{R}_+^{N \times C} \\ & \mathbf{V}_m = \mathbf{V}_n, n \in \mathcal{N}_m. \end{aligned} \tag{4.25}$$

Letting

$$\Psi_{\lambda_t}(\mathbf{U}_m, \mathbf{O}_m, \mathbf{V}_m, \mathbf{S}_m^t, s^t) := \Phi\left(\mathbf{U}_m, \mathbf{V}_m, \mathbf{O}_m, \mathbf{S}_m^t, s^t\right) + \frac{\lambda_t}{2}\|\mathbf{U}_m\|_F^2 + \frac{\lambda_t}{2M}\|\mathbf{V}_m\|_F^2 \tag{4.26}$$

and introducing dummy variables $\{\mathbf{P}_m\}_{m=1}^M$, (4.25) can be written as follows

$$\begin{aligned} \underset{\{\mathbf{U}_m, \mathbf{O}_m, \mathbf{V}_m\}_{m=1}^M}{\arg\min} \quad & \sum_{m=1}^{M} \left[\Psi_{\lambda_t}(\mathbf{U}_m, \mathbf{O}_m, \mathbf{V}_m, \mathbf{S}_m^t, s^t) + \mu_t\|\mathbf{P}_m\|_1 \right] \\ \text{s. to} \quad & \{\mathbf{U}_m, \mathbf{O}_m\} \in \mathbb{R}_+^{N_m \times C}, \mathbf{V}_m \in \mathbb{R}_+^{N \times C} \\ & \mathbf{O}_m = \mathbf{P}_m, \mathbf{V}_m = \mathbf{V}_n, n \in \mathcal{N}_m. \end{aligned} \tag{4.27}$$

In order to solve (4.27), introduce the variables $\{\bar{\mathbf{X}}_{nm}, \tilde{\mathbf{X}}_{nm}\}$, and modify the constraints $\mathbf{V}_m = \mathbf{V}_n, n \in \mathcal{N}_m$ as

$$\mathbf{V}_m = \bar{\mathbf{X}}_{nm}, \mathbf{V}_n = \tilde{\mathbf{X}}_{nm}, \bar{\mathbf{X}}_{nm} = \tilde{\mathbf{X}}_{nm}, n \in \mathcal{N}_m.$$

Introducing the dual variables $\{\mathbf{\Gamma}_m\}_{m=1}^M$, and $\{\{\bar{\mathbf{\Pi}}_{nm}, \tilde{\mathbf{\Pi}}_{nm}\}_{n\in\mathcal{N}_m}\}_{m=1}^M$, and temporarily ignoring non-negativity constraints, the resulting augmented Lagrangian can be written as

$$\mathcal{L}_\rho(\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_3,\mathcal{D}) = \sum_{m=1}^M \left[\Psi_{\lambda_t}(\mathbf{U}_m,\mathbf{O}_m,\mathbf{V}_m,\mathbf{S}_m^t,s^t)+\mu_t\|\mathbf{P}_m\|_1\right]$$
$$+\sum_{m=1}^M\left[\text{Tr}\left(\mathbf{\Gamma}_m^\top(\mathbf{O}_m-\mathbf{P}_m)\right)+\sum_{n\in\mathcal{N}_m}\text{Tr}\left\{\bar{\mathbf{\Pi}}_{nm}^\top(\mathbf{V}_m-\bar{\mathbf{X}}_{nm})+\tilde{\mathbf{\Pi}}_{nm}^\top(\mathbf{V}_n-\tilde{\mathbf{X}}_{nm})\right\}\right]$$
$$+\frac{\rho}{2}\sum_{m=1}^M\left[\|\mathbf{O}_m-\mathbf{P}_m\|_F^2+\sum_{n\in\mathcal{N}_m}\left\{\|\mathbf{V}_m-\bar{\mathbf{X}}_{nm}\|_F^2+\|\mathbf{V}_n-\tilde{\mathbf{X}}_{nm}\|_F^2\right\}\right] \quad (4.28)$$

where $\rho>0$, $\mathcal{P}_1:=\{\mathbf{V}_m\}_{m=1}^M$, $\mathcal{P}_2:=\{\mathbf{U}_m,\mathbf{O}_m,\mathbf{P}_m\}_{m=1}^M$, and $\mathcal{P}_3:=\{\{\bar{\mathbf{X}}_{nm},\tilde{\mathbf{X}}_{nm}\}_{n\in\mathcal{N}_m}\}_{m=1}^M$ denote primal variables, while $\mathcal{D}:=\{\mathbf{\Gamma}_m,\{\bar{\mathbf{\Pi}}_{nm},\tilde{\mathbf{\Pi}}_{nm}\}_{n\in\mathcal{N}_m}\}_{m=1}^M$ denotes the set of dual variables.

Towards applying ADMM to (4.28), an iterative strategy is pursued, entailing dual variable update as the first step, followed by alternate minimization of $\mathcal{L}_\rho(\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_3,\mathcal{D})$ over each of the primal variables, while holding the rest fixed to their most recent values. Since $\mathcal{L}_\rho(\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_3,\mathcal{D})$ is completely decoupled across the M computing agents, the problem can be solved in an entirely decentralized manner. The per-agent updates of the proposed algorithm during iteration k comprise the following steps.

[S1]: Dual variable update.

$$\mathbf{\Gamma}_m[k+1] = \mathbf{\Gamma}_m[k]+\rho(\mathbf{O}_m[k]-\mathbf{P}_m[k]) \quad (4.29)$$
$$\bar{\mathbf{\Pi}}_{nm}[k+1] = \bar{\mathbf{\Pi}}_{nm}[k]+\rho(\mathbf{V}_m[k]-\bar{\mathbf{X}}_{nm}[k]) \quad (4.30)$$
$$\tilde{\mathbf{\Pi}}_{nm}[k+1] = \tilde{\mathbf{\Pi}}_{nm}[k]+\rho(\mathbf{V}_m[k]-\tilde{\mathbf{X}}_{nm}[k]) \quad (4.31)$$

[S2]: Primal variable update.

$$\mathcal{P}_1[k+1] = \arg\min_{\mathcal{P}_1}\ \mathcal{L}_\rho(\mathcal{P}_1,\mathcal{P}_2[k],\mathcal{P}_3[k],\mathcal{D}[k+1]) \quad (4.32)$$
$$\mathcal{P}_2[k+1] = \arg\min_{\mathcal{P}_2}\ \mathcal{L}_\rho(\mathcal{P}_1[k+1],\mathcal{P}_2,\mathcal{P}_3[k],\mathcal{D}[k+1]) \quad (4.33)$$
$$\mathcal{P}_3[k+1] = \arg\min_{\mathcal{P}_3}\ \mathcal{L}_\rho(\mathcal{P}_1[k+1],\mathcal{P}_2[k+1],\mathcal{P}_3,\mathcal{D}[k+1]) \quad (4.34)$$

It can be shown that the splitting variables in $\mathcal{P}_3$ turn out to be redundant in the final algorithm. Letting $\bar{\mathbf{\Pi}}_m[k] := \sum_{n\in\mathcal{N}_m} \bar{\mathbf{\Pi}}_{nm}[k]$, it turns out that

$$\bar{\mathbf{\Pi}}_m[k+1] = \bar{\mathbf{\Pi}}_m[k] + (\rho/2)\left(|\mathcal{N}_m|\mathbf{V}_m[k] - \sum_{n\in\mathcal{N}_m}\mathbf{V}_n[k]\right) \tag{4.35}$$

and the dual variables $\tilde{\mathbf{\Pi}}_{nm}$ can be discarded. In addition, the per-iteration primal variable updates per agent simplify to (see Appendix C for derivations):

$$\begin{aligned}
\mathbf{V}_m[k+1] &= \underset{\mathbf{V}_m\in\mathbb{R}_+^{N\times C}}{\arg\min}\ \Psi_{\lambda_t}(\mathbf{U}_m[k],\mathbf{O}_m[k],\mathbf{V}_m,\mathbf{S}_m^t,s^t) && (4.36)\\
&\quad +\mathrm{Tr}\Big[(\rho/2)|\mathcal{N}_m|\mathbf{V}_m^\top\mathbf{V}_m + \mathbf{V}_m^\top\Big(\bar{\mathbf{\Pi}}_m[k+1] - (\rho/2)\{|\mathcal{N}_m|\mathbf{V}_m[k] && (4.37)\\
&\quad + \sum_{n\in\mathcal{N}_m}\mathbf{V}_n[k]\}\Big)\Big] && (4.38)\\
\mathbf{U}_m[k+1] &= \underset{\mathbf{U}_m\in\mathbb{R}_+^{N_m\times C}}{\arg\min}\ \Psi_{\lambda_t}(\mathbf{U}_m,\mathbf{O}_m[k],\mathbf{V}_m[k+1],\mathbf{S}_m^t,s^t) && (4.39)\\
\mathbf{O}_m[k+1] &= \underset{\mathbf{O}_m\in\mathbb{R}_+^{N_m\times C}}{\arg\min}\ \Psi_{\lambda_t}(\mathbf{U}_m[k+1],\mathbf{O}_m,\mathbf{V}_m[k+1],\mathbf{S}_m^t,s^t) && (4.40)\\
&\quad +\mathrm{Tr}\left(\mathbf{\Gamma}_m^\top[k+1]\mathbf{O}_m\right) + (\rho/2)\|\mathbf{O}_m - \mathbf{P}_m[k]\|_F^2 && (4.41)\\
\mathbf{P}_m[k+1] &= \underset{\mathbf{P}_m\in\mathbb{R}_+^{N_m\times C}}{\arg\min}\ (\rho/2)\|\mathbf{O}_m[k+1] - \mathbf{P}_m\|_F^2\\
&\quad -\mathrm{Tr}\left(\mathbf{\Gamma}_m^\top[k+1]\mathbf{P}_m\right) + \mu_t\|\mathbf{P}_m\|_1. && (4.42)
\end{aligned}$$

Algorithm 11 summarizes the steps involved in the per-agent decentralized ADMM algorithm.

4.6 Numerical Tests

4.6.1 Synthetic data

Data generation. An initial synthetic network with $N = 5,000$ nodes, and $C = 4$ communities of unequal sizes was generated using the stochastic blockmodel (SBM) [66]. Let δ_{ij} denote the probability that an edge exists between nodes i and j. In this experiment, $\delta_{ij} = 0.85$ whenever i and j belong to the same community, otherwise $\delta_{ij} = 0.15$. The second and third communities were allowed to overlap with each other. The initial SBM network was captured through an adjacency matrix $\mathbf{A}^{\text{init}} \in \{0,1\}^{N\times N}$. Matrix $\mathbf{A}^{\text{init}}$ was then decomposed into non-negative factors $\mathbf{U}^0$ and

Algorithm 11 Decentralized tracking algorithm per agent m

Input: $\{\mathbf{A}_m^t\}_{t=1}^T$, β, C, ρ

$\mathbf{U}_m^0$, $\mathbf{V}_m^0$, $\mathbf{O}_m^0 = \mathbf{P}_m^0 = \mathbf{0}$

$\bar{\mathbf{\Pi}}_m^0[0] = \mathbf{0}$, $\bar{\mathbf{\Gamma}}_m^0[0] = \mathbf{0}$

for $t = 1 \dots T$ **do**

  Set $s^t = (1 - \beta^t)/(1 - \beta)$

  Update $\mathbf{S}_m^t = \mathbf{A}_m^t + \beta \mathbf{S}_m^{t-1}$

  Update μ_t and λ_t

  $\mathbf{V}_m[0] = \mathbf{V}_m^{t-1}$, $\mathbf{U}_m[0] = \mathbf{U}_m^{t-1}$

  $\mathbf{O}_m[0] = \mathbf{P}_m[0] = \mathbf{O}_m^{t-1}$, $k = 0$

  repeat

    Receive $\{\mathbf{V}_n[k]\}_{n \in \mathcal{N}_m}$ from neighbors of m

    $\mathbf{\Gamma}_m[k+1] = \mathbf{\Gamma}_m[k] + \rho(\mathbf{O}_m[k] - \mathbf{P}_m[k])$

    Compute $\bar{\mathbf{\Pi}}_m[k+1]$ according to (4.35)

    Update $\mathbf{V}_m[k+1]$ via (4.36)

    Update $\mathbf{U}_m[k+1]$ via (4.39)

    Update $\mathbf{O}_m[k+1]$ via (4.40)

    Update $\mathbf{P}_m[k+1]$ via (4.42)

    Broadcast $\mathbf{V}_m[k+1]$ to single-hop neighbors

    $k = k + 1$

  until $\mathbf{U}_m[k], \mathbf{V}_m[k], \mathbf{O}_m[k]$ converge

  $\mathbf{U}_m^t = \mathbf{U}_m[k], \mathbf{V}_m^t = \mathbf{V}_m[k], \mathbf{O}_m^t = \mathbf{O}_m[k]$

end for

$\mathbf{V}^0$, that is, $\mathbf{A}^{\text{init}} = \mathbf{U}^0(\mathbf{V}^0)^\top$, using standard off-the-shelf NMF tools [135]. Anomalies were artificially induced by reconstructing a modified adjacency matrix $\mathbf{A}^0 = (\mathbf{U}^0 + \mathbf{O}^0)(\mathbf{V}^0)^\top$, where 98% of rows constituting $\mathbf{O}^0$ are all-zero, but the remaining 2% comprise non-zero entries, and have uniformly distributed indices between 1 and $5,000$. Figure 4.2 depicts $\mathbf{A}^0$, with anomalous nodes seen as unusually highly connected.

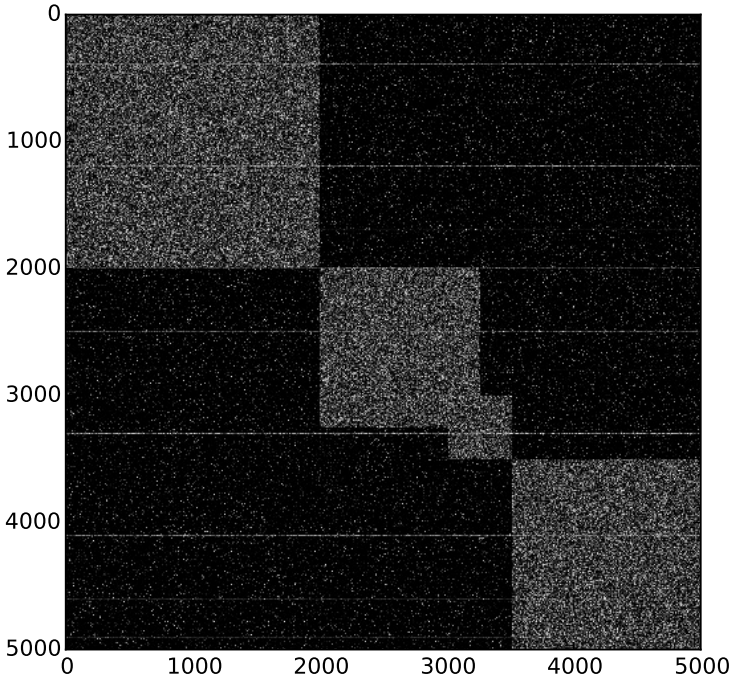


Figure 4.2: Initial synthetic adjacency matrix $\mathbf{A}^0$ with four overlapping communities.

In order to generate slowly evolving networks, four piecewise constant edge-variation functions were adopted: i) $f_1(t) = H(t)$; ii) $f_2(t) = H(t-50)$; iii) $f_3(t) = 1 - H(t-50)$; and iv) $f_4(t) = H(t) - H(t-25) + H(t-50) - H(t-75)$, where $H(t)$ denotes the unit step function, and $t = 1, \ldots, 100$. The time-series $\{\mathbf{A}^t\}_{t=1}^{100}$ was generated by setting the edge indicators to $a_{ij}^t = a_{ij}^0 f_\kappa(t)$, with κ uniformly selected at random from $\{1, 2, 3, 4\}$.

Numerical results. Algorithm 8 was initialized by setting $\mathbf{O}$ to an all-zero matrix, while $\mathbf{U}$ and $\mathbf{V}$ were initialized to NMF factors of $\mathbf{A}^0$ computed in batch. With $K = 4$ and $\beta = 0.97$, Algorithm 8

was run to track the constituent communities and anomalies. Selection of λ_t and μ_t is admittedly challenging under dynamic settings where data are sequentially acquired. Nevertheless, heuristics such as increasing μ_t over time work reasonably well when the unknowns vary slowly. It turns out that setting $\lambda_t = 0.09$, and $\mu_t = 0.5\sqrt{t}$ yielded remarkably good community tracking abilities. Generally, $5 - 10$ inner iterations sufficed for convergence per time step. In addition, successive initializations were done by warm restarts using solutions from the most recent time interval. Figure 4.3 depicts gray-scale plots of $\hat{\mathbf{U}}^t$ for $t = 30$ and $t = 80$, with entries scaled over the range $[0, 1]$ indicative of relative community affiliation strengths (white corresponds to strongest affiliation). As expected from the synthetic data, strong communities are recovered with a significant overlap between two of the node groups.

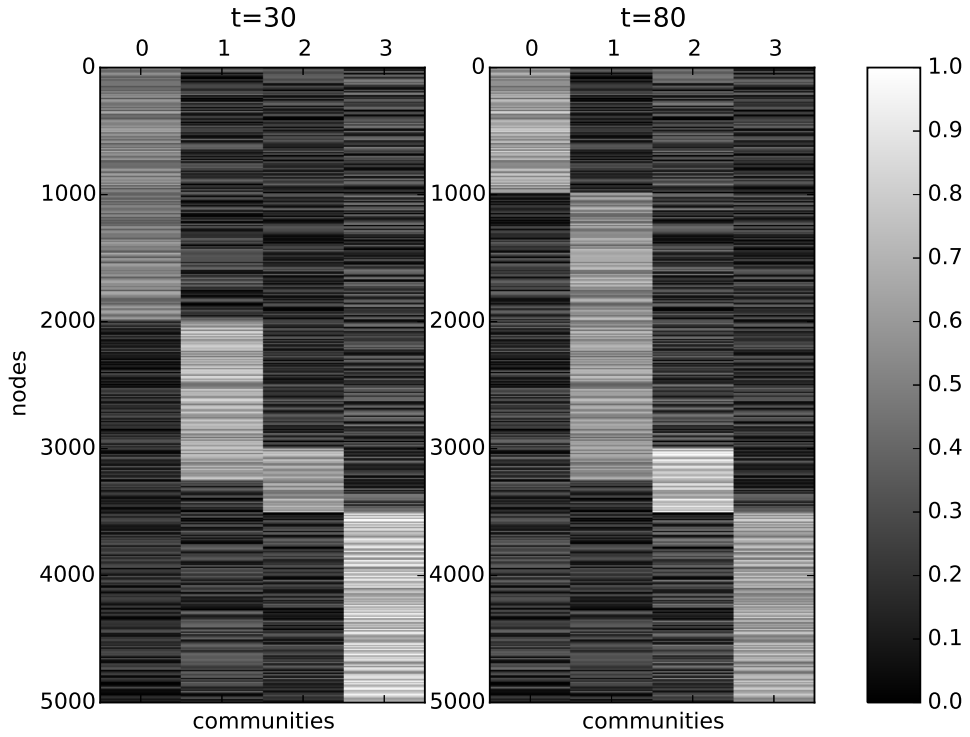


Figure 4.3: Plots of $\hat{\mathbf{U}}^t$ for $t = 30$ and $t = 80$, with entries normalized to $[0, 1]$.

In order to draw comparisons with the state-of-the-art methods in overlapping community discovery, experiments were run using two recent approaches. The first algorithm is known as the

democratic estimate of the modular organization of a network (DEMON), and was proposed in [49]. The key idea in [49] involves extraction of all egonets (subgraphs induced by neighborhoods of each node), and identifying local communities per egonet by a label propagation mechanism. The network-wide community assignments are then obtained by merging the per-egonet local communities. The second approach assumes that edges are generated by a stochastic blockmodel, with parameters that are indicative of overlapping community affiliations, and have known prior distributions [130]. A Bayesian approach is advocated for estimating the unknown model parameters, which are then used to cluster nodes, and to predict emergence of edges. The developed algorithm is known as relevance-based overlapping clustering with similarity-based smoothing (ROCS), and it assigns communities to all nodes, except those identified as outliers that do not belong to a specific community.

Figure 4.4 compares community detection error rates resulting from running Algorithm 8 against the ROCS, and DEMON algorithms per time interval. The community detection error is computed as a fraction of the total number of nodes that are assigned to wrong communities per t. It turns out that Algorithm 8 consistently outperforms ROCS and DEMON for $t > 10$. Algorithm 8 exploits the slow variations of the network topology by recursively accumulating past observations, effectively ensuring that per-interval community assignments benefit from more data. Consequently, as more data are acquired, the algorithm markedly outperforms both ROCS and DEMON. The outlier identification performance was also compared for Algorithm 8 and ROCS. Figure 4.5 plots the corresponding outlier detection errors, computed as the fraction of nodes that are misclassified as outliers per time interval. Although the initial error rate performance is poor, it is clear from Figure 4.5 that Algorithm 8 markedly outperforms ROCS with respect to outlier identification as more data are sequentially acquired.

Further tests were conducted to compare the relative error community detection performance of Algorithms 9, 10, and 11. For the decentralized implementation, a simple connected network of 10 computing agents was adopted, as shown in Figure 4.6. Figure 4.7 plots the errors with respect to the ground truth, resulting from running the algorithms using the synthetic network time-series. With initial batch solutions, community detection errors are initially small, followed by dramatic increases upon acquisition of sequential data. As more data are acquired, tracking with

premature termination leads to slightly lower errors on average than the SGD alternative, whose error performance is less stable. Decentralized iterations yield the slowest error decay, because in addition to seeking convergence to the batch solution, consensus per agent must be attained per time-interval. Furthermore, Table 4.2 lists average per-interval running times of Algorithms 9, 10, and 11. As expected, the need for consensus leads to slower average running times for Algorithm 11.

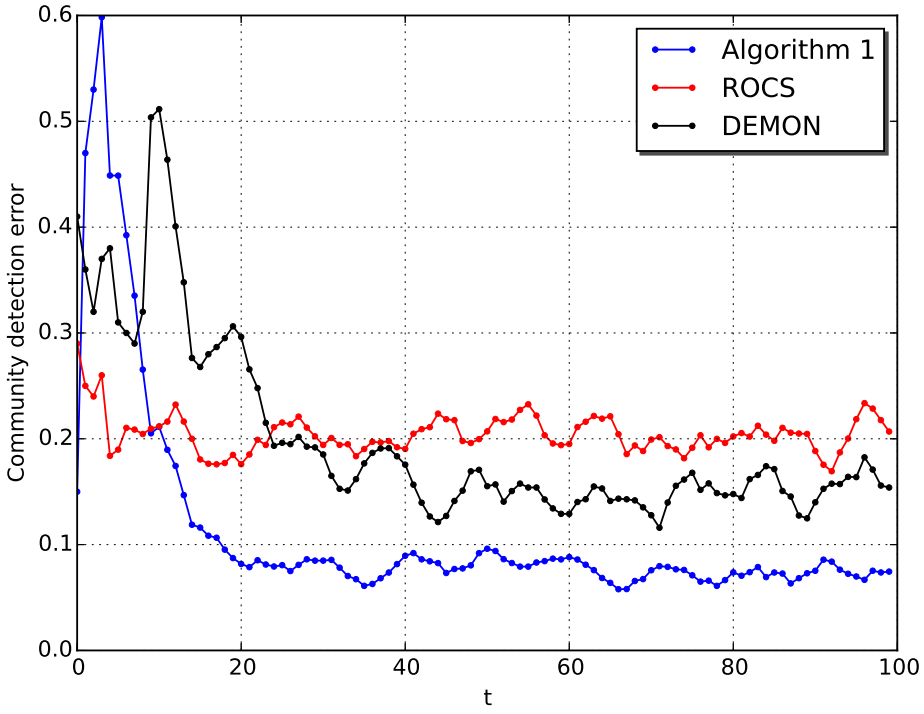


Figure 4.4: Comparison of community detection errors between Algorithm 8, ROCS, and DEMON algorithms.

4.6.2 Real data

Dataset description. The developed algorithms were tested on a time-series of real-world networks extracted from global trade flow statistics. Extracted under the *Correlates of War* project [26], the dataset captures information on annual bilateral trade flows (imports and exports) among countries between 1870 and 2009. The network time-series were indexed by trade years, and each node was

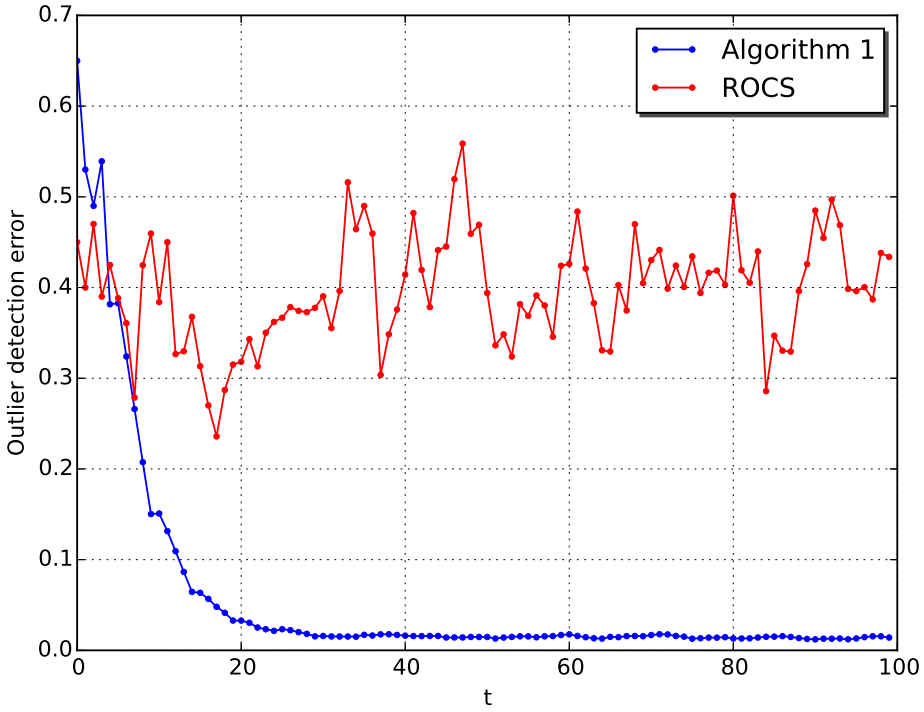


Figure 4.5: Comparison of outlier detection errors between Algorithm 8 and ROCS.

representative of a country, while directed and weighted edges were indicative of the volume and direction (export/import) of trade between countries measured in present-day U.S. dollars.

Since trade volumes between countries can vary by orders of magnitude, edge weights were set to logarithms of the recorded trade flows. It is also important to note that some countries did not exist until 50 or fewer years ago. As a result, network dynamics in the dataset were due to arrival and obsolescence of some nodes, in addition to annual changes in edges and their weights. Since this chapter assumes that a fixed set of nodes is available, the tracking algorithm was run for data ranging from 1949 to 2009 (i.e., $T = 60$), with $N = 170$ countries.

The objective of this experiment was to track the evolution of communities within the global trade network, and to identify any anomalies. Note that communities in the world trade network may be interpreted as regional trading consortia. Algorithm 10 was run with $C = 7$ communities, $\beta = 0.97$, $\alpha_u = \alpha_v = 0.002$, $\lambda_t = 0.5$, and $\mu_t = 1.0$, for all $t = 1, \ldots, T$. Initial values $\mathbf{U}^0$ and

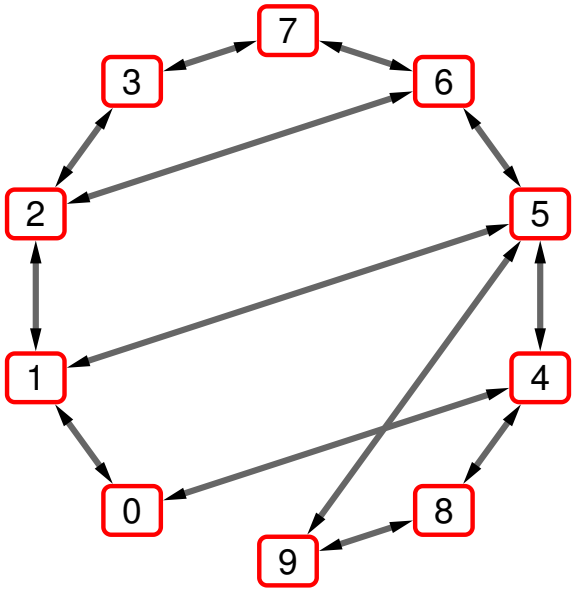


Figure 4.6: Simple 10-node network used for the decentralized community tracking task.

$\mathbf{V}^0$ were obtained by traditional NMF on $\mathbf{A}^0$, and $\mathbf{O}^0$ was set to an all-zero $N \times C$ matrix.

Numerical results. Running Algorithm 10 on this dataset revealed interesting insights about the evolution of global trade within the last sixty years. Figure 4.8 depicts stacked plots of countries and the communities they belonged to over a subset of years within the observation period. The horizontal axes are indexed by countries, and each community is depicted by a specific color. It is clear that over the years, more countries cultivated stronger affiliations within different communities. This observation suggests an increasing trend of globalization, with more countries actively engaging in significant trade relationships within different trade communities. Between 1949 and 1963, global trade was dominated by one major community, while the other communities played a less significant role. Based on historical accounts, it is likely that such trade dynamics were related to ongoing global recovery in the years following the second world war.

Figure 4.9 depicts visualizations of the global trade network for the years 1959 and 1990, with countries color-coded according to the community with which they are most strongly affiliated, and their names truncated to the first five letters for visual clarity. A core community of economic powerhouses (in green) comprises global leaders such as the United States, United Kingdom, Canada, France etc., as seen from the 1959 visualization. Interestingly, this core group of countries remains

	Country	**Years as anomaly**
1	German Federal Republic	$1954, 1955, 1956, 1966$
2	Russia	$1953, 1954, 1955, 1956$
3	Austria	$1955, 1957, 1960$
4	Pakistan	$1955, 1956, 1957$
5	Japan	$1954, 1955, 1965, 1967$
6	South Korea	$1955, 1965$
7	Yugoslavia	$1954, 1956$
8	Iraq	1967
9	Yemen Arab Republic	$1964, 1965$
10	Kenya	$1966, 1968$

Table 4.1: Anomalous countries in the global trade dataset.

	Avg. run time/interval (Synthetic data)	**Avg. run time/interval (Real data)**
Alg. 9	112.380 s	9.103 s
Alg. 10	49.662 s	6.012 s
Alg. 11	198.371 s	15.265 s

Table 4.2: Average per-interval running times for Algorithms 9, 10, and 11 on both datasets.

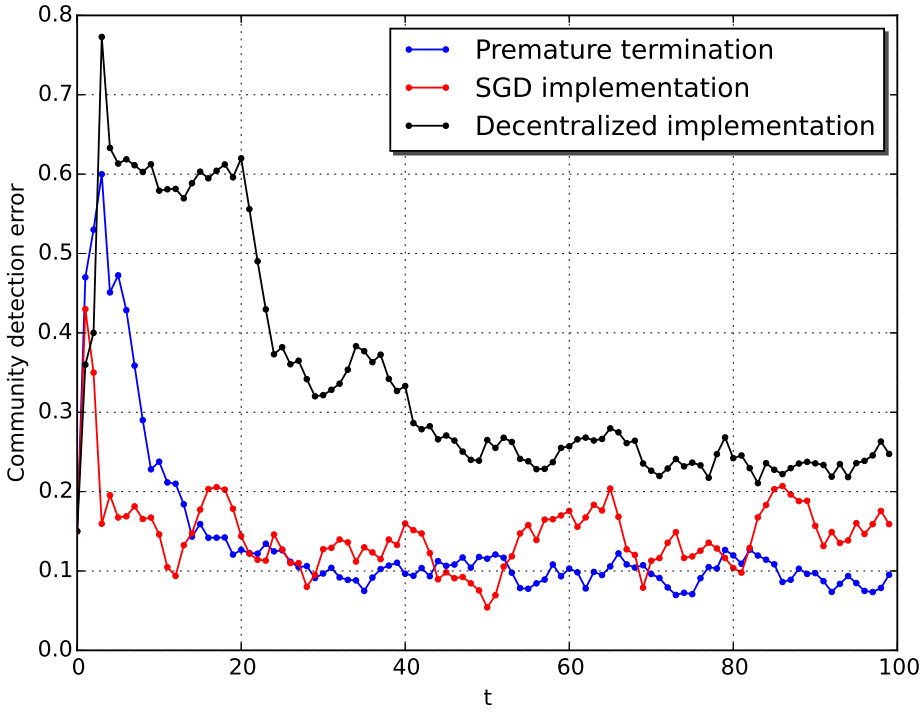


Figure 4.7: Error performance comparisons of improved implementations of Algorithm 8 with $\lambda_t = 0.09, \mu_t = 0.5\sqrt{t}$, and $\beta = 0.97$.

intact as a community in 1990. It turns out that geographical proximity and language play an important role in trade relationships. This is evident from two communities (colored maroon and yellow) which are dominated by South and Central American nations, with Spain and Portugal as exceptions that have strong cultural and language influences on these regions.

Another interesting observation from the 1990 visualization is the large community of developing nations (in blue). Most of these nations only existed as colonies in 1959, and are not depicted in the first drawing. However, 1990 lies within their post-colonial period, during which they presumably started establishing strong trading ties with one another.

Finally, Table 4.1 lists countries that were flagged off by Algorithm 10 as anomalies. The table shows each of these countries, along with a list of years during which they were identified as anomalies. In the context of global trade, anomalies are expected to indicate abnormal or irregular

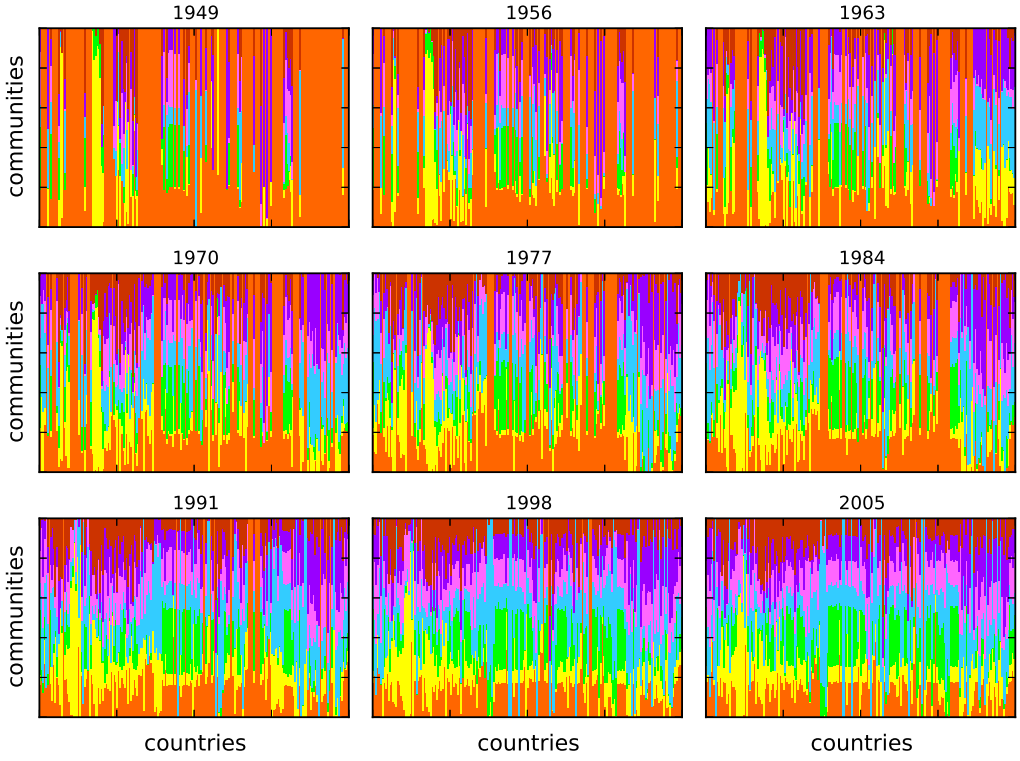
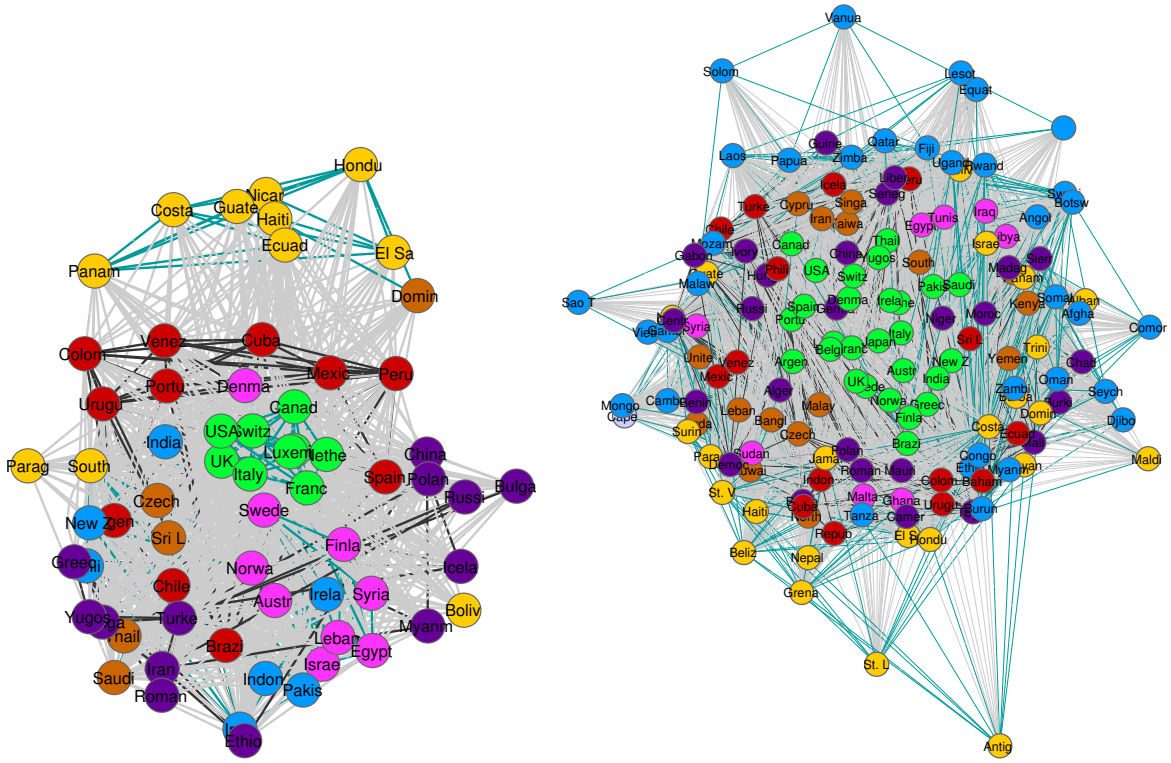


Figure 4.8: Overlapping communities in world trade flows dataset.

trading patterns. Interestingly, the German Federal Republic, Russia, and Japan were some of the most adversely affected countries by the second world war, and their trade patterns during a period of rapid economic revival may corroborate identification as anomalies. It is also possible that South Korea was flagged off in 1955 and 1965 because of its miraculous economic growth that started in the 1950s.

4.7 Chapter Summary

This chapter put forth a novel approach for jointly tracking communities in time-varying network settings, and identifying anomalous members. Leveraging advances in overlapping community discovery, a temporal outlier-aware edge generation model was proposed. It was assumed that the anomaly matrix is sparse, the outlier-free, noiseless factor model is low rank, and the network evolves slowly. Based on these assumptions, a sparsity-promoting, rank-regularized EWLSE was



(a) 1959 (b) 1990

Figure 4.9: Dominant communities identified in the largest connected component of the global trade network for the years 1959 and 1990.

advocated to jointly track communities and identify anomalous nodes. A first-order sequential tracking algorithm was developed, based on alternating minimization and recent advances in accelerated proximal-splitting optimization.

Motivated by contemporary needs for processing big data, often in streaming and distributed settings, a number of algorithmic improvements were put forth. Real-time operation was attained by developing a fast online tracking algorithm, based on stochastic gradient iterations. For settings involving distributed acquisition and storage of network data, a decentralized tracking algorithm that capitalizes on the separability inherent to ADMM iterations was developed.

Simulations on synthetic SBM networks successfully unveiled the underlying communities, and flagged off artificially-induced anomalies. Experiments conducted on a sequence of networks extracted from historical global trade flows between nations revealed interesting results concerning

globalization of trade, and unusual trading behavior exhibited by certain countries during the early post-world war era.

Chapter 5

Future Directions

This dissertation dealt with novel statistical models and efficient algorithms for dynamic inference over time-varying networks. Although motivating examples and experiments were predominantly drawn from social networks and the world-wide web, the advocated approaches easily carry over to a broader class of networks, e.g., biological networks. The present chapter concludes this dissertation by pointing out possible directions for future research, as well as opportunities for more extensive experimental validation on more comprehensive datasets.

5.1 Nonlinear time-varying structural equation models

Chapter 2 postulated a linear dynamic structural equation model (SEM) for tracking hidden topologies of networks that facilitate cascade propagation. Admittedly, assuming linearity often leads to tractable models that are amenable to efficient tracking algorithms. In reality, the dependence of cascades on the underlying network topology is more likely nonlinear, and assuming linearity may lead to inherently biased estimators. Several prior works have previously advocated nonlinear static SEMs, by resorting to polynomial expansions for the endogenous term, the exogenous term, or both [63, 75, 127]. However, these models generally do not scale to large networks because they assume complete knowledge of the topology, but with unknown edge weights.

In order to capture nonlinear dependencies, a potential direction is to cast (2.1) as the following

more general nonlinear SEM

$$y_{ic}^t = \phi(\mathbf{y}_c^t; \mathbf{a}_i^t) + \psi(\mathbf{x}_i; b_{ii}^t) + e_{ic}^t \tag{5.1}$$

where $\phi(.)$ and $\psi(.)$ are nonlinear functions, parameterized by the unknowns $\{\mathbf{a}_i^t, b_{ii}^t\}_{i=1}^N$ encoding the underlying causal structure in the network. Although selection of the "best" $\phi(.)$ and $\psi(.)$ is quite challenging, it is possible to leverage the rich framework of kernel methods to derive estimators that do not require exact prior knowledge of their functional forms [64, Chap. 5]. Furthermore, recent advances in multi-kernel learning approaches can be exploited for data-driven selection of the most representative kernels; see e.g., [33] and references therein.

5.2 State-space models for cascade evolution

This research direction builds upon the switched dynamic SEMs explored in Chapter 3. The developed algorithms were general, and assumed no prior knowledge about the evolution of network states. Nevertheless, if state evolution dynamics are known beforehand, one can in principle derive more efficient tracking algorithms for the hidden network states. For example, if the sequence of network states $\{\sigma(t)\}$ constitutes a first-order Markov chain, then the transition probabilities

$$\text{Pr}(\sigma(t)|\sigma(t-1), \ldots, \sigma(1)) = \text{Pr}(\sigma(t)|\sigma(t-1)) \tag{5.2}$$

suffice to describe the state evolution dynamics. If it is also assumed that the a posteriori probability $\text{Pr}(\sigma(t)|\mathbf{A}^t)$ belongs to a known family of distributions, then one can adopt *hidden Markov models* to learn the unknown state transition probabilities, and hence track the evolution of the switching network using Bayesian methods [36, Chap. 13].

Going beyond the assumption that network topologies exist in discrete and finite states, it may also be possible to endow the dynamic SEM introduced in Chapter 2 with a general nonlinear state space model. For instance, one may adopt the following model

$$\begin{aligned} \mathbf{A}^t &= \zeta(\mathbf{A}^{t-1}) + \mathbf{F}^t \\ \mathbf{Y}^t &= \mathbf{A}^t\mathbf{Y}^t + \mathbf{B}^t\mathbf{X} + \mathbf{E}^t \end{aligned} \tag{5.3}$$

where $\zeta(.)$ is a known nonlinear function, and $\mathbf{F}^t$ captures unmodeled state dynamics. The extended Kalman filter has well-documented merits for tracking time-varying parameters that adhere

to nonlinear state evolution models [74, Chap. 13], and will be adopted to estimate the unknowns in (5.3). Alternative approaches trading off performance for complexity include unscented Kalman and particle filters.

5.3 Applications in dynamic brain connectivity studies

An emerging theme in brain studies is the idea that complex cognitive behavior is mediated by network-based interactions between neurons at different resolution levels; see e.g, [41] and references therein. Contemporary works leverage imaging modalities such as functional magnetic resonance imaging (fMRI) to observe changes in blood oxygenation levels within the brain, in response to external stimuli. These fMRI signals constitute time series aggregated over prescribed brain regions of interest (ROI), and are typically used to study causal relationships between ROIs, e.g., via SEMs. Although most conventional analyses assume static neural network topologies, it is widely accepted that causal interactions in the brain undergo temporal changes over varying time scales. Consequently, dynamic models for tracking time-varying brain network topologies will play an important role in modern clinical and cognitive neurosciences.

The statistical models advocated by this dissertation to infer dynamic networks can be applied within the context of brain network studies. Naturally, the simplifying assumptions made in the context of social networks may not directly apply to brain networks. For example, it is well-known that fMRI data has a high spatial resolution at the expense of a low temporal resolution, which raises issues pertaining to the appropriate length of time intervals. One of the merits of the advocated dynamic SEMs is explicit incorporation of exogenous information, unlike competing approaches. Although external sensory stimuli can be viewed as exogenous inputs, current imaging technologies may not be well-equipped to explicitly measure such stimuli in a meaningful way. Our preliminary approaches in this direction can be found in [118].

Finally, the community and anomaly tracking approaches presented in Chapter 4 can be applied to the inferred brain networks. With respect to neural behavior, communities may be indicative of clusters of neurons or ROIs that collaborate to accomplish a certain task. On the other hand, identification of emerging anomalies may be indicative of brain disorders, which would potentially lead to early diagnoses in clinical neuroscience. Once again, domain knowledge will be crucial to

adjust the simplifying assumptions made with respect to social networks.

Bibliography

[1] L. A. Adamic, "The small world web," in *Research and Advanced Technology for Digital Libraries*. Springer, 1999, pp. 443–452.

[2] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web," *Science*, vol. 287, no. 5461, pp. 2115–2115, 2000.

[3] L. A. Adamic, R. J. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *Physical Review E*, vol. 64, no. 4, pp. (046 135)1–8, 2001.

[4] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *arXiv preprint arXiv:1211.3412*, 2012.

[5] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

[6] E. Almaas, R. V. Kulkarni, and D. Stroud, "Characterizing the structure of small-world networks," *Physical Review Letters*, vol. 88, no. 9, pp. (098 101)1–4, 2002.

[7] L. A. N. Amaral, A. Scala, M. Barthelemy, and H. E. Stanley, "Classes of small-world networks," *Proceedings of the National Academy of Sciences*, vol. 97, no. 21, pp. 11 149–11 152, 2000.

[8] D. Angelosante, J. A. Bazerque, and G. B. Giannakis, "Online adaptive estimation of sparse signals: where RLS meets the ℓ_1-norm," *IEEE Trans. Signal Process.*, vol. 58, pp. 3436–3447, Jul. 2010.

[9] D. Angelosante and G. B. Giannakis, "Sparse graphical modeling of piecewise-stationary time series," in *Proc. of Intern. Conf. on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011.

[10] B. Baingana, E. Dall'Anese, G. Mateos, and G. B. Giannakis, "Robust kriged Kalman filtering," in *Proc. of Asilomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2015.

[11] B. Baingana and G. B. Giannakis, "Centrality-constrained graph embedding," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[12] ——, "Tracking anomalous community memberships in time-varying networks," in *Proc. of Global Conf. on Signal and Info. Processing*, Atlanta, GA, Dec. 2014.

[13] ——, "Dynamic and decentralized learning of overlapping communities," in *Proc. of Intl. Conf. on Computational Advances in Multi-Sensor Adpative Processing*, Cancun, Mexico, Dec. 2015.

[14] ——, "Kernel-based embeddings for large graphs with centrality constraints," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing*, Brisbane, Australia, April 2015.

[15] ——, "Switched dynamic structural equation models for network topology tracking," in *Proc. of Global Conf. on Signal and Info. Processing*, Orlando, FL, Dec. 2015.

[16] ——, "Tracking switched dynamic network topologies from information cascades," *IEEE Transactions in Signal Processing (submitted)*, Dec. 2015.

[17] ——, "Joint community and anomaly tracking in dynamic networks," *IEEE Transactions in Signal Processing (to appear)*, Feb. 2016.

[18] B. Baingana, G. Mateos, and G. B. Giannakis, "Dynamic structural equation models for tracking cascades over social networks," in *Proc. of Neural Information Processing Systems Workshop on Frontiers of Network Analysis*, Lake Tahoe, NV, Dec. 2013.

[19] ——, "Dynamic structural equation models for tracking topologies of social networks," in *Proc. of Intl. Conf. on Computational Advances in Multi-Sensor Adpative Processing*, Saint Martin, Dec. 2013.

[20] ——, "Proximal-gradient algorithms for tracking cascades over social networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 563–575, Aug. 2014.

[21] ——, "A proximal splitting algorithm for tracking cascades over social networks," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing*, Firenze, Italy, May 2014.

[22] B. Baingana, P. Traganitis, G. Giannakis, and G. Mateos, "Big data analytics for social networks," in *Graph-Based Social Media Analysis*, I. Pitas, Ed. Boca Raton, FL: CRC Press, 2015, ch. 10, pp. 293–340.

[23] L. Bako, K. Boukharouba, E. Duviella, and S. Lecoeuche, "A recursive identification algorithm for switched linear/affine models," *Nonlinear Analysis: Hybrid Systems*, vol. 5, no. 2, pp. 242–253, 2011.

[24] A.-L. Barabasi and R. E. Crandall, *Linked: The new science of networks*. Random House Audio, 2010.

[25] M. Barahona and L. M. Pecora, "Synchronization in small-world systems," *Physical Review Letters*, vol. 89, no. 5, pp. (054 101)1–4, 2002.

[26] K. Barbieri, O. M. G. Keshk, and B. Pollins, "Trading data: Evaluating our assumptions and coding rules," *Conflict Management and Peace Science*, vol. 26, no. 5, pp. 471–491, Nov. 2009.

[27] A. D. Barbour and G. Reinert, "Small worlds," *Random Structures and Algorithms*, vol. 19, no. 1, pp. 54–74, 2001.

[28] A. Barrat and M. Weigt, "On the properties of small-world network models," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 13, no. 3, pp. 547–560, 2000.

[29] M. Barthélémy and L. A. N. Amaral, "Small-world networks: Evidence for a crossover picture," *Physical Review Letters*, vol. 82, no. 15, p. 3180, 1999.

[30] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha, "Robust detection of dynamic community structure in networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 23, no. 1, p. 013142, 2013.

[31] S. Basu, A. Shojaie, and G. Michailidis, "Network Granger causality with inherent grouping structure," *Journal of Machine Learning Research*, vol. 16, no. 2, pp. 417–453, Mar. 2015.

[32] J. A. Bazerque, B. Baingana, and G. B. Giannakis, "Identifiability of sparse structural equation models for directed and cyclic networks," in *Proc. of Global Conf. on Signal and Info. Processing*, Austin, TX, Dec. 2013.

[33] J. A. Bazerque and G. B. Giannakis, "Nonparametric basis pursuit via sparse kernel-based learning," *IEEE Signal Proc. Mag.*, vol. 30, no. 4, pp. 112–125, Jul. 2013.

[34] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, pp. 183–202, Jan. 2009.

[35] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[36] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[37] K. A. Bollen, *Structural Equation Models*. Wiley Online Library, 1998.

[38] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learning*, vol. 3, pp. 1–122, 2011.

[39] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, no. 1, pp. 309–320, 2000.

[40] M. Buchanan, *Nexus: Small Worlds and the Groundbreaking Theory of Networks*. WW Norton and Company, 2003.

[41] E. Bullmore and O. Sporns, "Complex brain networks: Graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3, pp. 186–198, 2009.

[42] X. Cai, J. A. Bazerque, and G. B. Giannakis, "Gene network inference via sparse structural equation modeling with genetic perturbations," in *Proc. of Intl. Workshop on Genomic Signal Processing and Statistics*, San Antonio, Texas, Dec. 2011, pp. 66–69.

[43] ——, "Gene network inference via sparse structural equation modeling with genetic perturbations," *PLoS Comp. Biology*, vol. 9, May 2013, e1003068 doi:10.1371/journal.pcbi.1003068.

[44] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.

[45] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572–596, 2011.

[46] Y. Chen, Y. Gu, and A. O. Hero III, "Sparse LMS for system identification," in *Proc. of Intern. Conf. on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, Apr. 2009.

[47] P. L. Combettes and J.-C. Pesquet, "A proximal decomposition method for solving convex variational inverse problems," *Inverse Problems*, vol. 24, no. 6, pp. 1–27, 2008.

[48] ——, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, ser. Springer Optimization and its Applications, H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Eds. Springer New York, 2011, pp. 185–212.

[49] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "DEMON: A local-first discovery method for overlapping communities," in *Proc. of the 18th ACM Intl. Conf. on Knowledge Discovery and Data Mining*, Beijing, China, Aug. 2012.

[50] I. Daubechies, M. Defrise, and C. D. Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, pp. 1413–1457, Aug. 2004.

[51] D. di Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. Eastwood, A. P. Wojtovich, S. J. Elliott, S. E. Schaus, and J. J. Collins, "Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks," *Nature Biotechnology*, vol. 23, no. 3, pp. 377–383, 2005.

[52] D. Durante and D. B. Dunson, "Nonparametric Bayes dynamic modeling of relational data," *arXiv pre-print arXiv:1311.4669*, Nov. 2013.

[53] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. New York, NY: Cambridge University Press, 2010.

[54] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *ACM SIGCOMM Computer Communication Review*. ACM, 1999, pp. 251–262.

[55] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, Feb. 2010.

[56] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, pp. 432–441, Dec. 2007.

[57] W. Fu, L. Song, and E. P. Xing, "Dynamic mixed membership blockmodel for evolving networks," in *Proc. of the 26th Annual International Conference on Machine Learning*, Montreal, Canada, Jun. 2009.

[58] G. B. Giannakis, G. Mateos, S. Farahmand, and H. Zhu, "USPACOR: Universal sparsity-controlling outlier rejection," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, May 2011.

[59] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 7821–7826, Jun. 2002.

[60] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proc. of Intl. Conf. on Advances in Social Networks Analysis and Mining*, Odense, Denmark, Aug. 2010.

[61] Q. Gu, J. Zhou, and C. H. Q. Ding, "Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs," in *Proc. of SIAM Intl. Conf. on Data Mining*, Columbus, OH, May 2010, pp. 199–210.

[62] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas, "Self-similar community structure in a network of human interactions," *Physical Review E*, vol. 68, no. 6, pp. (065 103)1–4, 2003.

[63] J. R. Harring, B. A. Weiss, and J.-C. Hsu, "A comparison of methods for estimating quadratic effects in nonlinear structural equation models," *Psychological Methods*, vol. 17, no. 2, p. 193, 2012.

[64] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. New York, NY: Springer, 2009.

[65] L. A. Hayduk, *Structural Equation Modeling with LISREL: Essentials and Advances*. Baltimore, MD: Johns Hopkins University Press, 1988.

[66] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[67] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Trans. on Signal Processing*, vol. 62, no. 1, pp. 211–224, Jan. 2014.

[68] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[69] D. Kaplan, *Structural Equation Modeling: Foundations and Extensions*, 2nd ed. Thousand Oaks, CA: Sage Publications, 2009.

[70] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, pp. (016 107)1–10, 2011.

[71] G. Karypis and V. Kumar, "Analysis of multilevel graph partitioning," in *Proc. of ACM/IEEE Conf. on Supercomputing*, San Diego, CA, Dec. 1995, p. 29.

[72] ——, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[73] G. Karypis and K. Vipin, "Multilevel k-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 96–129, 1998.

[74] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[75] A. Kelava, B. Nagengast, and H. Brandt, "A nonlinear structural equation mixture modeling approach for non-normally distributed latent predictor variables," *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 21, no. 3, pp. 468–481, Jun. 2014.

[76] J. Kleinberg, "Navigation in a small world," *Nature*, vol. 406, no. 6798, pp. 845–845, 2000.

[77] ——, "Small-world phenomena and the dynamics of information," in *Proc. of NIPS*, Vancouver, Canada, Dec. 2001, pp. 431–438.

[78] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. New York, NY: Springer, 2009.

[79] M. Kolar, L. Song, A. Ahmed, and E. P. Xing, "Estimating time-varying networks," *Ann. Appl. Statist.*, vol. 4, pp. 94–123, 2010.

[80] Y. Kopsinis, K. Slavakis, and S. Theodoridis, "Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls," *IEEE Trans. Signal Process.*, vol. 59, pp. 936–952, Mar. 2011.

[81] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *J. of Machine Learning Research*, vol. 10, pp. 719–743, Mar. 2009.

[82] D. L. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[83] ——, "Algorithms for non-negative matrix factorization," in *Proc. of Advances in Neural Information Processing Systems*, Denver, CO, Nov. 2000, pp. 556–562.

[84] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Machine Learning Research*, vol. 11, pp. 985–1042, Mar. 2010.

[85] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery in Data Mining*, Chicago, IL, Aug. 2005, pp. 177–187.

[86] J. Leskovec, "Web and blog datasets," *Stanford Network Analysis Project*, 2011. [Online]. Available: http://snap.stanford.edu/infopath/data.html

[87] Y. R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *ACM Trans. on Knowledge Discovery from Data*, vol. 3, no. 2, pp. 1–31, Apr. 2009.

[88] B. Liu, A. de la Fuente, and I. Hoeschele, "Gene network inference via structural equation modeling in genetical genomics experiments," *Genetics*, vol. 178, pp. 1763–1776, Mar. 2008.

[89] B. A. Logsdon and J. Mezey, "Gene expression network reconstruction by convex feature selection when incorporating genetic perturbations," *PLoS Comp. Biology*, vol. 6, Dec. 2010, e1001014. doi:10.1371/journal.pcbi.1001014.

[90] U. V. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[91] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. of Machine Learning Research*, vol. 11, pp. 19–60, Jan. 2010.

[92] S. Mankad and G. Michailidis, "Structural and functional discovery in dynamic networks with non-negative matrix factorization," *Physical Review E*, vol. 88, no. 4, p. 042812, Oct. 2013.

[93] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: Algorithms and applications," *IEEE Transactions on Signal Processing*, vol. 61, pp. 5374–5388, Nov. 2013.

[94] ——, "Dynamic anomalography: Tracking network anomalies via sparsity and low rank," *IEEE J. Sel. Topics Signal Process.*, vol. 7, pp. 50–66, Feb. 2013.

[95] A. McIntosh and F. Gonzalez-Lima, "Structural equation modeling and its application to network analysis in functional brain imaging," *Human Brain Mapping*, vol. 2, no. 1, pp. 2–22, 1994.

[96] N. Meinshausen and P. Buhlmann, "High-dimensional graphs and variable selection with the lasso," *Ann. Statist.*, vol. 34, pp. 1436–1462, 2006.

[97] S. Meyers and J. Leskovec, "On the convexity of latent social network inference," in *Proc. of Neural Information Proc. Sys. Conf.*, Vancouver, Canada, Feb. 2013.

[98] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[99] ——, "Smooth minimization of nonsmooth functions," *Math. Prog.*, vol. 103, pp. 127–152, 2005.

[100] M. E. J. Newman, "Models of the small world," *Journal of Statistical Physics*, vol. 101, no. 3-4, pp. 819–841, 2000.

[101] ——, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[102] ——, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, Jun. 2004.

[103] ——, "Modularity and community structure in networks," *Proc. of the Natl. Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006.

[104] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[105] P. Paatero, "Least squares formulation of robust non-negative factor analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 37, no. 1, pp. 23–35, 1997.

[106] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[107] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal, "Identification of hybrid systems: A tutorial," *European Journal of Control*, vol. 13, no. 2, pp. 242–260, 2007.

[108] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optimization*, vol. 1, pp. 123–231, 2013.

[109] J. Pearl, *Causality: Models, Reasoning, and Inference*, 2nd ed. New York, NY: Cambridge University Press, 2009.

[110] I. Ramirez and G. Sapiro, "An MDL framework for sparse coding and dictionary learning," *IEEE Trans. Signal Process.*, vol. 60, pp. 2913–2927, Jun. 2012.

[111] M. G. Rodriguez, D. Balduzzi, and B. Scholkopf, "Uncovering the temporal dynamics of diffusion networks," in *Proc. of 28th Intern. Conf. on Machine Learning*, Bellevue, WA, Jul. 2011.

[112] M. G. Rodriguez, J. Leskovec, and B. Scholkopf, "Structure and dynamics of information pathways in online media," in *Proc. of 6th ACM Intern. Conf. on Web Search and Data Mining*, Rome, Italy, Dec. 2010.

[113] E. M. Rogers, *Diffusion of Innovations*, 4th ed. New York, NY: Free Press, 1995.

[114] J. Roll, A. Bemporad, and L. Ljung, "Identification of piecewise affine systems via mixed-integer programming," *Automatica*, vol. 40, no. 1, pp. 37–50, 2004.

[115] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *Proc. of 6th ACM Intl. Conf. on Web Search and Data Mining*, Rome, Italy, Feb. 2013.

[116] R. Sandler and M. Lindenbaum, "Nonnegative matrix factorization with earth mover's distance metric," in *IEEE Conf. on Computer Vision and Pattern Recognition,*. Miami, FL: IEEE, Jun. 2009, pp. 1873–1880.

[117] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links - Part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, pp. 350–364, Jan. 2008.

[118] Y. Shen, B. Baingana, and G. B. Giannakis, "Nonlinear structural equation models for network topology inference," in *Proc. of Conf. on Information Sciences and Systems*, Princeton, New Jersey, Mar. 2016.

[119] B. Shipley, *Cause and correlation in biology: A user's guide to path analysis, structural equations and causal inference*. Cambridge University Press, 2002.

[120] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic, "A family of non-negative matrix factorizations for one-class collaborative filtering problems," in *Proc. of the ACM Recommender Systems Conference*, New York, NY, Oct. 2009.

[121] V. Solo and X. Kong, *Adaptive Signal Processing Algorithms: Stability and Performance*. Prentice Hall, 1995.

[122] P. Sprechmann, I. Ramirez, G. Sapiro, and Y. Eldar, "C-HiLasso: A collaborative hierarchical sparse modeling framework," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4183–4198, Sep. 2011.

[123] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proc. of the 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, Las Vegas, NV, Aug. 2008.

[124] P. Tseng, "Applications of a splitting algorithm to decomposition in convex programming and variational inequalities," *SIAM J. Control Optim.*, vol. 29, pp. 119–138, Jan. 1991.

[125] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the Facebook social graph," *arXiv preprint arXiv:1111.4503*, 2011.

[126] R. Vidal, "Recursive identification of switched arx systems," *Automatica*, vol. 44, no. 9, pp. 2274–2287, 2008.

[127] M. Wall and Y. Amemiya, "Estimation for polynomial structural equation models," *Journal of the American Statistical Association*, vol. 95, no. 451, pp. 929–940, Oct. 2000.

[128] Y.-X. Wang and Y. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[129] D. J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[130] J. J. Whang, P. Rai, and I. S. Dhillon, "Stochastic blockmodel with cluster overlap, relevance selection, and similarity-based smoothing," in *Proc. of the 13th IEEE Intl. Conf. on Data Mining*, Dallas, TX, Dec. 2013.

[131] S. Wright, "Correlation and causation," *Journal of Agricultural Research*, vol. 20, no. 7, pp. 557–585, 1921.

[132] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Trans. on Sig. Proc.*, vol. 57, pp. 2479–2493, 2009.

[133] M. Xiong, J. Li, and X. Fang, "Identification of genetic networks," *Genetics*, vol. 166, no. 2, pp. 1037–1052, 2004.

[134] K. S. Xu and A. O. Hero, "Dynamic stochastic blockmodels for time-evolving social networks," *IEEE Journal on Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 552–562, Aug. 2014.

[135] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. of 6th ACM Intl. Conf. on Web Search and Data Mining*, Rome, Italy, Feb. 2013.

[136] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Royal. Statist. Soc B*, vol. 68, pp. 49–67, 2006.

[137] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," *Pacific Journal of Optimization*, vol. 9, no. 1, pp. 167–180, 2009.

[138] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas, "Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 683–695, 2006.

[139] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using nonnegative matrix factorization," in *SIAM Conf. on Data Mining*, Bethesda, MD, Apr. 2006.

[140] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multidimensional hawkes processes," in *Proc. of 16th Intl. Conf. on Artificial Intelligence and Statistics*, Scottsdale, AZ, May 2013.

Appendix

A Derivation of ISTA iterations in Algorithm 2

This appendix gives a more detailed derivation of the ISTA iterations in Algorithm 2. Specializing to (2.5), note that (2.22) decomposes into

$$\begin{aligned}\mathbf{A}[k] &:= \arg\min_{\mathbf{A}} \left\{ \frac{L_f}{2}\|\mathbf{A} - \mathbf{G}_A[k-1]\|_F^2 + \lambda_t \|\mathbf{A}\|_1 \right\} \\ &= \text{prox}_{\lambda_t\|\cdot\|_1/L_f}(\mathbf{G}_A[k-1]) \qquad (4)\\ \mathbf{B}[k] &:= \arg\min_{\mathbf{B}} \left\{ \|\mathbf{B} - \mathbf{G}_B[k-1]\|_F^2 \right\} = \mathbf{G}_B[k-1] \qquad (5)\end{aligned}$$

subject to the constraints in (2.5) which so far have been left implicit, and $\mathbf{G} := [\mathbf{G}_A\, \mathbf{G}_B]$. Because there is no regularization on the matrix $\mathbf{B}$, the corresponding update (5) boils-down to a simple gradient-descent step. It follows that $\text{prox}_{\lambda_t\|\cdot\|_1/L_f}(\cdot) = \mathcal{S}_{\lambda_t/L_f}(\cdot)$, e.g., [50, 64]; so that

$$\mathbf{A}[k] = \mathcal{S}_{\lambda_t/L_f}(\mathbf{G}_A[k-1]). \qquad (6)$$

What remains now is to obtain expressions for the gradient of $f(\mathbf{V})$ with respect to $\mathbf{A}$ and $\mathbf{B}$, which are required to form the matrices $\mathbf{G}_A$ and $\mathbf{G}_B$. To this end, note that by incorporating the constraints $a_{ii} = 0$ and $b_{ij} = 0,\ \forall j \neq i, i = 1, \ldots N$, one can simplify the expression of $f(\mathbf{V})$ as

$$f(\mathbf{V}) := \frac{1}{2}\sum_{\tau=1}^{t}\sum_{i=1}^{N} \beta^{t-\tau}\|(\mathbf{y}_i^\tau)^\top - \mathbf{a}_{-i}^\top \mathbf{Y}_{-i}^\tau - b_{ii}\mathbf{x}_i^\top\|_F^2 \qquad (7)$$

where $(\mathbf{y}_i^\tau)^\top$, $\mathbf{a}_{-i}^\top$, and $\mathbf{Y}_{-i}$ have been previously defined. It is apparent from (7) that $f(\mathbf{V})$ is separable across the trimmed row vectors $\mathbf{a}_{-i}^\top$, and the scalar diagonal entries b_{ii}, $i = 1, \ldots, N$.

The sought gradients are readily obtained as

$$\nabla_{\mathbf{a}_{-i}} f(\mathbf{V}) = -\sum_{\tau=1}^{t} \beta^{t-\tau} \mathbf{Y}_{-i}^{\tau} (\mathbf{y}_i^{\tau} - (\mathbf{Y}_{-i}^{\tau})^{\top} \mathbf{a}_{-i} - \mathbf{x}_i b_{ii})$$

$$\nabla_{b_{ii}} f(\mathbf{V}) = -\sum_{\tau=1}^{t} \beta^{t-\tau} ((\mathbf{y}_i^{\tau})^{\top} - \mathbf{a}_{-i}^{\top} \mathbf{Y}_{-i}^{\tau} - b_{ii} \mathbf{x}_i^{\top}) \mathbf{x}_i.$$

Using the definitions for $\mathbf{\Sigma}^t$, $\mathbf{\Sigma}_{-i}^t$, $\boldsymbol{\sigma}_i^t$, $\boldsymbol{\sigma}_{-i}^t$, $\bar{\mathbf{Y}}^t$, and $\bar{\mathbf{Y}}_{-i}^t$, the gradient expressions for $i = 1, \ldots, N$ can be compactly expressed as

$$\nabla_{\mathbf{a}_{-i}} f(\mathbf{V}) = \mathbf{\Sigma}_{-i}^t \mathbf{a}_{-i} + \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i b_{ii} - \boldsymbol{\sigma}_{-i}^t \tag{8}$$

$$\nabla_{b_{ii}} f(\mathbf{V}) = \mathbf{a}_{-i}^{\top} \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i + \frac{1-\beta^t}{1-\beta} b_{ii} \|\mathbf{x}_i\|_2^2 - (\bar{\mathbf{y}}_i^{\tau})^{\top} \mathbf{x}_i. \tag{9}$$

From (5)-(6) and (8)-(9), the resulting parallel ISTA iterations are

$$\nabla_{\mathbf{a}_{-i}} f[k] = \mathbf{\Sigma}_{-i}^t \mathbf{a}_{-i}[k] + \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i b_{ii}[k] - \boldsymbol{\sigma}_{-i}^t \tag{10}$$

$$\nabla_{b_{ii}} f[k] = \mathbf{a}_{-i}^{\top}[k] \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i + \frac{(1-\beta^t)}{1-\beta} b_{ii}[k] \|\mathbf{x}_i\|_2^2 - (\bar{\mathbf{y}}_i^t)^{\top} \mathbf{x}_i \tag{11}$$

$$\mathbf{a}_{-i}[k+1] = \mathcal{S}_{\lambda_t/L_f} \left(\mathbf{a}_{-i}[k] - (1/L_f) \nabla_{\mathbf{a}_{-i}} f[k] \right) \tag{12}$$

$$b_{ii}[k+1] = b_{ii}[k] - (1/L_f) \nabla_{b_{ii}} f[k]. \tag{13}$$

B Proof of Proposition 2

In order to construct the proof for Proposition 2, it is necessary to state and prove Lemmas 1 and 2.

Lemma 1. *Under (as5), matrix* $\left(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top}\right)$ *is invertible with probability one.*

Proof: Note that $\det\left(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top}\right)$ is a polynomial whose variables are the nonzero entries of $\mathbf{A}^{\sigma(t)}$. Consequently, for some constants $\{\alpha_{ij}\}_{i,j}$, the probability that $\left(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top}\right)$ is not invertible can be equivalently written as

$$\begin{aligned}\Pr\{\det(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top})=0\} = &\ \Pr\{\prod_{i,j}(a_{ij}^{\sigma(t)}-\alpha_{ij})=0\}\\ = &\ \Pr\{\textstyle\bigcup_{i,j}(a_{ij}^{\sigma(t)}=\alpha_{ij})\}\\ \leq &\ \sum_{i,j}\Pr(a_{ij}^{\sigma(t)}=\alpha_{ij}).\end{aligned} \tag{14}$$

Since $a_{ij}^{\sigma(t)}$ is drawn from a continuous distribution (as5), it follows that $\Pr(a_{ij}^{\sigma(t)}=\alpha_{ij})=0$, implying that $\Pr\{\det(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top})\neq 0\}=1$. ■

Lemma 2. *Under (as4)-(as6), assuming* $kr\left(\mathbf{X}^{\top}\right)\geq 2K+1$ *implies that*

$$kr\left(\mathbf{Y}_t^{\top}\right)=kr\left(\mathbf{X}^{\top}\mathbf{B}^{\sigma(t)}\left(\mathbf{I}-\left(\mathbf{A}^{\sigma(t)}\right)^{\top}\right)^{-1}\right)\geq 2K+1$$

with probability one.

Proof: Defining $\bar{\mathbf{X}}^{\top}:=\mathbf{X}^{\top}\mathbf{B}^{\sigma(t)}$, note that $\text{kr}\left(\bar{\mathbf{X}}^{\top}\right)=\text{kr}\left(\mathbf{X}^{\top}\mathbf{B}^{\sigma(t)}\right)$. Let $\bar{\mathbf{X}}_{\mathcal{Q}}^{\top}$ denote a submatrix of $\bar{\mathbf{X}}^{\top}$, formed by selecting an arbitrary subset of columns indexed by the set $\mathcal{Q}\subset\{1,\ldots,N\}$ whose cardinality is $2K+1$. Given that $\text{kr}\left(\bar{\mathbf{X}}^{\top}\right)\geq 2K+1$, there exists a set of rows of $\bar{\mathbf{X}}_{\mathcal{Q}}^{\top}$ indexed by $\mathcal{P}\subset\{1,\ldots,C\}$ with cardinality $2K+1$, such that the resulting square submatrix $\bar{\mathbf{X}}_{\mathcal{Q},\mathcal{P}}^{\top}$ has full rank.

Since from Lemma 1, we have that $(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top})$ is invertible, it holds that $\mathbf{Y}_t^{\top}=\bar{\mathbf{X}}^{\top}\left(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top}\right)^{-1}$, or $\mathbf{Y}_t^{\top}=\bar{\mathbf{X}}^{\top}\mathbf{\Phi}$ where $\mathbf{\Phi}:=(\mathbf{I}-(\mathbf{A}^{\sigma(t)})^{\top})^{-1}$. Collecting the columns and rows of $\mathbf{Y}_t^{\top}$ indexed by $\mathcal{Q}$ and $\mathcal{P}$ into $(\mathbf{Y}_{\mathcal{Q},\mathcal{P}}^{t})^{\top}$, it follows that $(\mathbf{Y}_{\mathcal{Q},\mathcal{P}}^{t})^{\top}=\bar{\mathbf{X}}_{\mathcal{P}}^{\top}\mathbf{\Phi}_{\mathcal{Q}}$, where $\bar{\mathbf{X}}_{\mathcal{P}}^{\top}$ collects rows of $\bar{\mathbf{X}}^{\top}$ indexed by $\mathcal{P}$, and $\mathbf{\Phi}_{\mathcal{Q}}$ collects columns of $\mathbf{\Phi}$ indexed by $\mathcal{Q}$. Equivalently, one can write $(\mathbf{Y}_{\mathcal{Q},\mathcal{P}}^{t})^{\top}=\bar{\mathbf{X}}_{\mathcal{Q},\mathcal{P}}^{\top}\mathbf{\Phi}_{\mathcal{Q},\mathcal{Q}}$, where only the corresponding columns of $\bar{\mathbf{X}}_{\mathcal{P}}^{\top}$ and rows of $\mathbf{\Phi}_{\mathcal{Q}}$ are retained.

By Cramer's rule for matrix inversion, entry ϕ_{ij} of $\mathbf{\Phi}$ can be written as $\psi_{ij}(\mathbf{A}^{\sigma(t)})/\det(\mathbf{I} - \mathbf{A}^{\sigma(t)})$, where $\psi_{ij}(\mathbf{A}^{\sigma(t)})$ denotes a polynomial function of degree at most $(N-1)$, with nonzero entries of $\mathbf{A}^{\sigma(t)}$ as variables. Furthermore, the entries of $(\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top$ are linear combinations of N entries of $\mathbf{\Phi}_{\mathcal{Q}}$, while the determinant of $(\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top$ is a polynomial $\psi_{\text{det}}((\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top)$ of degree $2K+1$, with its entries as variables. As a result, one can write the determinant of $(\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top$ as

$$\det\left((\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top\right) = \frac{\bar{\psi}_{\text{det}}\left(\mathbf{A}^{\sigma(t)}\right)}{\det\left(\mathbf{I} - \mathbf{A}^{\sigma(t)}\right)^{(2K+1)N}} \tag{15}$$

where $\bar{\psi}_{\text{det}}\left(\mathbf{A}^{\sigma(t)}\right)$ is a polynomial of degree at most $(2K+1)N(N-1)$ with nonzero entries of $\mathbf{A}^{\sigma(t)}$ as variables, and is formed by the composition of the polynomials $\psi_{ij}(\mathbf{A}^{\sigma(t)})$ and linear combinations of entries of $\mathbf{\Phi}_{\mathcal{Q}}$.

It can be shown that $\bar{\psi}_{\text{det}}\left(\mathbf{A}^{\sigma(t)}\right)$ is not identically zero as follows. Since $\mathbf{A}^{\sigma(t)} = \mathbf{0}$ implies that $\mathbf{\Phi} = \mathbf{I}$, then $\det\left((\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top\right) = \det(\bar{\mathbf{X}}^\top_{\mathcal{Q},\mathcal{P}}) \neq 0$, because $\bar{\mathbf{X}}^\top_{\mathcal{Q},\mathcal{P}}$ is full rank by construction. Consequently, $\bar{\psi}_{\text{det}}\left(\mathbf{A}^{\sigma(t)}\right)$ depends on nonzero entries of $\mathbf{A}^{\sigma(t)}$, which are drawn from a continuous distribution. Similar to the argument made in the proof of Lemma 1, the probability that $\bar{\psi}_{\text{det}}\left(\mathbf{A}^{\sigma(t)}\right) = 0$ is zero. Since $(\mathbf{Y}^t_{\mathcal{Q},\mathcal{P}})^\top$ has full rank, then $\text{kr}\left(\mathbf{Y}_t^\top\right) \geq 2K+1$ with probability one. ■

Proposition 2: *Suppose data matrices $\mathbf{Y}_t$ and $\mathbf{X}$ adhere to (3.4) with $a^{\sigma(t)}_{ii} = 0$, $b^{\sigma(t)}_{ii} \neq 0, \ \ \forall\, i,t$, and $b^{\sigma(t)}_{ij} = 0 \ \ \forall\, i \neq j, t$. If (as1)-(as6) hold, and $p_s > 0 \ \ \forall s$, then $\{\mathbf{A}^s, \mathbf{B}^s\}_{s=1}^S$ can be uniquely identified with probability one as $T \to \infty$.*

Proof: First rewrite (3.4) as

$$\mathbf{Y}_t^\top \mathbf{F}_t = \mathbf{X}^\top \tag{16}$$

where $\mathbf{F}_t := (\mathbf{I} - (\mathbf{A}^{\sigma(t)})^\top)(\mathbf{B}^{\sigma(t)})^{-1}$. Note that both $\mathbf{A}^{\sigma(t)}$ and $\mathbf{B}^{\sigma(t)}$ can be uniquely determined from $\mathbf{F}_t$. Specifically, with $a^{\sigma(t)}_{ii} = 0 \ \ \forall i$, then $\text{Diag}(\mathbf{F}_t) = (\mathbf{B}^{\sigma(t)})^{-1}$, implying that $\mathbf{B}^{\sigma(t)} = (\text{Diag}(\mathbf{F}_t))^{-1}$ and $\mathbf{A}^{\sigma(t)} = \mathbf{I} - \mathbf{B}^{\sigma(t)}\mathbf{F}_t^\top$. As a result, identifiability of $\mathbf{F}_t$ implies identifiability of both $\mathbf{A}^{\sigma(t)}$ and $\mathbf{B}^{\sigma(t)}$, and the argument is by contradiction.

If there exists $\bar{\mathbf{F}}_t$ satisfying (16), then $\mathbf{Y}_t^\top \tilde{\mathbf{F}}_t = \mathbf{0}$, where $\tilde{\mathbf{F}}_t := \mathbf{F}_t - \bar{\mathbf{F}}_t$. Since the columns of $\tilde{\mathbf{F}}_t$ have at most $2K+1$ nonzero entries each, and $\text{kr}\left(\mathbf{Y}_t^\top\right) \geq 2K+1$ (Lemma 2), it follows that $\tilde{\mathbf{F}}_t = \mathbf{0}$. Hence, $\mathbf{F}_t = \bar{\mathbf{F}}_t$ which is a contradiction. Consequently, $\mathbf{F}_t$ is uniquely identifiable, and so are $\mathbf{A}^{\sigma(t)}$ and $\mathbf{B}^{\sigma(t)}$ for all t.

Recalling that $\mathbf{A}^{\sigma(t)} = \sum_{s=1}^{S} \chi_{ts}\mathbf{A}^s$ and $\mathbf{B}^{\sigma(t)} = \sum_{s=1}^{S} \chi_{ts}\mathbf{B}^s$, the unique solution from (16) coincides with a specific pair in the set $\{(\mathbf{A}^s, \mathbf{B}^s)\}_{s=1}^{S}$ per interval t. Following a similar argument to the proof of Proposition 1, recall that χ_{ts} are Bernoulli random variables with $p_s = \text{Pr}(\chi_{ts} = 1)$ per t. Hence, the number of times state s is activated over T time intervals follows a binomial distribution. With $T_s := \sum_{t=1}^{T} \chi_{ts}$ denoting the total number of activations of state s over $t = 1, \ldots, T$, it turns out that $\text{Pr}(T_s \geq 1) = 1 - \text{Pr}(T_s = 0) = 1 - (1 - p_s)^T$. Given that $\lim_{T\to\infty} \text{Pr}(T_s \geq 1) = 1$ only if $p_s > 0$, all states can be uniquely identified with probability 1 as $T \to \infty$, whenever $p_s > 0 \ \ \forall s$.

■

C Derivation of decentralized updates in (4.34)

Recalling the constraint $\bar{\mathbf{X}}_{nm} = \tilde{\mathbf{X}}_{nm}$, the per-iteration update in (4.34) becomes

$$\bar{\mathbf{X}}_{nm}[k+1] = \underset{\bar{\mathbf{X}}_{nm}}{\arg\min} \ \frac{\rho}{2}\Big\{\|\mathbf{V}_m[k+1] - \bar{\mathbf{X}}_{nm}\|_F^2 + \|\mathbf{V}_n[k+1] - \bar{\mathbf{X}}_{nm}\|_F^2\Big\} \\ - \mathrm{Tr}\left\{\left(\bar{\mathbf{\Pi}}_{nm}[k+1] + \tilde{\mathbf{\Pi}}_{nm}[k+1]\right)^\top \bar{\mathbf{X}}_{nm}\right\} \quad (17)$$

whose solution turns out to be

$$\bar{\mathbf{X}}_{nm}[k+1] = \frac{1}{2\rho}\left(\bar{\mathbf{\Pi}}_{nm}[k+1] + \tilde{\mathbf{\Pi}}_{nm}[k+1]\right) + \frac{1}{2}\left(\mathbf{V}_m[k+1] + \mathbf{V}_n[k+1]\right). \quad (18)$$

Assuming that $\bar{\mathbf{\Pi}}_{nm}[0] = -\tilde{\mathbf{\Pi}}_{nm}[0]$, then it can be shown by a simple inductive argument that $\bar{\mathbf{\Pi}}_{nm}[k] = -\tilde{\mathbf{\Pi}}_{nm}[k], n \in \mathcal{N}_m$ [93]. Consequently,

$$\bar{\mathbf{X}}_{nm}[k] = \frac{1}{2}\left(\mathbf{V}_m[k] + \mathbf{V}_n[k]\right) \quad (19)$$

and

$$\bar{\mathbf{\Pi}}_{nm}[k+1] = \bar{\mathbf{\Pi}}_{nm}[k] + \frac{\rho}{2}\left(\mathbf{V}_m[k] - \mathbf{V}_n[k]\right). \quad (20)$$

Due to (19), note that $\bar{\mathbf{X}}_{nm}[k] = \bar{\mathbf{X}}_{mn}[k], n \in \mathcal{N}_m$, and that if $\bar{\mathbf{\Pi}}_{nm}[0] = -\bar{\mathbf{\Pi}}_{mn}[0]$, then $\bar{\mathbf{\Pi}}_{nm}[k] = -\bar{\mathbf{\Pi}}_{mn}[k], n \in \mathcal{N}_m$.

Focusing on the update for $\mathbf{V}_m$ in (4.32), and dropping the irrelevant terms yields

$$\underset{\mathbf{V}_m}{\arg\min} \quad \Psi_{\lambda_t}(\mathbf{U}_m[k], \mathbf{O}_m[k], \mathbf{V}_m, \mathbf{S}_m^t, s^t) + \mathrm{Tr}\left\{\sum_{n\in\mathcal{N}_m}\left(\bar{\mathbf{\Pi}}_{nm}^\top[k+1](\mathbf{V}_m - \bar{\mathbf{X}}_{nm}[k])\right)\right\} \\ + \frac{\rho}{2}\sum_{n\in\mathcal{N}_m}\|\mathbf{V}_m - \bar{\mathbf{X}}_{nm}[k]\|_F^2. \quad (21)$$

Eliminating constants in the second term, one obtains

$$\mathrm{Tr}\left\{\sum_{n\in\mathcal{N}_m}\bar{\mathbf{\Pi}}_{nm}^\top[k+1]\mathbf{V}_m\right\} = \mathrm{Tr}\left\{\mathbf{V}_m^\top\bar{\mathbf{\Pi}}_m[k+1]\right\} \quad (22)$$

where $\bar{\mathbf{\Pi}}_m[k+1] := \sum_{n\in\mathcal{N}_m}\bar{\mathbf{\Pi}}_{nm}[k+1]$. Using (20),

$$\sum_{n\in\mathcal{N}_m}\bar{\mathbf{\Pi}}_{nm}[k+1] = \sum_{n\in\mathcal{N}_m}\bar{\mathbf{\Pi}}_{nm}[k] + \frac{\rho}{2}\left(|\mathcal{N}_m| - \sum_{n\in\mathcal{N}_m}\mathbf{V}_n[k]\right) \quad (23)$$

leading to the dual variable update

$$\bar{\mathbf{\Pi}}_m[k+1] = \bar{\mathbf{\Pi}}_m[k] + \frac{\rho}{2}\left(|\mathcal{N}_m| - \sum_{n\in\mathcal{N}_m} \mathbf{V}_n[k]\right). \tag{24}$$

Expansion of the third term in (21) leads to

$$\frac{\rho}{2}\sum_{n\in\mathcal{N}_m} \|\mathbf{V}_m - \bar{\mathbf{X}}_{nm}[k]\|_F^2 = \frac{\rho}{2}\mathrm{Tr}\left\{|\mathcal{N}_m|\mathbf{V}_m^\top\mathbf{V}_m - 2\mathbf{V}_m^\top \sum_{n\in\mathcal{N}_m} \bar{\mathbf{X}}_{nm}\right\} \tag{25}$$

and (19) can be used to further expand $\sum_{n\in\mathcal{N}_m} \bar{\mathbf{X}}_{nm}$ as follows

$$\sum_{n\in\mathcal{N}_m} \bar{\mathbf{X}}_{nm} = \frac{|\mathcal{N}_m|}{2}\mathbf{V}_m[k] + \frac{1}{2}\sum_{n\in\mathcal{N}_m} \mathbf{V}_n[k]. \tag{26}$$

Upon substituting (26) into (25), it turns out that

$$\frac{\rho}{2}\sum_{n\in\mathcal{N}_m} \|\mathbf{V}_m - \bar{\mathbf{X}}_{nm}[k]\|_F^2 = \frac{\rho}{2}\mathrm{Tr}\left\{|\mathcal{N}_m|\mathbf{V}_m^\top\mathbf{V}_m - 2\mathbf{V}_m^\top\left(|\mathcal{N}_m|\mathbf{V}_m[k] + \sum_{n\in\mathcal{N}_m}\mathbf{V}_n[k]\right)\right\} \tag{27}$$

and $\mathbf{V}_m[k+1]$ can be obtained by solving

$$\begin{aligned}\mathbf{V}_m[k+1] = \underset{\mathbf{V}_m\in\mathbb{R}_+^{N\times C}}{\arg\min} \quad & \Psi_{\lambda_t}(\mathbf{U}_m[k], \mathbf{O}_m[k], \mathbf{V}_m, \mathbf{S}_m^t, s^t) \\ & + \mathrm{Tr}\Big[(\rho/2)|\mathcal{N}_m|\mathbf{V}_m^\top\mathbf{V}_m + \mathbf{V}_m^\top\Big(\bar{\mathbf{\Pi}}_m[k+1] \\ & - (\rho/2)\{|\mathcal{N}_m|\mathbf{V}_m[k] + \sum_{n\in\mathcal{N}_m}\mathbf{V}_n[k]\}\Big)\Big]\end{aligned} \tag{28}$$

whose closed-form solution is readily available. The remaining updates for $\mathbf{U}_m[k+1]$, $\mathbf{O}_m[k+1]$, and $\mathbf{P}_m[k+1]$ follow in a straightforward manner from the ADMM primal variable updates, and they are all available in closed form. Note that solving for $\mathbf{P}_m[k+1]$ entails completion of squares, resulting in a standard Lasso problem whose per-entry solutions are available in closed form using the *soft thresholding* operator.