On the application of Bayesian inference to network estimation problems

Caitlin Gray

November, 2020

Thesis submitted for the degree of
Doctor of Philosophy
in
Applied Mathematics
at The University of Adelaide
Faculty of Engineering, Computer and Mathematical Sciences
School of Mathematical Sciences

THE UNIVERSITY
of ADELAIDE

Contents

List of Tables

List of Figures

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed: .... .. Date: 24/7/2020

Acknowledgements

This thesis has become a reality, in no small part, from the kind and generous support of many people. I truly hope they all know how grateful I am.

To my amazing supervisors Professor Matt Roughan and Dr. Lewis Mitchell, this thesis does not exist without the two of you. For the mentorship, the inspiration, the opportunities and the generosity. For the countless hours over coffee, coke, fidget spinners and knick knacks; my figure labels will forever be large and informative, and I am sure I have learnt something about research too. The profound impact your guidance has had on me both personally and professionally cannot be understated.

To the maths chats crew: Dennis, Angus, Andrew, Max and Phill, for the weekly chats. An absolute highlight of my week, each of you have provided such valuable insight at every step and got me out of many a deep dark hole of notational despair.

And finally, to my family and friends for their wisdom and encouragement. There are, of course, too many to mention them all. To Vanessa, for showing me how it is done and being the 'essa' to my 'cat'. To Angus, for the insight and coffee. To Alex, Alex and Tim for being there from the start, these 7 years have been an absolute pleasure. To Mum, for understanding even without understanding. To Simon, for everything.

Funding

This work has been supported by the Data to Decisions CRC (D2D CRC), CSIRO's Data61, the ARC Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS) and an Australian Government Research Training Program (RTP) Scholarship.

Abstract

Interconnected network structures play a crucial role in many aspects of our lives. Understanding these networks and the dynamic processes that can propagate over them gives rise to many interesting questions. While many network and dynamical models are probabilistic, many of the current methods to solve key network science problems do not account for the uncertainties in the systems. In this thesis we look at networks in a probabilistic setting and use Bayesian methods to simulate from distributions of these networks to address key network science problems. This allows deeper understanding about the system of interest and allows us to quantify uncertainty for improved decision making. Specifically, in Chapters 3 and 4 we develop an algorithm that simulates connected random networks and explore how variations to the algorithm can improve performance of sampling the high dimensional posterior. This develops techniques that we further apply in Chapters 5 and 6 to an interesting inverse problem in social network analysis - the 'Network Inference problem'. The underlying premise is to infer network structure, how people are connected, when we can only observe things moving between actors in the social network, e.g., tweets of news articles. By using a Bayesian method we can provide uncertainty estimates for the inferences we make. We consider a variety of methods to extend the applicability to a wide range of data types, including streaming data, and test the inference on both simulated and real data. Finally, in Chapter 7, we consider the 'Network Tomography problem', which aims to infer node properties when we only have information about paths in the network. We highlight the benefits of Bayesian inference methods in two specific applications to aid decision making when identifying routing policies over the internet. Throughout this work we highlight that applying Bayesian inference techniques to novel applications in network science can expand the types of networks we can simulate, provide uncertainty quantification for making informed decisions and gain results and insight in low and messy data regimes.

Thesis publications and manuscripts

Published Journal Articles

Gray, C., Mitchell, L., and Roughan, M. 'Generating connected random graphs'. *Journal of Complex Networks*, 7, 6 (2019), 896-912.

Gray, C., Mitchell L., and Roughan M. 'Bayesian Inference of Network Structure From Information Cascades'. *IEEE Transactions on Signal and Information Processing over Networks*, 6 (2020) 371-381.

Gray, C., Mosig, C., Roughan, M., Waehlisch, M., Pelsser, C., Bush, R., and Schmidt, T. 'BGP beacons, Network Tomography, and Bayesian computation to locate Route Flap Damping.' *Proceedings of the ACM Internet Measurement Conference*, (2020) 492-505.

Chapter 1

Introduction

Networks can be used to explain and understand our interconnected complex world. They represent the interaction framework between the basic elements of complex systems.

Each day we interact with a wide variety of network structures, both physically and virtually. We interact with the power and water networks as we get ready for work, the transportation network on our commute and our social networks over coffee. We also socialise virtually using online social network platforms, like Facebook and Twitter, through the world wide web which is powered by the networked infrastructure of the internet. Network science is a large and diverse field that develops tools and techniques to analyse and derive insight from these networks.

To add another layer of complexity beyond the networks themselves, we often observe dynamics occurring on these networks and some networks themselves are dynamic [68]. Many spreading processes operate over networks: both information and disease flow readily across human social connections, malicious viruses spread through computer networks and internet protocols propagate network reachability information across the autonomous systems.

These complex systems present and interesting area of research and it has given rise to interesting models to gain understanding about the networks through the observation of dynamics, especially in the face of uncertainty It is important to consider that these complex systems contain inherent uncertainties; in the networks themselves, the dynamics observed and the data collected to measure them. Moreover, many networks of interest are unobservable, e.g., social networks, the Internet infrastructure, disease contact networks *etc.* Bayesian methods are a suite of techniques that can be used to make inferences and decisions while quantifying uncertainty. Throughout this work we highlight the wide, and as yet under-explored,

applicability of Bayesian inference in the space of network science by demonstrating how probabilistic methods can provide insight that is lost through treating the system as fixed. In a Bayesian framework we will consider distributions of networks, where each possible network has some associated probability, which is often unknown. To gain insight from these distributions we will often refer to simulating or sampling networks from these distributions. Here, we mean this in a Bayesian sense in which we estimate some distribution of networks by taking many possible samples from this distribution[1]. In some cases the entire distribution is needed, in others on a single sample is required. One may consider sampling synonymous with simulation in this context.

This thesis has three main sections, each highlighting different uses of Bayesian inference for sampling networks, inferring network structure and inferring network node and edge properties. The three sections: Chapters 3 & 4, Chapters 5 & 6 and Chapter 7, while complementary, can be read individually.

We begin in the heart of network simulation to sample networks from given random graph models to highlight the flexibility of Bayesian methods. Sampling networks from a given model is a common scenario in the study of networks and ranges from easy to complex, both conceptually and computationally. Often many samples of networks are required to estimate properties that cannot be derived analytically. Additionally, samples from a specific model serve as a 'null model' with which real-world networks can be compared, so that we can determine which properties are 'surprising' given the null model. For instance, by comparing the structure of protein networks [90] with a null model that retains some properties of the real network, researchers could determine that interaction and regulatory networks favour links between highly connected and low-connected pairs of proteins. When we use samples of random networks as null-models, we must ensure that the model accounts for other known properties in the network. Therefore, we need flexibility in the properties of the networks we can sample. Chapters 3 and 4 develop Bayesian methods to sample networks with arbitrary properties, with a detailed application to sampling connected networks.

Connectivity is a common property of real world networks. Physical networks are often connected by design, usually due to the constraint that information should reach every part of the network. For instance, internet routing requires packets to be able to reach any part of the network, often by many different paths. Additionally, often we are interested in the large connected component of social networks that participate in message passing. For example, in an epidemic the infected individuals form a connected contact network. A similar idea holds

[1]This is in contrast to surveying, or taking a subset, of a network of interest. e.g., sampling a subsection of the Twitter network.

with information moving through social networks, although we must consider the potential for outside sources of information. Hence, sampling networks according to some model, but with the guarantee of connectivity is an important task.

Another interesting problem in network science is to understand and infer underlying network structures from how information spreads among people. Gathering the network structure directly can present some issues: "Is it computationally feasible to collect the network structure? Do the required methods even exist?" On the other hand, collecting information about processes that move over a network, for example information sharing, can be relatively easy. This creates an inverse problem. We have data that was produced by a process on a network and from this we wish to infer the network structure. This specific inverse problem is often referred to as the 'Network Inference problem'. Chapter 5 tackles this problem in a Bayesian context to estimate network structure and quantify the uncertainty of the estimates obtained. We show that our method can not only provide accurate estimates of the network structure, but also provide estimates with quantified uncertainty in low data regimes. Chapter 6 extends the work of Chapter 5 to infer directed networks, consider streaming data and improve the efficiency of the algorithm.

Social networks are not the only network with dynamical processes occurring over them. The internet is a canonical example of a complex network over which many processes take place. The internet is, at the basic level, a forwarding network that sends messages, in the form of packets, between computers. These packets move hop-by-hop through specialised computers called *routers* to reach their destination. The network originated as a military project, called ARPANET [29], designed to have a decentralised computer network system. Since then, there has been an expansion in the number of internet service providers that own and operate these routers. The decentralisation, while beneficial in many ways, often prevents global knowledge of the network and the protocols each service provider is implementing. This is common in real world networks: while it may be possible to gather information about the local structure, the entire network structure of the network is unknown.

The internet is a complex system with many actors controlling different parts of the network and so nodes can exhibit different properties in how they transmit information. Internet scientists are interested in how these entities employ certain processes. In contrast to social information sharing which often propagates in a tree like manner — one individual can spread the message to many others — the internet transmits information through paths. Packets of information move over the network from one node to the next through a single path of the network. It is often easy to collect path data by sending packets through the network.

The 'Network Tomography problem' uses data gathered about paths in the network to infer node or link properties. We focus on binary network tomography that attempts to identify whether nodes show a certain property from path data. However, binary tomography is NP-hard in general and most solutions to the problem provide poor results when there is insufficient data or noise in the system or measurements. In Chapter 7 we frame the binary network tomography problem in a probabilistic setting and use Bayesian inference to infer nodes displaying a particular property based on labelled path data. We apply this specifically to nodes displaying a phenomenon called Route Flap Damping (RFD), in which nodes in the internet suppress routing information that is causing congestion. The Bayesian framework we develop allows for precise identification of nodes displaying RFD, and we show that the method generalises to other properties of interest. We show that up to 10% of nodes display this property, a far higher estimate than some claims that RFD was completely eliminated from the internet.

Throughout this thesis we highlight the wide applicability of Bayesian inference methods and their benefits in providing uncertainty estimates in network science problems. In Chapter 8 we conclude with a summary and an outlook for future work.

Chapter 2

Background

This thesis lies in the intersection between network science and Bayesian inference. We must, therefore, start out with a solid background of the important aspects of both of these fields. In this chapter, we will introduce both network science, with a focus on terminology and network models, and a variety of Bayesian inference methods that will be used throughout the thesis. We will introduce current and previous literature for further reading and highlight more specific related work in each chapter.

2.1 Networks

The foundation of graph and network theory has long been attributed to Leonhard Euler with his solution to the Königsberg bridge problem regarded as the first paper on graph theory. This area, originally a branch of discrete mathematics, has laid the foundations for network science applications in all areas of science, from chemistry and computer technology to sociology and linguistics [100]. Euler describes a *network* or *graph* in its most abstract form as a collection of points, called nodes, connected by lines, called edges. Throughout this work we will refer to networks and graphs, nodes and vertices as well as edges and links as interchangeable pairs.

Networks are everywhere in our lives; we use them, make them and are even part of them. The Internet, ecological food networks, social networks, transport systems — the list goes on. A leading introductory textbook on networks, aptly named 'Networks', by Mark Newman [100] presents a thorough overview of these networks, and more, and how they came to be. Networks are often complex systems and

network science is often used to abstract some of the details, for example why nodes act in certain ways, and investigate the connectivity structure. Nonetheless, there are enough aspects of these networks to fill the large research area of 'Network Science' – stemming from and closely related to Graph Theory, although slightly more applied. Network science covers the tools for working with networks, from the simple node-edge network representation of Erdös, to mathematical prediction of dynamics over networks.

Newman [100] splits networks into three overarching types: social networks, information networks and biological networks. Table 2.1, taken from 'Networks' [100], highlights some examples from each of these categories.

The study of social networks is so prevalent these days that it can be considered a field of study itself. While many consider social networks to refer to the online social networks of Facebook or Twitter, the study of the interaction between individuals emerged as early as the beginning of the 20th century with Jacob Moreno [94]. A social network is a collection of individuals or groups that can be represented as vertices of the network and relationships of different types, such as friendship, business, kinship or political relations, which form the edges between them. Mathematical understanding of these connections can help us to answer important questions. For example, "Who are the important actors in a network? And, how do networks contribute to the spread of communicable diseases?"

Information networks, by far, contain the most diverse set of networks. An example that is particularly interesting — one could argue the most widely studied — is the physical Internet. Depending on the applications there can be slightly different ways to define the Internet, but often we have computers or routers as nodes and cables as edges. One interesting characteristic of the Internet is that although 'we' built the network, we do not actually know the global structure; largely due to its decentralised construction. It is important to understand the structure to ensure it is efficient and robust. As a result Internet measurement is itself an entire field of research [120]. In contrast to the physical Internet the world wide web is a logical network that connects web pages to each other via hyperlinks. This is also a widely studied network as often links are created between similar topics, and so study of the structure can give us some understanding of how 'knowledge' is organised.

To understand these networks, and tackle specific problems like network inference and tomography, we must first define some basic terminology and introduce ways to model these networks.

Table 2.1: Vertices and edges in networks. [100]: Table 6.1

Network	Vertex	Edge
Internet	Computer or router	Cable or wireless data connection
World wide web	Web page	Hyperlink
Citation Network	Article, patent or legal case	Citation
Power grid	Generating station or substation	Transmission line
Friendship network	Person	Friendship
Metabolic network	Metabolite	Metabolic reaction
Neural Network	Neuron	Synapse
Food web	Species	Predation

2.1.1 Fundamentals

We will consider a *graph* or *network* $G = (N, E)$ with vertex set $N = \{N_1, \dots N_n\}$ of size n and edge set E of length e, where $E \subset N \times N$. A *Node pair* or *dyad* (i, j) is an *edge* if $(i, j) \in E$. There are many ways to represent a network, each with its own benefits either mathematically or computationally. The most intuitive is the matrix representation in the form of an *adjacency matrix*. For a network with n nodes labelled $1, \dots, n$, the adjacency matrix A has elements

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

In an *undirected* network, intuitively, links can go both ways, so if $(i, j) \in E$ then $(j, i) \in E$. Undirected graphs have symmetric adjacency matrices. A *directed* graph contains nodes in which node i is connected to node j but the converse is not necessarily true. In general, two nodes can be connected with more than one link, and so graphs can contain *multi-edges* in which a pair of nodes is connected with more than one edge, and so the adjacency matrix can have $A_{ij} > 1$. Nodes can also have *self-edges* that connect a node i, as one might guess, to itself, resulting in $A_{ii} = 1$. Both links and nodes can also have *attributes*. The most common attribute is an edge weight that can result in a *weighted adjacency matrix*, but nodes can also have names or types attributed to them.

A basic metric to understand the connectivity properties of networks is the *degree* of the nodes. The degree, z_i, of a node i is the number of edges that attach to it. The average degree of a graph, z, is the mean number of connections of each node. In directed graphs, a node has both an *in-degree*, counting the number of edges pointing towards the node, and an *out-degree*, which refers to the number of edges that emanate from it. The *neighbourhood* of a node is the set of nodes to which

an edge exists. Similarly to the degree, nodes can have both an *out-neighbourhood* and an *in-neighbourhood*.

The *degree distribution* is one of the most fundamental network properties and describes the frequency of node degrees [100]. It is often given in terms of p_k, where p_k is the fraction of nodes in the network that have degree k. There are many distributions one can imagine the node degrees to follow. The Poisson distribution is found for the Erdős-Rényi random graph, while heavy tailed distributions are found in many real-world complex networks [4].

Network connectivity plays an important role in the dynamics that can be observed on top of the network. A *path* exists between two nodes i and j if there is a sequence of nodes, $i = n_1, \ldots, n_p = j$, such that $(n_j, n_{j+1}) \in E$. Two nodes i and j are said to be *connected* if there exists a path between the two nodes, and a graph is connected if all pairs of nodes are connected.

Paths are vital in modelling information transmission, especially if all communication channels are fully captured in the network. The *shortest path* between nodes i and j is often called the *geodesic* and the length of the largest geodesic is the *diameter* of the graph. This can be the number of hops from i to j or nodes may have some notion of distance between them. In this case edges have length d_{ij} and the shortest path has an intuitive meaning. The shortest path between two nodes is an important metric in communication and transport networks, as optimal pathways can save time and resources. The average shortest path is the average length of geodesics over all pairs of nodes [9].

Networks can be split into subgraphs. A *subgraph* is a set of nodes and edges selected from the original graph. A subgraph can be used to investigate nodes that display a certain property, for example a subgraph of nodes with $z_i < 5$. When nodes are the focus of the analysis, we use a vertex-induced subgraph. The nodes of interest and all edges that connect them are taken from the original graph. This is in comparison to an edge-induced subgraph in which certain edges are taken from the original graph with the vertices that are attached to them.

2.1.2 Random graphs

Random graphs have been investigated for centuries as a primary model of interconnected systems as well as to understand the impact of inherent network properties on the system itself and in contrast to other systems. Random graphs are developed to have properties that mimic real world networks and can help to understand how their properties impact network dynamics. A random graph is a model of a network that can be described by a probability distribution, $P(G)$, over

the space of all possible graphs G. $P(G)$ can have a closed form or be described as a generation process to create the graph G. In random graphs some parameters or properties are fixed but the actual connections of the graph will differ each time they are created. A single sample from the model is a single *realisation* of a statistical *ensemble* of all possible combinations of connections [9]. Considering an ensemble rather than a specific realisation allows researchers to understand the impact of conserved network properties rather than specific connections.

There are an ever growing number of ways to model networks with random graphs [99, 100], and here we will introduce some basic models that will be applicable in later chapters.

Erdős-Rényi random graphs

Paul Erdős, Alfréd Rényi and Edgar Gilbert studied in detail the graphs known today as Erdős-Rényi (ER) graphs, or even just the random graph model [45, 52]. Although initially studied by Salomonoff and Rapoport [114], it is more often associated with Erdős and Rényi due to their publication of several seminal papers on the graph. Gilbert defines the $G(n, m)$ random graph as a function of the number of nodes n and edges m [52]. To create the graph, m node pairs are chosen uniformly at random and edges added between them. To create simple graphs, with no multi-edges or self-edges, distinct node pairs should be chosen without replacement. If multigraphs are desired simply choose node pairs with replacement. The Erdős-Rényi (ER) graph is denoted $G(n, p)$ and consists of n nodes with each node pair having an edge with probability p. The ER graph is more mathematically tractable as the inclusion of the parameter p allows probabilistic methods to be used. Erdős and Rényi show the structural properties of the graphs change with the probability parameter p. Basic properties can be calculated for the ensemble, for example the average degree is $z = (n-1)p$, and the degrees of each node follow a Poisson distribution. All ER networks with n nodes, m edges, and parameter p have the same likelihood,

$$P(G) = p^m (1-p)^{\binom{n}{2}-m}. \tag{2.2}$$

The ER random graph is one of the most widely studied graph models. It has been vital in understanding how real world networks behave, particularly which properties are statistically significant. While one of its major benefits is its relative simplicity, the ER graph lacks some important properties of real world graphs. For example, real world networks often have heterogeneous edge probabilities, varying levels of clustering and a lack of large scale structure. This has led to

the development of countless random graph models to better model real world networks we observe [99, 100]. Next, we will focus on one extension of the ER graph to incorporate heterogeneous edge probabilities.

Waxman graphs

Spatially embedded random networks (SERNs) are, unsurprisingly, random networks that have a spatial structure. They stem from the notion that the probability or strength of a link existing in real world networks is often dependent on the distance between the nodes. Often, in physical networks, longer links are more expensive to build or maintain and so are less likely than shorter links. Many real world networks display this spatial structure, and so various types of SERNs are used in social, infrastructure and epidemiological modelling [6, 13, 36, 78]. For example, data networks require cables that are naturally cheaper between nodes that are closer together. Social networks also display spatial structure as it is 'easier' to maintain relationships with those that are close to us in proximity [122].

One of the most well known SERNs is the Waxman graph [143]. The Waxman graph uses a Euclidean distance metric between points and the probability of a link decreases exponentially with distance. These graphs do not prevent long links, but reduce their probability and allow for the ratio of short to long links to be tuned for desired properties of the graph.

We create a SERN by placing n nodes uniformly at random within some defined region R of a metric space Ω with distance metric $d(x, y)$ which relates the coordinates of node x and y to a distance. For ease, we consider R to be the unit square with a Euclidean distance metric. Each pair of nodes is made adjacent independently, with probability p_{ij}, which is a function of edge length, d_{ij}. In the Waxman case,

$$p_{ij} = qe^{-sd_{ij}}, \tag{2.3}$$

for $q, s \geq 0$. The parameter s controls the extent to which spatial structure is incorporated into the graph. When $s = 0$ we recover the ER random graph, with edge probability q. In general, the q value controls the overall edge density in the graph. We note that the parameterisation in (2.3) differs from much of the literature on Waxman graphs. Unfortunately, the parameters traditionally used, (α, β), are often used interchangably with little explanation, not only in the literature but also in open source implementations of these networks [62]. Therefore, we chose to follow recent work [119] and use this parameterisation in (2.3).

Some basic properties of the Waxman graph can be derived; we are particularly

interested in the average degree of the network to control the sparsity. For instance, it is shown [119] that the average node degree is given by

$$z = (n-1)q\tilde{G}(s), \tag{2.4}$$

where $\tilde{G}(s)$ is the Laplace transform of the probability density function between a pair of random points (the Line-Picking Problem), see references for further details [50, 119].

In this section we have considered two important random graph models that will be useful in the coming chapters. However, as mentioned there are of course many important models. The well known Configuration Model creates a random graph with arbitrary degree distributions by defining a sequence of degrees that each node will have and connecting them randomly [28,100]. Another well known model is the Barabási-Albert model that generates graphs with a power-law degree distribution. Watts and Strogatz described the 'small-world' model to account for clustering and characteristically small path lengths observed in social networks. Stochastic block models (SBMs), also known as planted partition models or inhomogeneous random graphs, are widely studied class of models that are used to discover the latent 'block' structure of networks and are often used for detecting communities in networks [1]. Next, we consider how to sample graphs from these, and other, ensembles.

2.1.3 Sampling graphs

Often, we would like to generate graphs from a random graph ensemble for other applications, but the probability $P(G)$ is often intractable. Instead we use a generative process that creates graphs from the desired ensemble. Here we give a brief overview of three possible techniques to motivate the simulation method developed in Chapter 3.

Generative models Sometimes it is easy to sample a random graph that comes from the underlying ensemble. Simple random graph ensembles, such as the Erdős-Rényi model and many basic SERN models, have a very simple generative process. I.e., choose the desired number of nodes, n, and connect each node pair with probability p. Other random graph models are defined by their generative heuristic and subsequent investigation of the ensemble and its properties have been proposed. For example, the Barabási-Albert random graph model was introduced as a model that generated graphs with power law degree distributions [4]. Its generation process and employs a rich-get-richer phenomenon where new nodes connect to

existing nodes proportionally to their existing degrees. Although generative models are convenient, they are usually not flexible - even simple additional constraints may break down the heuristic and we must consider adaptation or replacement of the generative techniques.

Rejection sampling Rejection sampling is a common technique to sample from conditional distribution. The basic form of rejection sampling takes many samples from the distribution and rejects those that do not satisfy a required condition. This same approach can be used to generate networks that display a desired property by simply rejecting graphs that do not. While appropriate in some cases, there are many situations where this method is too slow as the space of interest is very small compared to the vast space the process samples from. For example, rejection sampling of simple graphs from the configuration model may take exponential time in the size of the graph for some degree sequences [48]. A common solution to high rejection rates is the use of Markov Chain methods to improve sampling efficiency.

Markov chain traversals Using Markov chains to sample from distributions, particularly where other methods like naive rejection sampling fail, is quite common. Markov chain traversals make small changes to the graphs (e.g., change one link) to step between graphs in the ensemble. (See Section 2.3 for more details on this process.) The use of these methods on graphs came to prominence in work by Broder [15]. Jerrum and Sinclair [72] subsequently showed when these chains rapidly mix under certain conditions. This type of sampling is widely used in developing algorithms for sampling perfect matchings of graphs (each graph has exactly one edge) and graph colourings (each node is assigned a colour that is different to its neighbours) [70, 71]. Similar methods have been used to sample from the configuration model [100], bipartite graphs as well as the connected configuration model. Canonical path methods are often used to determine upper bounds on the mixing time of these chains [32, 33], and both sequential algorithms [8] and heuristics [53, 139] have been used to speed up mixing of these chains.

2.2 Dynamics on Networks

Exploring dynamics over networks is an active field of research; at its heart is the inherent link between network structure and both the possible and observed dynamics. Network structure can dramatically impact dynamics, while the desired or occurring dynamics can sometimes impact the network structure.

Epidemics, for instance, are the transmission of a contagion over the links in the contact network of a society. There are countless publications investigating the impact of the network structure on the size and spread of epidemic models. Conversely, recently we have witnessed the presence of the COVID-19 epidemic leading to society-wide attempts to change our underlying social network structures, at least for a short period of time, to reduce the spread of the virus [147].

The spreading of information across social networks is another example of dynamics that are linked to the network structure. Many networks facilitate dynamics of one form of another. Cascading failures in electricity networks, the transmission of packets over the Internet and activity in brain and cognition networks are all areas of research in the field of dynamics on networks.

2.2.1 Information cascades

Information cascades are the movement of information through social networks and are one example of dynamics that can be observed over complex networks. Information cascades occur when people adopt ideas through the observation of others, and so this information spreads, or *cascades*, through networks. Each person participating in the cascade becomes *active* and can cause a chain reaction that causes movement of ideas through the network.

Information cascades and the associated adoption of new ideas has been studied from many angles. Psychologists and social scientists endeavour to understand the brain and individual level behaviours of adopting opinions. Mathematicians model these processes and collect and analyse data to understand these high level complex systems. A seminal work is from communications literature [118] studying the diffusion of innovations on business and these ideas permeate through many fields including sociology, computer science, physics *etc.*

Cascades are often associated with the spread of ideas, pictures and products on social media platforms due to the ease with which information can spread through such a network, such as *viral* memes. Cascades can be observed both online and offline: for example, when social movements start without significant promotion or organisation or in the stock market when there are significant fluctuations without a significant new piece of information.

The literature has proposed many different cascade models. There are generally two approaches: feature based approaches and model based approaches.

Feature based approaches Feature based machine learning approaches use indicative metrics to model and predict information propagation. They identify potentially relevant features of cascades to determine if a cascade will go viral or to predict the size of the cascade. Example features include content, network statistics, temporal metrics and community structure [61,148]. Many of these machine learning techniques perform well in predicting certain features of information spread. However, the methods are computationally expensive due to the feature engineering and training requirements. They are also deterministic and so are not useful in providing the probabilistic models that we will see are required for Bayesian modelling.

An alternate method is to use one of several mathematical models to explain the process of information diffusion.

Diffusion models Models of information diffusion have largely stemmed from the social influence literature and are used to model the cascading process, often with a stochastic process. They build on the assumption that when making a decision individuals often use the ideas and opinions of people around them. Diffusion models can be used at the population level to explain the process by which an innovation is communicated through certain channels among members of a social network [118] and was used in early models to explain the adoption of innovation. Agent based segregation models and coordination models have also been explored for modelling contagion in networks [96, 123, 124]. Epidemiological models have also been used in compartmental models of social influence as it is often proposed that transmission of an idea is similar to transmission of a virus or bacteria. These are often stochastic models where individuals transition between states, e.g., susceptible and infected, with some probability. In these models, individuals are in states according to whether they have engaged in the activity, are influencing others or are susceptible to influence [128].

Threshold models Another common model family for decision making are threshold models. Threshold models posit that individuals consider participating in a cascade dependending on the fraction or number of others already engaged [137]. Often individuals will not participate in a fad or innovation unless a certain proportion of their friends participate. Threshold models range in complexity and are easily applied to networks. Granovetter [56] proposed one of the first threshold models to analyse collective action in a binary decision making process. He assumes that people have a threshold based on their own personality and views, and if the proportion of people engaging in an activity is above this they will join. Watts [141] extended this model to include network structure and assume that

individuals can only be influenced by their friends. It is difficult in these models to determine the thresholds of individuals and they often require detailed network data. There are many extensions that generalise these models and require looser assumptions [73, 110].

The Independent Cascade model The Independent Cascade Model introduced by Kempe [74] is a seminal model in the influence maximisation space and has since been used as a basic cascade model in many different areas of network science. Its conceptual simplicity has made this model applicable in many areas including viral marketing modelling and social influence prediction. The NP-hard problem of using the model to determine the most influential users in a network is still an active area of research. The model begins with an active set of nodes. When a node u becomes active, say in step t, it has one chance to activate each of its currently inactive neighbours w. It succeeds with parameter p_{uw}. Often $p_{uw} = p$ is a constant over the network but inferring non-homogeneous parameters is also an interesting problem [54]. If u succeeds then w becomes active and in the next step will attempt to activate its own inactive neighbours. Each active node will only attempt to activate each neighbour once. The process terminates when there are no more activations possible.

Developing models for information diffusion on networks is important in understanding how the network structure impacts and interacts with the diffusion [57]. We will use the independent cascade model in Chapter 5 and Chapter 6 to infer network structure from observed cascades.

2.3 Bayesian Inference

Bayesian inference was pioneered by Thomas Bayes along with Pierre-Simon Laplace. In a Bayesian framework model parameters are random variables about which we can make probability statements. Probabilities describe the plausibility of some proposition[1] [11]. Since its inception, Bayesian inference has grown into a vast subject area that has been useful in many applied sciences from medicine to sports science [121].

We are interested in inferring the distribution of a, sometimes multivariate, parameter θ given some condition D. Often D is observed data. Before data is collected,

[1]This is in contrast to the frequentist approach pioneered by E. Pearson, J. Neyman and R. Fisher in which probabilities are limiting relative frequencies and parameters are fixed unknown constants.

there is some *a priori* knowledge of the system that is encompassed within a *prior distribution* of our parameter/s, $P(\theta)$. The *posterior distribution*, $P(\theta|D)$, the distribution after data has been observed, is then found using Bayes' rule:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}, \tag{2.5}$$

where θ is the parameter of interest. $P(D|\theta)$ is the likelihood model for the process we are observing. $P(D)$ is a normalising factor, often called the *evidence*. Specifically,

$$P(D) = \int_{\theta} P(D|\theta)P(\theta)d\theta. \tag{2.6}$$

In typical inference problems, this denominator is ignored as it is not a function of θ, and so posterior inference is based on

$$P(\theta|D) \propto P(D|\theta)P(\theta). \tag{2.7}$$

However, in applications when model choice is of interest $P(D)$ should be considered. If all the terms in Equation 2.5 are known then exact evaluation of the posterior is possible. However, as $P(D)$ is an integral over the, possibly high dimensional, parameter, it is often hard to calculate and is only available in a closed form when the prior distribution is a conjugate of the posterior. Therefore, sampling methods are used to approximate the posterior distributions by simulation. We will introduce these methods with the application towards Bayesian inference in mind; however they can also be applied to a wider set of sampling problems.

Monte Carlo Methods Monte Carlo methods are an extremely powerful tool that replaces theory by experiment to give approximations to the quantities of interest. At the heart, Monte Carlo simulations use random numbers to directly simulate the process or calculation of interest [63]. The classic example is taking random samples from a probability distribution to calculate an intractable integral, e.g., the expectation, with respect to the probability measure. While early Monte Carlo methods were computed by hand [83], these methods emerged as viable tools with the development of computers as they enabled the large number of realisations required for reliable approximations.

Markov Chain Monte Carlo While basic Monte Carlo methods are extremely powerful, problems arise when producing the simulations required to approximate the distribution of interest becomes impossible or intractable. To overcome this

common problem, Markov Chain Monte Carlo (MCMC) methods use a Markov chain to take samples from the stationary distribution that matches the distributions of interest.

MCMC techniques are often used in Bayesian inference as the posterior distributions are usually difficult to determine analytically. Let the distribution of interest be π. MCMC methods create a *Markov chain* that has a stationary distribution π equivalent to the distribution of interest. Rather than sampling from π directly, for example using a simple Monte Carlo simulation or an accept-reject method, MCMC methods explore the space iteratively, taking more and more samples until an overall picture can be obtained. The Markov chain $\mathcal{X} = \{X^{(0)}, X^{(1)}, \dots X^{(k)}\}$, with stationary distribution π, can be considered as dependent samples from π.

Pairing Monte Carlo simulation with Markov chain theory has been around for almost as long as Monte Carlo methods themselves. Despite this, only recently have MCMC methods been effectively used across many applied sciences. This delay is largely attributed to the computational power available; recent advances in computation has made taking a huge number of samples a reasonable task. These methods are used to sample from a distribution of interest that may otherwise be difficult or impossible to sample from. Robert and Casella, dominant figures in the MCMC space, present a succinct yet thorough history of MCMC as a chapter in the book 'Handbook of Markov Chain Monte Carlo' [16, 116].

There are a few requirements on the construction of this chain to ensure we sample from our distribution of interest, namely detailed balance and ergodicity. Full definitions and proofs are available in Robert and Casella [16]. There are many different, often application dependent, ways to construct and take samples from this Markov Chain. One of the main challenges in using MCMC methods is setting up a Markov chain that converges rapidly to and efficiently explores the space. We will explore two common sampling techniques: Metropolis Hastings and Hamiltonian Monte Carlo.

2.3.1 Metropolis-Hastings

The Metropolis-Hastings (MH) method is a well known MCMC method, thanks to its simplicity and wide ranging applicability [64, 92]. The MH sampler takes steps around the space according to some proposal distribution — often a random walk — and at each step decides whether to accept the next step or remain in the current state.

Consider sampling from the target distribution $\pi(\theta)$. We use the MH algorithm

to create a Markov chain $\theta^{(1)}, \theta^{(2)}, \cdots$. To do so, we choose a proposal distribution $Q(\theta'|\theta)$ to propose the next candidate θ' from the current state θ. The proposal distribution must be able to explore the entire space in a finite number of steps [117].

The candidate parameter θ' is accepted using a Metropolis update probability given by

$$\alpha = \min\left(1, \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}\right). \tag{2.8}$$

Thus, if the proposal distribution is symmetric then

$$\alpha = \min\left(1, \frac{\pi(\theta')}{\pi(\theta)}\right). \tag{2.9}$$

The chain is generated from the proposed parameter θ' as follows

$$\theta^{(t+1)} = \begin{cases} \theta', & \text{if accepted,} \\ \theta^{(t)}, & \text{otherwise,} \end{cases} \tag{2.10}$$

where θ' is generated from $Q(\theta'|\theta^{(t)})$. The chain will always accept steps into regions of higher posterior density, as well as sometimes moving into areas of lower density in order to explore the space. This chain will converge to the stationary distribution π if it is *irreducible*. That is, if Q has support over the entire state space of π. This is a relatively light constraint on Q for theoretical convergence of the chain. However, in practice choosing an appropriate Q is not a trivial task when aiming to achieve efficient exploration.

Random Walk Metropolis Hastings Arguably the most intuitive MH algorithm is random walk MH where a random perturbation is proposed at each step. That is, $Q(\theta'|\theta^{(t)}) = \theta^{(t)} + \epsilon_t$, where ϵ_t is some small perturbation, usually drawn from a uniform or Gaussian distribution. While the acceptance ratio is independent of variance of ϵ_t, the value of θ' can be highly dependent on the step size. Proposals that attempt to explore the space by taking large steps are likely to have a high rejection rate. Conversely, small steps will take many iterations to explore the entire space.

Gibbs' sampling Gibbs' sampling is a special case of MH that utilises knowledge of the distribution π to ensure $\alpha = 1$ in all transitions. It is often useful in cases when π is multi-dimensional; however requires knowledge of the conditional distributions of π. When sampling from the joint distribution $\pi(x, y)$, we alternately take samples $x' \sim f_{X|Y}(x|y^t)$ and $y' \sim f_{Y|X}(y|x')$, where $f_{X|Y}$ and $f_{Y|X}$ are the conditional distributions.

Set $\theta^{(0)}$
for $t = 1...K$ **do**
  Generate $\theta' \sim Q(\theta'|\theta^{(t-1)})$
  Take $\theta^{(t)} = \begin{cases} \theta', & \text{with probability } \alpha \\ \theta^{(t-1)}, & \text{with probability } 1-\alpha. \end{cases}$
    where $\alpha = \min\left(1, \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}\right)$
end for

Algorithm 2.1: General Metropolis-Hastings algorithm [117].

There are of course many other variants and extensions of Metropolis-Hastings. Component-wise MH addresses sampling from higher dimensional posteriors where the update of every parameter at once is impractical and has extremely low acceptance rates. Instead, one updates and moves each parameter (or block of parameters) independently. This improves mixing in higher dimensional spaces but highly correlated parameters can cause slow mixing. Multiple-try MH attempts to improve mixing of the chain by increasing step size and acceptance rate [85]. Reversible jump MH (and MCMC) allow for a varying number of dimensions in the posterior distribution [60]. Improvements and extensions of these methods continue to be developed.

2.3.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is an MCMC method that attempts to improve exploration of the space, acceptance rates of proposals and mixing of the Markov Chain. In the above algorithms local perturbations are made to the current state and this often leads to highlighly correlated samples as large steps are often rejected, and can become prohibitive in large state spaces. HMC uses the geometry of the space of interest in order to make large jumps. Originating in the physics discipline of lattice field theory and initially known as Hybrid Monte Carlo [39], HMC computes proposals by following contours of the space using Hamiltonian dynamics.

The standard and most intuitive introduction to HMC begins with the visualisation of a frictionless ball sliding over an undulating surface. This system is described by the position ($\mathbf{q}$) and momentum ($\boldsymbol{\rho}$) of the ball at any point in time. The ball has potential, $U(\mathbf{q})$, and kinetic energy, $K(\boldsymbol{\rho})$, and energy is converted between the two as the ball moves around the surface. The position vector of this ball is equivalent to the current state of our parameter θ. However, we continue with $\mathbf{q}$ as notation for the position vector in this section to be consistent with much of the

literature. The potential energy $U(\mathbf{q}) = 1 - \log(\pi(\mathbf{q}))$, where $\pi(\mathbf{q})$ is our posterior distribution of interest. Momentum parameters are introduced artificially to allow movement throughout the space. We combine the potential and kinetic energies to form the Hamiltonian

$$H(\mathbf{q}, \boldsymbol{\rho}) = U(\mathbf{q}) + K(\boldsymbol{\rho}). \tag{2.11}$$

Hamiltonian dynamics dictate how $\boldsymbol{\rho}$ and $\mathbf{q}$ change. Specifically,

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial \rho_i}, \tag{2.12}$$

$$\frac{d\rho_i}{dt} = \frac{\partial H}{\partial q_i}. \tag{2.13}$$

Which simplify using Equation 2.11,

$$\frac{dq_i}{dt} = \frac{\partial K}{\partial \rho_i}, \tag{2.14}$$

$$\frac{d\rho_i}{dt} = \frac{\partial U}{\partial q_i}. \tag{2.15}$$

In practice, the momentum vector $\boldsymbol{\rho}$ is a sample from a multivariate normal distribution, $\mathcal{N}(0, \Sigma)$. At each step the ball is given some momentum, sampled from $\mathcal{N}(0, \Sigma)$ and moves around the surface according to these equations.

To use Hamiltonian dynamics to sample from our distribution of interest, we first translate the density of this distribution to a potential energy with position variables and auxiliary momentum variables. The energy function is given by

$$P(\mathbf{q}, \boldsymbol{\rho}) = \frac{1}{Z} e^{-H(\mathbf{q}, \boldsymbol{\rho})}, \tag{2.16}$$

where Z is a normalising constant. Here, $\mathbf{q}$ is our variable of interest and $\boldsymbol{\rho}$ are auxiliary variables that enable us to use Hamiltonian dynamics. The joint density $P(\mathbf{q}, \boldsymbol{\rho})$ has a marginal distribution $P(\mathbf{q}) = \frac{1}{Z'} e^{-U(\mathbf{q})}$, where Z' is a normalising constant. Recall, that $U(\mathbf{q})$ is a function of our distribution of interest, so sampling from the augmented density in Equation 2.16 will provide samples of $\mathbf{q}$ that follow our target density.

To move throughout the space we propose changes to the position and momentum variables, traverse the resulting Hamiltonian trajectory and accept each step using a Metropolis update. The resulting Markov Chain samples from our distribution of interest [97, 116]. As it is impossible to solve Hamitonian's equations for anything beyond the most simple systems, a numerical implementation of these mechanics

is required. The most widely used discrete integrator is the Leapfrog integrator, a close relative of the modified Euler method [17]. The integrator has step size ϵ and alternates between half-step updates for momentum $\boldsymbol{\rho}$ and full-step updates for the position $\mathbf{q}$. To move from current iteration t to the next iteration $t+1$,

$$\begin{aligned} \boldsymbol{\rho}_{t+1/2} &= \boldsymbol{\rho}_t - \frac{\epsilon}{2}\frac{\partial U}{\partial \mathbf{q}_t} && (2.17)\\ \mathbf{q}_{t+1} &= \mathbf{q}_t + \epsilon\Sigma^{-1}\boldsymbol{\rho}_{t+1/2} && (2.18)\\ \boldsymbol{\rho}_{t+1} &= \boldsymbol{\rho}_{t+1/2} - \frac{\epsilon}{2}\frac{\partial U}{\partial \mathbf{q}_{t+1}}. && (2.19) \end{aligned}$$

Applying L leapfrog steps a total of $T = \epsilon L$ time is simulated, and the final $(\mathbf{q}_T, \boldsymbol{\rho}_T)$ is proposed as the next step in the Markov Chain. Tuning is often required to choose ϵ, L and Σ^{-1}, and often preliminary runs and the resulting trace plots are used to determine these values. See again the Handbook of Markov Chain Monte Carlo [97] and PyStan[2] documentation [129] for more details on tuning.

To summarise, HMC proceeds in two steps:

Step 1: Sample new values $\boldsymbol{\rho}$ for the momentum variables from the multivariate normal distribution.

Step 2: A discretised Hamiltonian trajectory of length L and step size ϵ is performed on state $(\mathbf{q}, \boldsymbol{\rho})$ to propose $(\mathbf{q}_T, \boldsymbol{\rho}_T)$. A Metropolis update is used to accept or reject the proposal.

$$\begin{aligned} \alpha &= \min\left(1, \frac{P(\mathbf{q}_T, \boldsymbol{\rho}_T)}{P(\mathbf{q}, \boldsymbol{\rho})}\right), && (2.20)\\ &= \min\left(1, \frac{e^{-U(\mathbf{q_T})-K(\boldsymbol{\rho}_T)}}{e^{-U(\mathbf{q})-K(\boldsymbol{\rho})}}\right). && (2.21) \end{aligned}$$

The momentum vector, $\boldsymbol{\rho}_T$, is then discarded, and these two steps repeated until a satisfactory number of samples have been taken.

The trajectories follow level curves of the Hamiltonian and result in multi-dimensional proposals. As the discretisation of the trajectory results in only approximate conservation of H over the trajectory, the MH update is required. Discarding the momentum parameters after each step allows a new level curve to be explored at each step, improving mixing, and is necessary for ergodicity of the chain. Note that a negation of the final leapfrog step is required to ensure reversibility and detailed balance at the end of the trajectory, but as $K(\rho) = K(-\rho)$ this does not impact the proposal in practice.

[2]PyStan is a Python implementation of HMC.

The major advantage of HMC is the ability to propose high probability proposals that are far from our current point, which reduces correlation between samples and improves mixing. In contrast to MH, it can pass through regions of lower density more easily and escape local optima. However, it can only be implemented on continuous random variables. To implement HMC for general problems the Stan package [129] provides a robust and flexible interface for sampling continuous random variables. In Chapter 7, we use the PyStan implementation in Python to implement HMC and the required optimisations for tuning parameters.

2.3.3 Sequential Monte Carlo

Sequential Monte Carlo (SMC) is a Monte Carlo algorithm that uses the premise that it may be easier to approximately sample from a sequence of distributions that approach our distribution of interest, rather than the distribution itself. It is not an MCMC algorithm; however it is complementary as MCMC methods are often used within the SMC set-up. SMC uses importance sampling at each step to eventually take samples from our distribution of interest.

Importance Sampling Importance sampling is a generalisation of classic Monte Carlo estimation and can be used when we do not know how to sample from the probability distribution of interest, but can evaluate the probability of given samples. This is often the case in Bayesian inference. Instead of sampling directly from our distribution $P(\theta)$, we consider an *importance distribution*, $g(\theta)$. Independent samples (or particles) $\theta_i \sim g(\theta)$ for $i = 1, \ldots, S$ are drawn from $g(\theta)$. These samples are then reweighted so they reflect our target distribution rather than the importance distribution via the reweighting formula

$$w(\theta_i) = \frac{\pi(\theta_i)}{g(\theta_i)}. \tag{2.22}$$

After normalisation, the result is a weighted sample from the distribution $\pi(\theta)$ which can be used to estimate our probability distribution (or integral) of interest. Care should be taken to ensure that the importance distribution has at least the same support as the distribution of interest, and that the tails of the importance distribution are thicker than the distribution of interest to avoid infinite variance. This makes importance sampling difficult to apply in general as, typically, very little is known about the posterior distribution. Sequential Monte Carlo attempts to overcome this drawback by sequentially building the importance distributions.

Sequential Monte Carlo Chopin [27] presented the seminal work on SMC to use a sequence of target distributions where the final target is our distribution of interest, and Del Moral [38] provided a more general framework. We have some set of distributions $P_t(\theta)$ with $t = (0, 1, \dots T)$. This sequence approaches the posterior distribution. Often P_0 is easy to sample from while P_T is the distribution of interest. In Bayesian inference P_0 is the prior distribution and P_T is the posterior. At each step the previous distribution is used as the importance distribution at the current step, allowing for a smooth transition between our prior and posterior. We begin by generating S equally weighted random particles and reweight these at each step to ensure that the empirical distribution of these samples converge to P_t as $S \to \infty$ for each t. As the particles traverse through the sequence they will tend to become skewed or degenerate. Therefore, resampling, and a subsequent 'move' step, is required to repopulate our samples. Let θ_t^i be the particles representing P_t and w_t^i represent the corresponding particle weights. A general algorithm is as follows:

Step 0: Draw θ_0^i for i in $1, \dots, S$ from $P_0(\theta)$ and set $w_0^i = 1/S$

Step 1: Reweight particles

$$w_t^i = w_{t-1}^i \times \frac{P_t(\theta_{t-1}^i)}{P_{t-1}(\theta_{t-1}^i)} \tag{2.23}$$

Step 2: Normalise the weights

$$W_t^i = \frac{w_t^i}{\sum_j w_t^j} \tag{2.24}$$

Step 3: Resample and move: Resample the particles according to their weight, and use MCMC to move the particles. Set $w_t^i = 1/S$ for each particle.

There are a variety of ways to resample and move the particles at each step. Often, the resampling step is only employed if the 'effective sample size' (ESS) reduces below a certain threshold. The ESS determines the number of perfect samples that the weighted samples are equivalent to prevent the sample being dominated by a few large weights. Resampling tends to duplicate particles with larger weight and remove those of small weight, focussing on the 'important' regions of the space and diversifying the samples.

The design of the annealing schedule $P_0(\theta), \dots P_T(\theta)$ is dependent on the application, but usually falls into two main categories: *likelihood annealed* SMC and *data annealed* SMC. In likelihood annealed SMC the sequence of distributions are designed to transition from the easy to sample P_0, to the distribution of interest P_T.

Often this is achieved by setting $P_t(\theta) = P_T(\theta)^\gamma$ and the temperature parameter γ is varied from 0 to 1 for a smooth transition. Data annealed SMC introduces the data in batches by using $P_t(\theta|D_{1:t})$, where $D_{1:t}$ is the data up to batch t. This method is often used when data is observed in a streaming fashion or if it is difficult to find an initial sample with non zero mass from the parameter space of interest $P(\theta|D)$.

Here we have introduced some background in two large fields of study: network analysis and Bayesian algorithms to aid the reader in understanding the rest of this thesis. In each chapter we provide more details as required as well as references for further reading.

Chapter 3

Generating Connected Networks

This chapter contains contents from the publication:

> Gray, C., Mitchell, L., and Roughan, M. Generating connected random graphs. *Journal of Complex Networks*, 7, 6 (2019), 896-912.

Please see the appendix for full authorship statements.

Random graphs are commonly used as underlying models in many fields, such as computer networking, biology, social sciences and physics [7, 35, 57, 104, 105]. The ability to generate random graphs with desired properties is crucial, as they may be used in conjunction with complex models, for instance a routing protocol in computer networking [143]. However, it is often the case that random graph theory jumps ahead of synthesis. There are many works on analysis of random graphs and their ensembles, but relatively few on how to generate them efficiently with all the desired properties. For example, the Waxman network [143] is a well known spatial network with theoretically well understood properties; however, only recently has there been advances in ways to find computationally efficient ways to generate large graphs for use in real world applications and studies [119]. Further, there is very little work on generating graphs that impose further restrictions to these models to match the networks of interest.

Real-world networks come with countless properties that one may consider modelling. For example, networks with specific global properties such as average number of edges, global clustering levels *etc*, or node level properties such as a specific degree distribution. Most random graph models focus on modelling one of these properties and then analysing other properties of the resulting ensembles that may

or may not match an observed network. Many current methods for generating random graphs result in networks with some undesirable properties for a particular application. For instance,

- the graphs may not be connected, e.g., the Gilbert-Erdős-Rényi model or spatial Waxman graph [143]; or
- the graphs may not be *simple*, i.e., they might have multi-edges or self-loops, e.g., the configuration model.

While one might argue that this is a modelling problem, there are nevertheless many instances in the literature where a model matches enough properties of the real networks in question that it is useful, except for one deficiency such as noted above.

Examples include:

- using the Waxman graph to model physical networks that are inherently connected, e.g., router networks; and
- using the configuration model that generates graphs with self-loops and multi-edges to model simple networks.

There have been many attempts to sample uniformly from graph models [32, 71]. While this may be useful when requiring only a graph with the desired property, the natural question remains of how to sample graphs while ensuring we maintain the conditional ensemble of the underlying graph model. This is essential in many applications; for example, when estimating parameters, or in applications of Approximate Bayesian Computation where the ensemble encompasses prior knowledge of the system.

In this chapter we present an algorithm to produce random graphs from a known ensemble $P(G)$ conditioned on an extra desired property of the network. Our algorithm uses MCMC methods to sample from the ensemble of interest. We focus here on generating connected networks, however the method generalises to conditioning on other properties. We show the algorithm samples graphs from the desired distribution and demonstrate the algorithm on spatially embedded random networks (SERNs), in particular the Waxman random graph [143]. We analyse the complexity to show that the algorithm is $\mathcal{O}(K)$ for K iterations on sparse graphs. Convergence of MCMC methods is particularly important, especially as one may expect the dimensionality to be a source of difficulties. Although we cannot tightly bound the number of iterations required for convergence theoretically, we introduce empirical results to show evidence for convergence scaling like $\mathcal{O}(n^2)$ in the number of nodes in the graph.

The algorithm we present not only has practical applications in that one can generate graphs with desired properties for use in various applications, but also, such a simulation algorithm could be used to estimate the probability of such graphs in an ensemble.

3.1 Mathematical overview

We introduced the mathematical formalities of networks Section 2.1, and recap here for convenience. A graph $G = (N, E)$, consists of a set N containing n nodes with edge set $E \subset N \times N$ of size e.

We are primarily concerned here with undirected graphs, i.e., $(i, j) \in E$ then $(j, i) \in E$, and simple graphs with no self edges, i.e., edge (i, i) does not exist. However, the algorithm presented here can generalise to directed random graphs and graphs with self edges. Also, the algorithm can be used to generate simple graphs from ensembles that do not necessarily produce them.

We say that two nodes are 'connected' if a path (a sequence of edges) exists between the two nodes. The graph is connected if all pairs of nodes (i, j) are connected.

We will demonstrate our method on spatially embedded random networks, specifically the Waxman ensemble. Recall, we create a SERN by placing n nodes uniformly at random within some defined region R of a metric space Ω with distance metric $d(x, y)$. Each node n_i has coordinate vector $\mathbf{x_i}$ and each pair of nodes is made adjacent independently, with probability p_{ij}, which is a function of the distance d_{ij} between the points. In the Waxman case,

$$p_{ij} = qe^{-sd_{ij}}, \tag{3.1}$$

for graph parameters s and q and distance between the two nodes $d_{ij} = d(\mathbf{x_i}, \mathbf{x_j})$.

Recall that the Metropolis-Hastings (MH) algorithm [64,92], draws samples from a distribution of interest. We will use this technique to sample from the distribution of networks with our desired property. Consider the target distribution $\pi(\theta)$ we wish to sample from. We use the MH algorithm with proposal distribution $Q(\theta'|\theta)$ to move through the space. The proposed parameter value θ' is accepted with probability

$$\alpha = \min\left(1, \frac{\pi(\theta')Q(\theta|\theta')}{\pi(\theta)Q(\theta'|\theta)}\right). \tag{3.2}$$

The chain is generated from the proposed parameter θ' as follows:

$$\theta^{(t+1)} = \begin{cases} \theta', & \text{if accepted,} \\ \theta^{(t)}, & \text{otherwise,} \end{cases} \tag{3.3}$$

where θ' is generated from $Q(\theta'|\theta^{(t)})$.

3.1.1 Related work

Markov chain traversals of graphs have been used to sample from a variety of graph spaces [48]. Exponential random graphs use MCMC based sampling methods to sample parameters for ensembles of networks with very specific properties to match real world networks [87]. There has been much focus on using MCMC to sample networks that have a desired degree sequence [2, 32, 131]. This is often achieved through the use of an 'edge swaps' proposal distribution that preserves the degree sequence of the network throughout the MCMC process. Much of this work focuses on uniform sampling of the configuration model; that is, networks with a given degree sequence. These have applications when using the configuration model directly or as null models [2]. Other works sample uniformly from graphs with power-law distributions in a similar manner [53]. Uniform sampling can be useful in some situations; however, we are often interested in sampling from a model with a more complicated underlying distribution, and in ensuring we do not oversample rare graphs. Therefore, here we focus on sampling from spaces of graphs that have a non-uniform distribution.

Recently, the 'edge-switch' proposal in MCMC methods have been used to sample bipartite graphs with only expected degrees that provide a framework to study partially observed networks [115], and the extension of the double swap to a triple swap to allow sampling of 'loopy' graphs [102]. Another related work, [146], uses link switches to generate synthetic networks preserving properties of a real graph input with privacy and significance testing applications.

3.2 Connectedness

We present our algorithm in the context of generating connected random networks. The property of connectedness is often observed in physical networks, such as a telecommunications network, where there is the requirement that a path exists between all nodes. Other physical examples include the Internet routing network. It is also important in the application of social networks. In general each individual

may not be connected to all others through some path. However, in the application of epidemics and information diffusion there is particular interest in the network over which information propagates. To participate in a cascade the individual must have come into contact with the contagion; therefore, there is necessarily a path between all individuals in the network over which the cascade is observed. We will study this problem in more detail in Chapter 5.

Many random graph generators do not consider connectivity and simply take the giant component of the resulting graphs or prove properties like the distribution of connected component size in the asymptotic limit. However, in many applications we are interested in generating connected networks of fixed size from our distribution.

Rejection sampling is commonly used to generate networks that display a desired property; however this is often inefficient. For connectedness, the probability of all nodes being connected can be very low even for quite reasonable parameter values, and so rejection sampling is often not practical. While the probability of connectedness has not been found analytically for Waxman graphs, simple simulations can show that connected graphs are often unlikely. Figure 3.1 shows the proportion of Waxman graphs that are connected after 200 samples, for a variety of parameters. As expected, we can see that as the dependence on distance becomes stronger (s parameter increasing) the probability of connectedness decreases. Additionally, the traditional $\mathcal{O}(n^2)$ sampler makes running even a few hundred samples of the Waxman graph expensive.

Markov chain methods have been used to produce connected random networks with a prescribed degree sequence [48, 139], with a particular focus on networks in peer-to-peer applications [32]. Generating connected graphs with a given degree sequence has been discussed at length in the literature using Markov chain Monte Carlo (MCMC) methods [102, 115, 139, 146]. In contrast to the current work, the existing MCMC algorithms use 'edge swaps' to give a uniform sample over the graph space.

3.3 Generating connected random networks

We assume a random graph model that generates an ensemble of sometimes unconnected graphs, and that the model provides a probability distribution across the ensemble, i.e., the probability $P(G)$ for each graph G. Even if we assume that this probability is calculable, direct simulation from the distribution is usually intractable due to the size of the ensemble. Usually, however, there is an algorithm

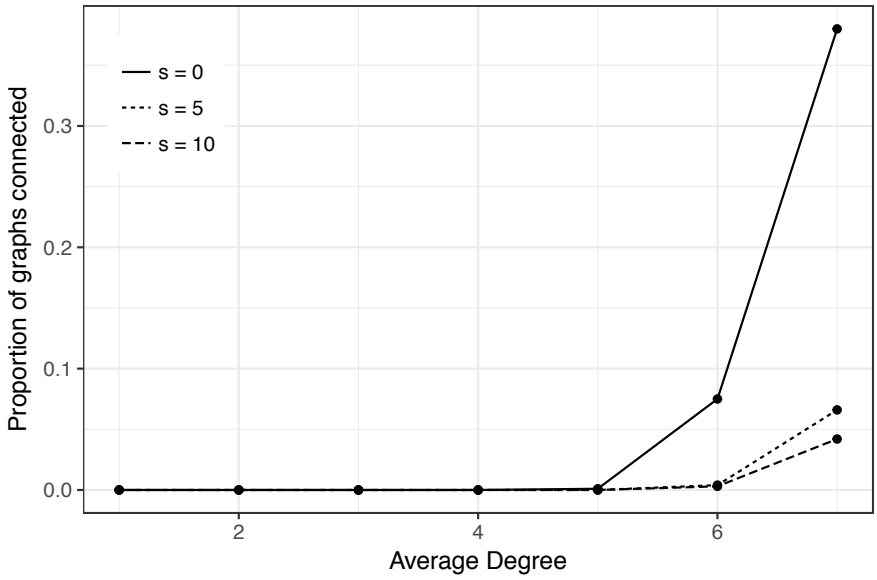


Figure 3.1: Proportion of connected networks in 1000 samples of a Waxman network with $n = 1000$.

to generate graphs from the ensemble.

Given the model, we would like to generate connected graphs with the same conditional probability distribution as the model of interest, i.e., we would like to generate connected graphs G with probabilities

$$P\{G|G \text{ is connected}\} = \frac{P\{G \text{ and } G \text{ is connected}\}}{P\{G \text{ is connected}\}}, \tag{3.4}$$

where the numerator is given by:

$$P\{G \text{ and } G \text{ is connected}\} = \begin{cases} P(G), & \text{for } G \text{ connected} \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

However, as noted we do not know $P\{G|G \text{ is connected}\}$ and we can not simulate directly from this ensemble. This leads naturally to the use of well known MCMC methods as the basis for the sampling algorithm.

We implement the Metropolis-Hastings method to generate a Markov chain that will result in samples from the desired distribution; this algorithm is given in Algorithm 3.1. The algorithm produces a new graph $G' = (N, E')$ based on the old graph G. The two main components are a symmetric proposal distribution that can explore the entire space and a tractable acceptance ratio.

1: Generate $G^{(-1)}$ from the model
2: Connect $G^{(-1)}$ to get $G^{(0)}$
3: **for** k=1..K **do**
4: Generate a random node pair (i, j)
5: **if** $(i, j) \in E$ **then**
6: Remove the edge: $E' = E \setminus \{(i, j)\}$
7: **if** G' is connected **then**
8: accept G' with probability $P(G')/P(G)$
9: **else**
10: reject G'
11: **end if**
12: **else**
13: Add edge: $E' = E \cup \{(i, j)\}$
14: accept G' with probability $P(G')/P(G)$
15: **end if**
16: **end for**

Algorithm 3.1: Metropolis-Hastings method for generating connected graphs.

We initialise the algorithm using the underlying model to create a random graph, $G^{(-1)}$, with $P(G^{(-1)}) > 0$. This graph can be chosen arbitrarily, but we expect faster convergence from graphs that are in regions of higher density in the posterior distribution, and so we choose this initial graph using the standard algorithm for the class of graphs of interest. Often, this graph will be unconnected; therefore, we connect the network by adding arbitrary links. The graph need not be necessarily chosen with the correct probability, so in this case almost any procedure to obtain connectivity is adequate. The result is a connected random graph $G^{(0)}$ to be used as the input to the MH algorithm.

The process described in detail below.

Step 1 – Proposal: The probability density $Q(G'|G)$, is the proposal distribution that gives the next candidate for the algorithm. An advantageous feature of the Q for the MH algorithm is that it is symmetric, i.e., $Q(G'|G) = Q(G|G')$, as this simplifies the acceptance ratio.

Here we perform the algorithm link by link. At each step, we select a node pair (i, j) at random, and consider adding or removing a link to obtain the new network. In practice we choose two distinct nodes uniformly at random and consider the possible link between them. Mathematically,

1. if $(i, j) \in E$ then $E' = E \setminus \{(i, j)\}$, i.e., the edge is removed,

2. if $(i,j) \notin E$ then $E' = E \cup \{(i,j)\}$, i.e., the edge is added.

All node pairs are chosen with equal probability, so $Q(G'|G) = 1/(n(n-1))$ for all G and G' that differ by one link. Therefore, the transition is symmetric.

This proposal has been used in graph sampling previously, notably in applications related to sampling exponential random graphs, e.g., [87], and there is no consideration of connectivity in this proposal step.

Step 2 – Acceptance: The Metropolis-Hastings acceptance ratio (the probability of accepting the proposed transition) given that the proposal is symmetric is given by

$$\alpha = \min\left\{1, \frac{P\{G'|G' \text{ is connected}\}}{P\{G|G \text{ is connected}\}}\right\}. \tag{3.6}$$

To determine a tractable acceptance ratio, we consider the connectivity of each proposed graph. Recall, we start with a valid connected graph $G^{(0)}$. If G' is unconnected, then $P\{G'|G' \text{ is connected}\} = 0$, so unconnected graphs will never be accepted; therefore, we remain in the space of connected graphs.

We use this to establish a tractable ratio. When G and G' are connected, the conditionals can be dropped from the probabilities, as $P\{G \text{ is connected}\}$ is constant over the ensemble.

This gives

$$\alpha = \min\left\{1, \frac{P(G')}{P(G)}\right\}, \tag{3.7}$$

for connected graphs G and G'. The ratio is tractable in many cases where we can calculate the ratio of the probability distributions. If all edges are independent then this can be calculated very quickly.

The process is iterated a number of times until the Markov chain converges and the networks are being sampled from the stationary distribution of interest.

To implement this algorithm we must check the connectivity of the graph when removing a link. There are a variety of algorithms for checking connectivity [44]. We use a simple breadth first search with complexity $\mathcal{O}(n+e)$, as we are interested in sparse graphs with $e \sim \mathcal{O}(n)$, meaning the search is $\mathcal{O}(n)$. After removing a link (i,j) the graph remains connected if and only if a path still remains between i and j. Therefore, determining if the graph still has a path between i and j, although still $\mathcal{O}(n)$, is likely to be faster than the worst case, especially on spatial graphs.

This algorithm will work well on networks where each edge exists (conditionally) independently of any other, e.g., the ER graph, inhomogeneous random graphs [10],

or SERNs. In these cases the calculation of $P(G')/P(G)$ is a simple ratio of edge probabilities. In principle this algorithm can be applied to any model in which every graph has positive probability prior to adding that extra constraint, although $P(G')/P(G)$ may be hard to calculate. Note also that the algorithm, as described here, will work only for graph models that assign positive probability to graphs with a different number of edges. For example, the configuration model network has a fixed degree sequence, hence a fixed number of edges. Therefore, the proposal of adding or removing a single edge will be inappropriate as it will break the degree sequence. A simple change of proposal distribution allows for sampling from these networks [48, 139].

3.4 Algorithmic discussion

3.4.1 Theoretical convergence

Theorem 1 *Algorithm 3.1 generates samples from the random graph ensemble with probability distribution* $P\{G|G \text{ is connected}\}$.

Proof 1 *Theorem (7.4) of Robert & Casella [117] states that the chain produced by the Metropolis-Hastings algorithm (Algorithm 3.1) converges to the stationary distribution* π *if:*

1. *it is irreducible, and*
2. *it is aperiodic.*

Consider the Markov chain produced by Algorithm 3.1. We show there exists a sequence of a finite number of steps with positive probability from any connected graph H *to any other connected graph* H', *i.e.,* $P(H \to H') > 0$. *The Markov chain is irreducible if all states communicate; that is, if every state can be reached from any other in finite time. This requires the proposal to have support over the entire space in a finite number of steps. We must ensure that the graph remains connected in all steps. Therefore, consider adding all edges not in* H *to create a clique. Then remove the edges in subsequent steps to reach* H'.

$$P(H \to H_{\text{clique}} \to H') = P(H \to H_{\text{clique}})P(H_{\text{clique}} \to H'). \tag{3.8}$$

If every connected graph in the ensemble has non-zero probability, both terms on the RHS have positive probability. Therefore, the chain is ***irreducible****.*

A sufficient condition for the Markov chain to be aperiodic is to choose Q such that the probability of the event $\left\{X^{(t+1)} = X^{(t)}\right\}$ is non-zero for some state. If the removal of an edge destroys connectivity the transition is rejected and the chain remains in the current state. Therefore, the chain is ***aperiodic****.*

Note that the acceptance probability construction ensures $\pi = P\{G|G$ is connected$\}$. Hence, by Theorem (7.4) of Robert & Casella, Algorithm 3.1, with acceptance probability α (3.7) converges to the distribution of interest.

Unfortunately this result only assures us that after infinite time the process will be sampling from the distribution of interest. We show evidence for convergence in finite time in Section 3.6.

3.5 SERN example

Here we present the example of spatially embedded networks to demonstrate the algorithm.

Edges in a SERN are independent (conditional on distance), and hence the probability distribution of a spatially embedded random network is given by

$$P(G) = \prod_{(i,j)\in E} p_{ij} \prod_{(i,j)\notin E} (1 - p_{ij}), \tag{3.9}$$

where p_{ij} is the probability of an edge for the specific SERN of interest. For example, in the case of a Waxman network the edge (i, j) is given by

$$p_{ij} = qe^{-sd_{ij}}, \tag{3.10}$$

for nodes separated by distance d. In the Waxman formulation, d is calculated by the Euclidean distance.

Using (3.9) above, the acceptance probability when adding a link (i, j) becomes

$$\frac{P(G')}{P(G)} = \frac{p_{ij}}{1 - p_{ij}}, \tag{3.11}$$

and for removing a link is

$$\frac{P(G')}{P(G)} = \frac{1 - p_{ij}}{p_{ij}}. \tag{3.12}$$

While the probability distribution of the ensemble is known, it is often difficult in practice to determine the value of $P(G)$ explicitly. Here, we only require the ratio of the probabilities between each pair of graphs, a much easier calculation.

Often, we assume that we are dealing with sparse graphs. Dense graphs are more likely to be connected, and so would not require this algorithm. Additionally, in physical networks there exists a cost constraint of constructing links, resulting in many sparse real-world networks.

3.5.1 Complexity

The complexity of an algorithm determines how effectively it can be used on a wide range of problems.

Theorem 2 *Algorithm 3.1 with K iterations has computational complexity $\mathcal{O}(K)$, independent of the size of the graph, for sparse graphs.*

Proof 2 *We use a neighbourhood list stored in a hash map to describe the edges in the graph. This results in an expected $\mathcal{O}(1)$ operation to check edges for existence and add/remove edges at each iteration. We check for connectivity when edge removal is proposed. The breadth first search algorithm is $\mathcal{O}(n)$ for a sparse network with n nodes. For a sparse graph the number of edges is $\mathcal{O}(n)$, and so the probability of selecting an edge to delete is $\mathcal{O}(1/n)$. That is, for large n*

$$P\{\textit{edge } (i,j) \textit{ exists}\} \sim \frac{1}{n}. \tag{3.13}$$

So, the probability there exists an edge between the two chosen nodes, requiring the $\mathcal{O}(n)$ connectedness routine, decreases like $1/n$. Therefore, each iteration is on average $\mathcal{O}(1)$, and overall the algorithm is $\mathcal{O}(K)$ in the number of iterations.

3.5.2 Single link Markov chain: General Waxman

Theorem 1 guarantees convergence in infinite time; however, to be practical we would like it to mix in a reasonable number of steps. We would like to estimate the number of iterations K required to have evidence that the Markov chain has sufficiently converged to the stationary distribution.

In this section we introduce the mixing time of MCMC processes on graphs in the case of the general Waxman with no constraints (i.e., not necessarily connected). The mixing time will depend on the number of nodes in the graph, and we begin by demonstrating that the general Waxman has a mixing time of $\mathcal{O}(n^2)$ using this algorithm. We then introduce the connectivity constraint and show empirically that this extra constraint does not slow mixing in practice.

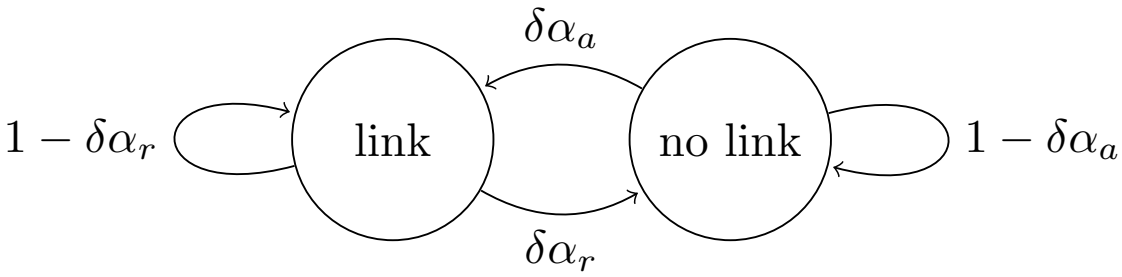


Figure 3.2: Single link in the Markov Chain.

Let us consider a general Waxman with independent links. Our method is performed by proposing a change to a single node pair in each step. Therefore, let us consider that we choose a node pair (i, j) in the graph G with probability δ. In this case we choose all node pairs with equal probability, i.e., $\delta = 1/(n(n-1))$. While we will analyse one node pair, by choosing a link in the graph with probability δ we are considering the graph as a whole.

Figure 3.2 shows the transition probabilities of one node pair. Note that the probability of remaining in the state is through two processes; either not choosing the node pair or choosing the link and not accepting the change. i.e., $1 - \delta + \delta(1-\alpha) = 1 - \delta\alpha$. For the node pair chosen the probability of accepting a change is

$$\alpha_{\mathrm{a}} = \min\left(1, \frac{p_{ij}}{1-p_{ij}}\right) \quad \text{if adding,}$$
$$\alpha_{\mathrm{r}} = \min\left(1, \frac{1-p_{ij}}{p_{ij}}\right) \quad \text{if removing.}$$

Combining the probability of choosing the node pair (i, j) and the transition probabilities, the transition matrix of node pair (i, j) is

$$P = \begin{matrix} \\ \text{link} \\ \text{no link} \end{matrix} \begin{matrix} \begin{matrix} \text{link} & \text{no link} \end{matrix} \\ \begin{pmatrix} 1-\delta\alpha_r & \delta\alpha_r \\ \delta\alpha_a & 1-\delta\alpha_a \end{pmatrix} \end{matrix} . \tag{3.14}$$

This converges to the stationary probability of a link between nodes i and j

$$\begin{aligned} P(\text{link}) &= \frac{\alpha_a}{\alpha_a + \alpha_r}, \\ &= \frac{\min\left(1, \frac{p_{ij}}{1-p_{ij}}\right)}{\min\left(1, \frac{p_{ij}}{1-p_{ij}}\right) + \min\left(1, \frac{1-p_{ij}}{p_{ij}}\right)}, \\ &= \begin{cases} \frac{\frac{p_{ij}}{1-p_{ij}}}{\frac{p_{ij}}{1-p_{ij}}+1}, & \text{if } p_{ij} < 0.5, \\ \frac{1}{1+\frac{1-p_{ij}}{p_{ij}}} & \text{if } p_{ij} \geq 0.5, \end{cases} \\ &= p_{ij}. \end{aligned}$$

Hence, the MCMC process will produce a Waxman network with the required link probability.

The mixing of the Markov chain is important in the application of the algorithm in finite time. The spectral gap controls the rate of exponential decay to equilibrium and the relaxation time gives an indication of how fast the chain converges. The two eigenvalues of the transition matrix (3.14) are $\lambda_1 = 1$ and $\lambda_2 = 1 - \delta\alpha_r - \delta\alpha_a$, giving a spectral gap of $\gamma^* = \delta(\alpha_r + \alpha_a)$. Note that we select edge (i, j) with probability $\delta \sim 1/N^2$ and $\alpha_r + \alpha_a$ is constant for any given link.

Thus, the relaxation time is given by,

$$\begin{aligned} t_{\text{rel}} &= \frac{1}{\gamma^*}, \\ &= \frac{1}{\delta(\alpha_r + \alpha_a)}, \end{aligned}$$

where

$$\begin{aligned} \alpha_r + \alpha_a &= \begin{cases} \frac{1}{1-p_{ij}}, & \text{if } p_{ij} < 0.5, \\ \frac{1}{p_{ij}} & \text{if } p_{ij} \geq 0.5, \end{cases} \\ &\in [1, 2]. \end{aligned}$$

Therefore, in general, $t_{\text{rel}} \sim n^2$ for the graph, and we expect that $K \sim \mathcal{O}(n^2)$ for the algorithm to converge.

Above we assume that node pair transitions are independent, as they would be in the standard Waxman graph. However, when we consider connectedness, the presence or absence of other edges may prevent a particular edge being removed.

This will increase the mixing time of the chain as it is possible that the most likely path from one graph to another travels through some unconnected graph. Nevertheless, the above analysis gives us a lower bound on, and an intuition about the mixing time of our algorithm.

Due to the complicated dependence structure that connectedness introduces between edges, the transition matrix does not simplify as above. A theoretical analysis on the mixing time of sampling from $P\{G|G \text{ is connected}\}$ requires a more complex handling of the graph of the underlying Markov chain. Canonical path methods and bounding Markov chain properties like 'conductance' have been used to investigate the mixing time of Markov chains on graphs in different applications, such as bipartite and regular graphs [32, 33, 47]. These methods can prove upper bounds on the mixing times of chains and show mixing is polynomial in the size of the network. Bounds using these methods are at present very loose, sometimes as high as $\mathcal{O}(n^{45})$ [33]. Experimental evidence is often used to provide support for faster mixing chains [33], and we use this approach below in Section 3.6.

3.6 Connected Waxman Graphs

Section 3.4.1 showed that Algorithm 3.1 will converge to the distribution of connected Waxman graphs. We expect approximate convergence in $K \sim \mathcal{O}(n^2)$ steps. The critical question becomes, how long is required in practice? We implement the algorithm described above using the NetworkX package in Python [62] to produce connected SERNs, and provide the code[1].

In order to simulate networks in finite time we must provide evidence for the convergence of the chain. Many applications of MCMC use visual means to determine when the chain seems to have converged. Here we use a heuristic that uses statistics of the graph.

Summary statistics are often used to describe network ensembles. Here we utilise the distributions of two summary statistics over the ensemble to determine convergence, the distribution of average degrees and average path length. When we condition on connectedness, we expect a slight increase in average degree to allow for connectedness. This results in a shift in the distribution of the average degree over the ensemble. Conversely, we expect the average path length to decrease as the starting graph $G^{(0)}$ will have longer links than a typical Waxman graph (as we added random links to connect the graph). Note that the average edge length has particular significance in SERNs, and with the average number of edges (closely

[1] https://github.com/caitlin-gray/ConnectedGraphGen

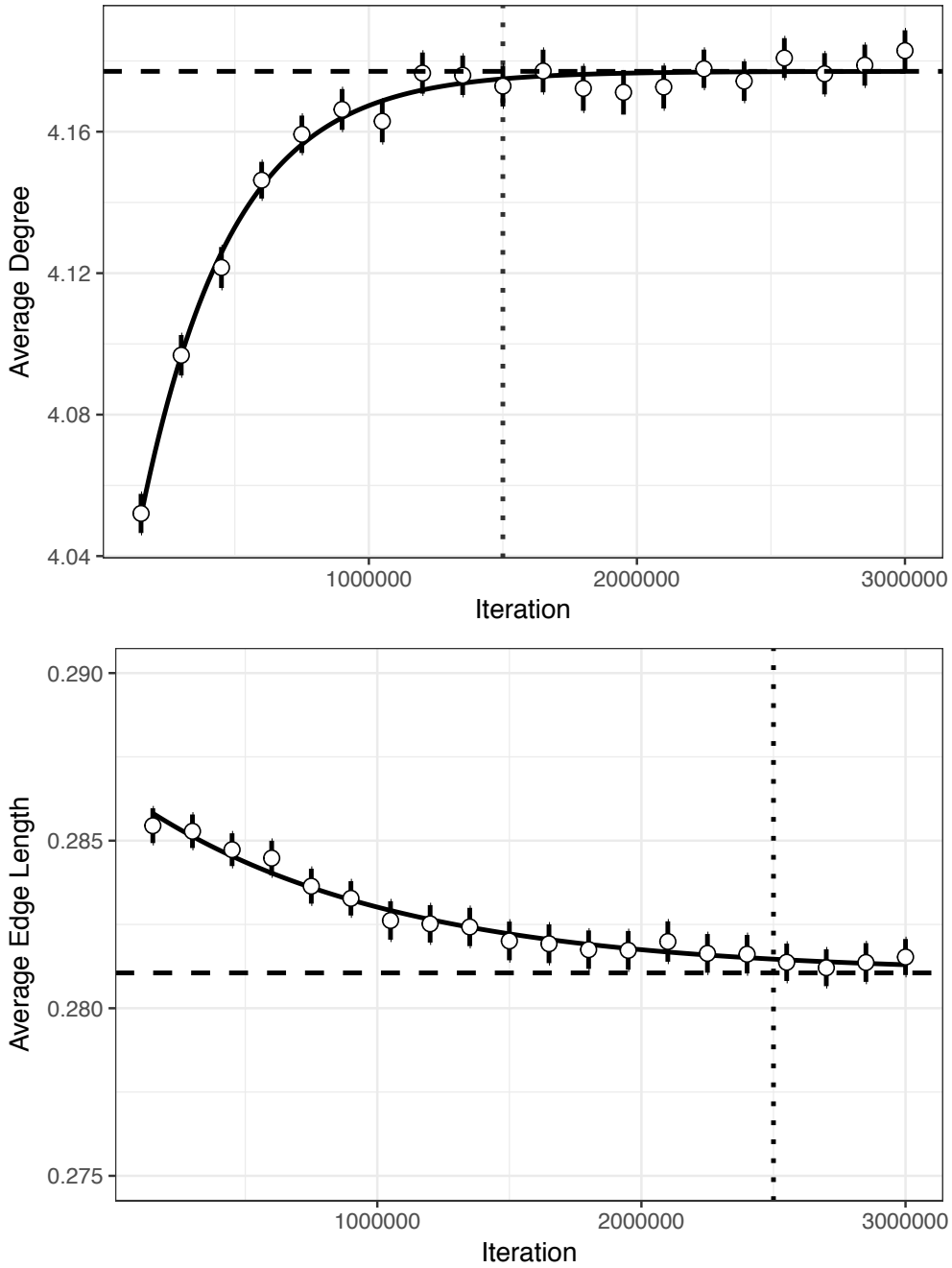


Figure 3.3: Summary statistics over the MCMC process for a Waxman network with $n = 1000$ nodes. The mean of the average degree and average edge length over 200 chains are shown with 95% confidence intervals. The solid fitted regression curve is shown, and the dashed line represents the fitted asymptote. Note that there is evidence for convergence at approximately 1.5 and 2.5 million iterations for average degree and average edge length respectively.

related to average degree) creates a minimal set of sufficient statistics for the parameters of the Waxman graph [119]. After convergence we expect no change in the distribution of summary statistics of the network as they are being drawn from the same underlying distribution. We investigate the change in these statistics to provide evidence for convergence.

Figure 3.3 shows the confidence intervals of average degree in 200 chains of the MCMC process, i.e., values at intervals along the process in 200 runs of the algorithm. This demonstrates a steady increase in average degree as the algorithm progresses. We suggest that there is no significant change in average degree after 1.5 million iterations, and we have reached the average degree of $P\{G|G \text{ is connected}\}$. The average edge length changes significantly but the magnitude of the change is much smaller. Additionally, it appears to converge slightly slower than the average degree, reaching within 99.9% of the fitted asymptote at $\sim$2.5 million iterations. Therefore, we have evidence that the system has converged and we are sampling from the posterior distribution of connected Waxman networks.

3.6.1 Iterations until convergence

To estimate K, the number of steps required until convergence, we must investigate how the number of iterations to convergence scales with the number of nodes in the network. Therefore, determining convergence by eye is insufficient. We develop a framework to automate the process and give estimates of the required iterations to convergence. First, we use non-linear least squares to fit an exponential function to the average degree over the iterations and determine when the average degree distribution is no longer changing. The function, of the form

$$f(x) = Z + Ae^{-Bx}, \tag{3.15}$$

is fitted to the full data (not just the means) to determine the parameter Z. This fitted parameter is the average degree of the target ensemble $P\{G|G \text{ is connected}\}$ after convergence, see Figure 3.3. We define strong evidence for convergence to be when the fitted values are within 0.1% of this value.

We apply this framework to the MCMC process for varying n to determine the scaling of convergence. From the results in Figure 3.4 we note that the line of best fit is a power-law with an exponent of 1.99 ± 0.04. We conclude that the mixing time of this algorithm (number of iterations to convergence) is approximately $\mathcal{O}(n^2)$. This agrees with the theoretical analysis in Section 3.5.2. We note that we see the same results when fitting other functions, for example a logistic curve.

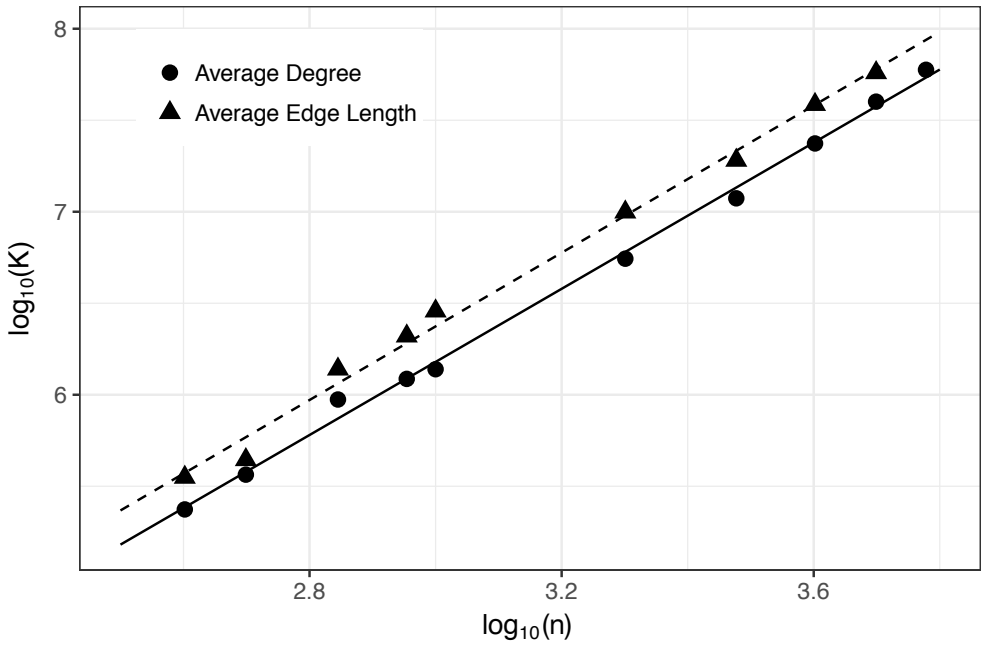


Figure 3.4: Log-log plot of the iterations to convergence of the algorithm for varying size networks using average degree (circles) and average edge length (triangles) as the summary statistic. The slope of the fitted line for average degree (solid) is 1.99 ± 0.04, and for average edge length (dashed) is 2.01 ± 0.06. This supports the $\mathcal{O}(n^2)$ mixing time expected over the edges in a network.

To provide further evidence for this $\mathcal{O}(n^2)$ complexity we conduct a similar analysis with the average edge length. We again fit an exponential model and as before the parameter Z is the asymptote taken to be the average edge length of the target ensemble. The iterations until convergence, as calculated by the average edge length is shown in Figure 3.4 (triangles). The average edge length converges more slowly than the average degree. This is expected as some links cannot be removed until other links provide new paths through the network. It displays the same scaling, with the exponent of the line of best fit of 2.01 ± 0.06, providing further evidence for convergence $\mathcal{O}(n^2)$.

Combining with results from Section 3.5.1, we demonstrate convergences in these statistics of $\mathcal{O}(n^2)$, giving evidence that we have converged to the distribution of interest.

The practical implications of this analysis is to determine how much 'burn-in' is appropriate for graphs of size n. Burn in is the number of samples discarded from the chain to ensure we are taking dependent samples from the distribution of interest. Our analysis shows that the two statistics of the graph have different coefficients of convergence time, and care should be taken to determine sufficient burn-in time to have convergence of the graph itself. Often burn in is conservative and our analysis in the above section can be used to determine an appropriate number of iterations for the chain [112]. For example, from Figure 3.3 we may choose to implement 3 or 4 million iterations before taking a sample. Note that despite this the scaling remains $\mathcal{O}(n^2)$.

3.6.2 Small network analysis

In the above discussion we have used graph statistics to determine the scaling of the convergence. Ideally, we would like to observe the entire ensemble of graphs and determine when we observe convergence for a variety of graph sizes. There are many reasons this is not feasible. We would have to compare our distribution with the accept-reject algorithm which is prohibitively slow and was the reason for the MCMC process in the first place. Also, the number of graphs in the ensemble is $\mathcal{O}(2^{n^2})$, making sampling enough graphs to get a good estimate of the ensemble infeasible for even reasonable n. To give evidence that the entire ensemble, not just some statistics, converges, we restrict the network to $n = 5$. Here there are only $2^{10} = 1024$ graphs in the ensemble, a reasonable enough number to find the posterior probabilities of each graph using both the accept-reject method and the proposed MCMC algorithm. Initially, we perform accept-reject sampling to determine the empirical distribution of the connected graph distribution. We take one million samples from the Waxman graph distribution and reject samples that

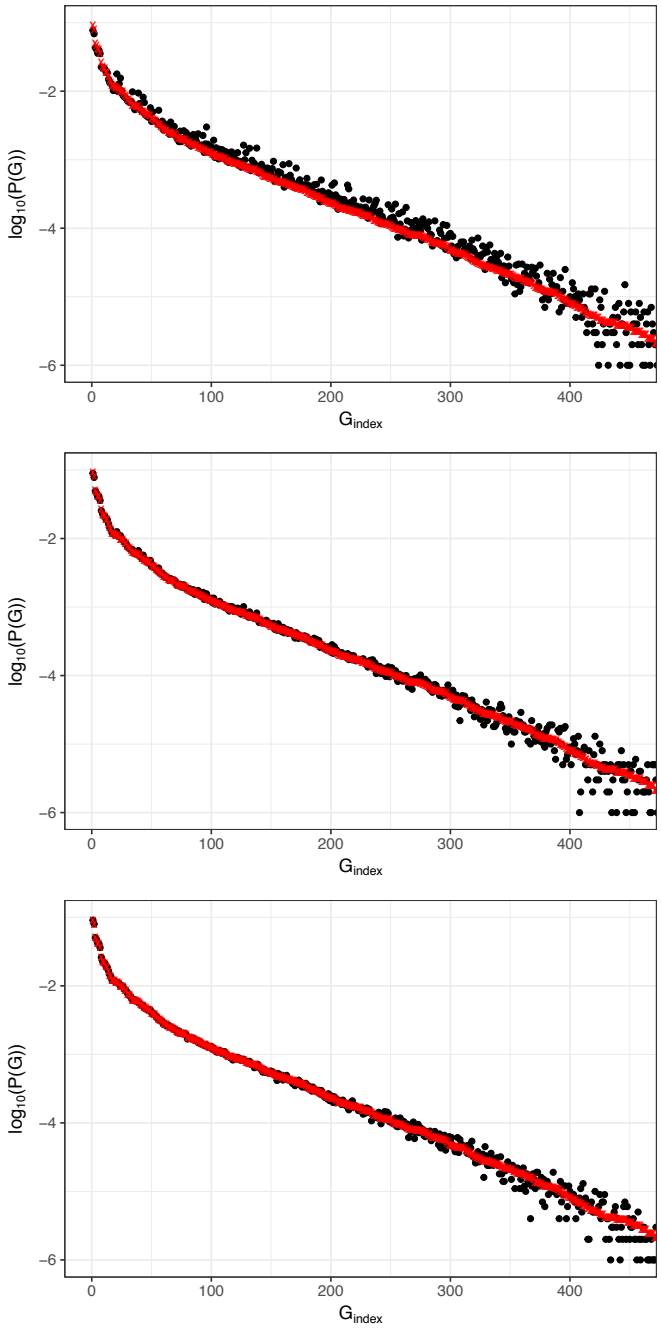


Figure 3.5: Probability distribution $P(G)$ for the 450 most probable networks for a Waxman graph with $n = 5$, $s = 5$ and average degree $z = 3$ using the accept-reject method (red) and the MCMC algorithm (black). Top: $K = n^2$. Middle: $K = 3n^2$. Bottom: $K = 10n^2$. The graphs are ordered by frequency in the accept reject algorithm.

are not connected, resulting in just over 200,000 connected samples. We find that this is sufficient to generate a good estimate of the probability of each graph and increasing the number of samples does not change the results. Next, we run one million MCMC chains and save the samples after K iterations. Figure 3.5 shows the probability of each network in the ensemble, ordered by probability from the accept-reject method (red), and the associated probabilities from the MCMC chains (red) for 3 values of K. For $K = n^2$, there is still some noise in the MCMC estimates. This is expected as after n^2 iterations each edge is only expected to have been chosen once on average. After $K = 3n^2$ — using the intercept of 3 from the above analysis — produces almost indistinguishable distributions to the accept-reject method as well as samples from the ensemble with $K = 10n^2$. This gives evidence that the convergence time for the graph statistics are also reasonable for the entire graph ensemble.

3.7 Discussion

We have introduced our algorithm in the context of generating connected networks. However, this method generalises to generate networks from the probability distribution given by

$$P(G|G \text{ has some properties}), \tag{3.16}$$

assuming the properties can be tested. For example, generating a network without self loops or multi-edges would be easily implemented as above. Although we only condition on connectedness here, the process is not restricted to a single property, a set of properties can be used. There are many constraints this method could be applied to in this form, or by changing the proposal. Other types of connectivity, e.g., k-connectivity, and using other models beyond the Waxman graph are natural extensions.

This type of method also has applications in modelling many real world networks. For example, the Internet is an inherently connected network that can be modelled by connected Waxman graphs. Generating samples from the desired model can be used when modelling dynamics over these networks or in conjunction with other inference methods, such as Approximate Bayesian Computation [127]. This method can also be used in other fields. Networks are often used to model animal ancestries that often have more conditions than a typical random graph model. The relationships between animals must satisfy a variety of conditions. At the very least the network must be a directed graph with no cycles. Further, there may be some constraint on the number of offspring an animal can have, or even the number of offspring in a certain time interval. Once likelihoods have been derived

or collected from data, the method can include these types of constraints that are based on node attributes (e.g., birth date).

Different constraints types It is worth noting that the choice of proposal distribution, $Q(G'|G)$, impacts the properties that can be tested. In this implementation, the proposal considers individual node pairs, and at each step changes a single link. This proposal is practical if we can explore the space $P(G|G$ has some properties) in single edge transitions. For example, graph ensembles with constraints including, G has no self loops, G is connected, G has a fixed diameter *etc*can all be explored in single link changes. Conversely, constraints that are broken by any single link change require a different proposal. For example, if we were to fix the number of triangles or exact degree sequence (i.e., the configuration model) in the network, our proposal distribution would need to facilitate this. In these cases, an 'edge swap' proposal that keeps the number of links constant [53, 139] would be an appropriate choice. The new proposal distributions may no longer be symmetric so we must consider the ratio $Q(G'|G)/Q(G|G')$ in our sampling process.

Different network types The above algorithm assumes that the probability distribution of the network has the form of a SERN given in (3.9). However, other probability distributions can easily be used. Note that we must be able to calculate the ratio of probabilities of graphs that differ by one step in our proposal. The ensemble of exponential random graph models (ERGMs) [87] has a simple likelihood calculation and the ERGM literature often uses methods from the MCMC family to sample the graphs.

Sampling directed graphs can also be implemented as above; instead of considering link (i, j) equivalent to (j, i), they are distinct edges that can be proposed and edited separately. Perhaps the ensemble of interest has weighted or multi edges. We can sample this family given the probability distribution that provides the likelihood of each graph and its corresponding edge weights. However, we would need to consider the proposal distribution to ensure we explore the entire space of graphs. For example, the proposal above removes edges if the node pair already has an edge. To sample multi-graphs if we choose an edge then the proposal must either remove or add another edge. Similarly, for weighted graphs we may propose adding an edge and drawing a weight from some distribution based on the underlying model.

Possible computational speed ups If the complexity or computation time of each step can be reduced, either by using more efficient processes or reducing the number of required processes, the overall efficiency can be improved. As shown above, the connected graph algorithm introduced has an average complexity of $\mathcal{O}(1)$ *per iteration* as to the $\mathcal{O}(n)$ connectivity check is only required every n iterations. In general, checking that the proposed graphs satisfy the constraints presents a possible bottleneck.

A speed up heuristic, proposed by Gkantsidis *et al.* [53], on a simple edge switch Markov Chain, attempts to reduce the requirement of checking connectedness by only running the check after T 'switch' transitions and rejecting if disconnected. This produces a concatenation of Markov Chains that maintain the required stationary distribution. This improves the run time of the algorithm that would otherwise check for connectedness at each step due to the edge flip proposal distribution. While we could use this speed up factor in the single link Metropolis-Hastings method proposed, we only check for connectedness when the proposal removes a link, compared to every step. Rejecting all T transitions (both link additions and removals) if the graph becomes disconnected would slow mixing. Hence, it is unlikely that this speed up method would produce the same dramatic increase in complexity observed in [139]. This highlights the importance of considering the proposal and the algorithm for testing the constraint together when designing the sampler for a specific constrained ensemble.

Alternative algorithms to sample connected graphs present opportunities for improving complexity in various situations. Eppstein *et al.* [44] present a dynamic connectivity check in $\mathcal{O}(\sqrt{n})$ per change in the graph. This is promising; however this update is required at every addition or deletion of an edge, rather than only at deletion, so would not improve overall performance in the case described above. These types of dynamic algorithms provide an opportunity to allow sampling of graphs with other properties, e.g., k-connectivity, more efficiently.

While the various additions described above would not improve the complexity of the specific example given above, they would be quite useful in other applications of the algorithm to sample from alternative ensembles with various constraints. For example, sampling from an ensemble with a diameter of certain size would require a constraint check at each step and may benefit from the 'Gkantsidis' speed up. Additionally, if an edge flip proposal is used the improved complexity of the connectedness routine at each step Eppstein *et al.* may provide marked improvements.

Initialisation of the model We initialise the algorithm by simulating a graph from the model of interest; e.g., the Waxman network, and connecting arbitrarily. However, any connected network can be used in this step as the MCMC process by design forgets the initial point of the Markov chain. In general, any network with non-zero probability in the ensemble is sufficient. This is particularly useful where the generation from the $P(G)$ of interest is computationally expensive. However, starting 'further' from the distribution of interest may increase time to convergence. We discuss this further in Section 4.1.

Sampling multiple graphs We have focussed here on the simulation of a single graph, assuming that multiple graphs can be sampled by running multiple chains in parallel. As all chains of the MCMC end up sampling from the posterior, an easy way to get multiple samples is to run multiple chains past convergence in parallel and take samples from each. We can also sample multiple graphs from the same chain. MCMC creates dependent samples by construction, in our case only one link might change between samples. While these will still give an estimate of the posterior (albeit with higher variance) thinning of the chain will need to be employed if one wishes to create near independent samples. We expect the number of iterations between notionally independent samples to be of the same order (not necessarily the same time) of mixing time, $\mathcal{O}(n^2)$ with the proposal distribution used above. This is intuitive as each node pair must have the opportunity to change to create independent graphs.

This chapter described an algorithm to create random networks from a known ensemble conditioned on an extra desired property. We used a Bayesian framework, implemented with Metropolis-Hastings, to generate connected random networks. This implementation can be extended to include other desired properties of a network. We demonstrated the time complexity is $\mathcal{O}(n^2)$ with strong evidence of convergence to the desired ensemble. Future work includes applying this algorithm to other constraints and networks. In the next chapter we investigate extending the proposal distribution, $Q(G'|G)$, to improve the efficiency of the algorithm. We also touch on more advanced methods such as Sequential Monte Carlo in this context.

Chapter 4

Extending the connected network sampler

In the previous chapter, we developed a connected graph sampler and highlighted that the method was able to sample from the distribution in reasonable time. We introduced potential extensions to improve the convergence complexity and indicated alternative Bayesian methods could be used to sample from these distributions.

While the scaling and computation time for the above sampler is not prohibitive, the scaling of $\mathcal{O}(n^2)$ is far from ideal when modelling large real world networks. In this chapter we implement and compare extensions to the samplers introduced above, to improve this scaling to $\mathcal{O}(e)$ in the number of edges. This is a significant improvement, particularly in sparse graphs with $e \sim \mathcal{O}(n)$.

The ability to sample from the network efficiently also provides the opportunity to investigate the differences between the connected and regular Waxman ensembles. Beyond gaining an understanding of the conditional ensemble and using the samples created, these comparisons can be used to reduce bias in the current maximum likelihood estimators for the Waxman ensemble when conditioning on connectedness.

Specifically, in this chapter, we introduce two versions of another proposal distribution called the 'Tie-No Tie' (TNT) sampler and compare its convergence and time complexity to the original case. We take a slight detour through an algorithmic discussion on possible ways to store and implement these samplers efficiently before comparing the asymptotic complexity, runtime and runtime to convergence. We find that although the TNT sampler is more computationally intensive at each step, the improvement in convergence complexity to $\mathcal{O}(e)$ results in markedly lower

runtime when compared to the original sampler. We also apply another Bayesian sampling method, SMC, by introducing a very general likelihood annealed SMC that extends the scope of what types of graphs we can sample and properties on which we can condition. Finally, we generate comparisons between properties of the connected network ensemble and the original Waxman and consider their implications in using current maximum likelihood estimation methods to model real-world graphs.

4.1 TNT proposals to improve convergence

There are two ways to improve the complexity or computational requirements of these types of algorithms: reduce computation at each step, or reduce the number of iterations required for convergence. We touched on possible considerations for improving computation at each step in Chapter 3, and here we introduce a modification to the proposal distribution to improve the number of iterations for convergence.

One of the bottlenecks in the overall performance of these algorithms is the required number of iterations for convergence. Performance improvement of the algorithm comes by reducing the number of rejected proposals and improving mixing of the chain.

The Exponential Random Graph Model (ERGM) [87] literature has pioneered the use of different proposals for sampling graphs from these types of models. The R package `ergm` [69] has no less than 22 different proposals for MCMC sampling of ERGMs. These allow for the sampling of different types of graphs such as bipartite graphs, directed graphs and graphs with fixed degree sequence. One very relevant proposal is the 'Tie-No Tie' (TNT) proposal [95] designed as a simple extension to the single dyad proposal used above. As we are often in the realm of sparse graphs, using the TNT proposal (or variants thereof) has the potential to improve convergence and allow sampling of larger graphs in more reasonable times.

4.1.1 TNT sampler

When using MCMC with the single dyad proposal each dyad has equal probability of selection. In sparse graphs non-links are much more likely, and so this algorithm will often propose adding a link. However, both the prior (standard Waxman) and posterior (connected Waxman) are often sparse graphs, and so, to retain this

sparsity, the move is likely to be rejected. The TNT proposal aims to reduce the rejection rate.

The TNT sampler selects, with probability $p = 0.5$, the set of links or non-links. A dyad from the chosen set is then selected uniformly at random. This dyad is then toggled (edge to non-edge or vice versa) to create the new proposal.

Despite both the original algorithm and the TNT chains having the same limiting distribution, the bias introduced towards changing existing links reduces rejecting proposals in sparse graphs and often leads to faster mixing times [69].

Algorithmic considerations

In practice, we must consider how to implement this sampler, particularly how to efficiently choose the dyad at each step. In Chapter 3 this was as simple as choosing two distinct nodes in $\mathcal{O}(1)$ time and storing the existing edges in $\mathcal{O}(e)$ memory. Conceptually the TNT sampler is quite simple; however as we propose a new dyad at every step, we would like to ensure that the memory and computation required to choose dyads with this new sampler do not outweigh the potential gains. As with many algorithms, the set-up we choose will depend on the desired trade-off between space and time. We present two options here:

1. **High memory, lower computation:** A classic TNT sampler with $\mathcal{O}(n^2)$ space and an $\mathcal{O}(1)$ time-complexity update every step
2. **Low memory, higher computation** A modified TNT sampler with $\mathcal{O}(e)$ space and an $\mathcal{O}(e)$ time-complexity update every $\mathcal{O}(1/e)$ steps, otherwise an $\mathcal{O}(1)$ update; and so on average $\mathcal{O}(1)$ updates.

Recall, the original sampler introduced in Chapter 3 has an update of $\mathcal{O}(1)$ and $\mathcal{O}(e)$ space. While the original sampler has lower complexity and space at first glance, we will see that the convergence improvements of the TNT samplers dramatically improve overall performance.

1) High memory, lower computation To implement the original TNT sampler, we maintain a list of the edges and non-edges in a single contiguous array of length n^2 along with a pointer of the number of edges e. The dyads with index less than the pointer are edges and those above this index are non-edges.

To choose and update edges: Choose the set of edges with probability 0.5, and in this case choose a random index in $[0, e-1]$. If the move is accepted we

remove the selected edge by switching the edge with the edge at index $e-1$ and we reduce the pointer, e, by one.

To choose and update non edges: Alternatively, we choose the set of non-edges with probability 0.5, and in this case choose a random index in $[e, n^2]$. If the move is accepted we add the selected edge by switching the chosen non-edge with the non-edge at index e and we increase the pointer, e, by one.

This proposal is no longer symmetric so we must take this into account when calculating the acceptance ratio.

$$\frac{Q(G|G')}{Q(G'|G)} = \begin{cases} (e+1)/e^c, & \text{if adding edge from G to G'}, \\ (e^c+1)/e, & \text{if removing edge from G to G'}, \end{cases} \tag{4.1}$$

where $e^c = n(n-1)/2 - e$ is the number of non-edges. This proposal step is $\mathcal{O}(1)$ in computation but $\mathcal{O}(n^2)$ in memory because we need to store the complete list of possible edges. It also assumes constant time access to the array, which is possible in theory, but for large n we need to consider that constraints on cache memory may lead to degradation in actual performance.

2) Low memory, higher computation To reduce the space requirements without hindering complexity, we use a slightly altered version of the TNT sampler. Each step chooses the set of links or *dyads* with probability $p = 0.5$. Note the choice of links or dyads, rather than links or non-links. We could potentially choose a link from both pathways. This removes the requirements of knowing the non-edges, reducing the required storage and lookup cost.

To reduce the, potentially large, overheads of changing the size of the edge list, we store the edges in an array of fixed size. A variable pointer of the number of edges, e, determines the number of non-zero elements in the array. We choose the length of this list carefully to be more than the edges we expect and much less than n^2. For large graphs this is easy to do.

To choose and update edges: Choose the set of edges with probability 0.5; then, choose a random index in $[0, e-1]$. To remove an edge we put the last element of the array in its place and adjust the pointer.

To choose dyads: Choose the dyads with probability 0.5, and then choose a dyad by choosing two distinct random nodes between 1 and n and propose the switch. If adding, add the edge to the end of the array of edges and adjust the pointer. If removing, we will have to do an, at worst, $\mathcal{O}(e)$ search for the link to remove. The removal of an edge in this way only occurs with probability $1/(2e)$ each iteration and so has little impact computation time.

This proposal is no longer symmetric so we must take this into account when calculating the acceptance ratio.

$$\frac{Q(G|G')}{Q(G'|G)} = \begin{cases} \frac{n(n-1)}{2(e+1)} + 1, \text{if adding edge from G to G'}, \\ \frac{2e}{n(n-1)}, \text{if removing edge from G to G'}. \end{cases} \tag{4.2}$$

Overall, this is $\mathcal{O}(1)$ in computation and $\mathcal{O}(e)$ in storage. Therefore, for sparse graphs this will decrease storage requirements.

4.1.2 Convergence comparison

The TNT sampler is designed to improve mixing and decrease time until we can reasonably assume convergence. To test convergence we look at traces of the summary statistics of the graph as introduced in Section 3.6. We fit an exponential curve to the summary traces to determine the asymptotic average value and then estimate K, the number of steps until convergence. As in the previous chapter we define convergence to be within 0.1% of the asymptotic value. In the following section we show results for the low memory TNT sampler; the classic TNT sampler gives indistinguishable results.

Figure 4.1 shows the summary statistics over the MCMC process using the TNT sampler, initialised with an Erdős-Rényi graph with $z = 4$. That is, each node is given an (x, y) coordinate and then each node pair is connected with some constant probability, $p \approx z/n$. There is an increase in the average degree (Figure 4.1 (top)) and a drop in average edge length (Figure 4.1 (bottom)) due to extra edges added for connectedness and the removal of unlikely long edges. By eye, we can see evidence for convergence after about $10,000$ iterations. In comparison, the original sampler converges in $\sim$2 million iterations.

The Erdős-Rényi initialisation results in a few interesting characteristics of the traces. Namely, each plot has two distinct trends, one curve does not explain the entire data. For the trace of the average degree in Figure 4.1 (top) there is a steep drop in the first few hundred iterations before following the expected increase to a slightly higher degree than the original $z = 4$. As the initial ER graph has nodes attached with equal probability there is no dependence on distance, and so the graph will have long links that are unlikely in the Waxman ensemble. The TNT sampler preferences edge removal and quickly accepts removal of the unlikely long links to drop the average degree. Subsequently, to maintain connectivity and yet sample from the Waxman ensemble of interest the average degree increases as more short links are required. To ensure a good fit and accurate estimate of the number

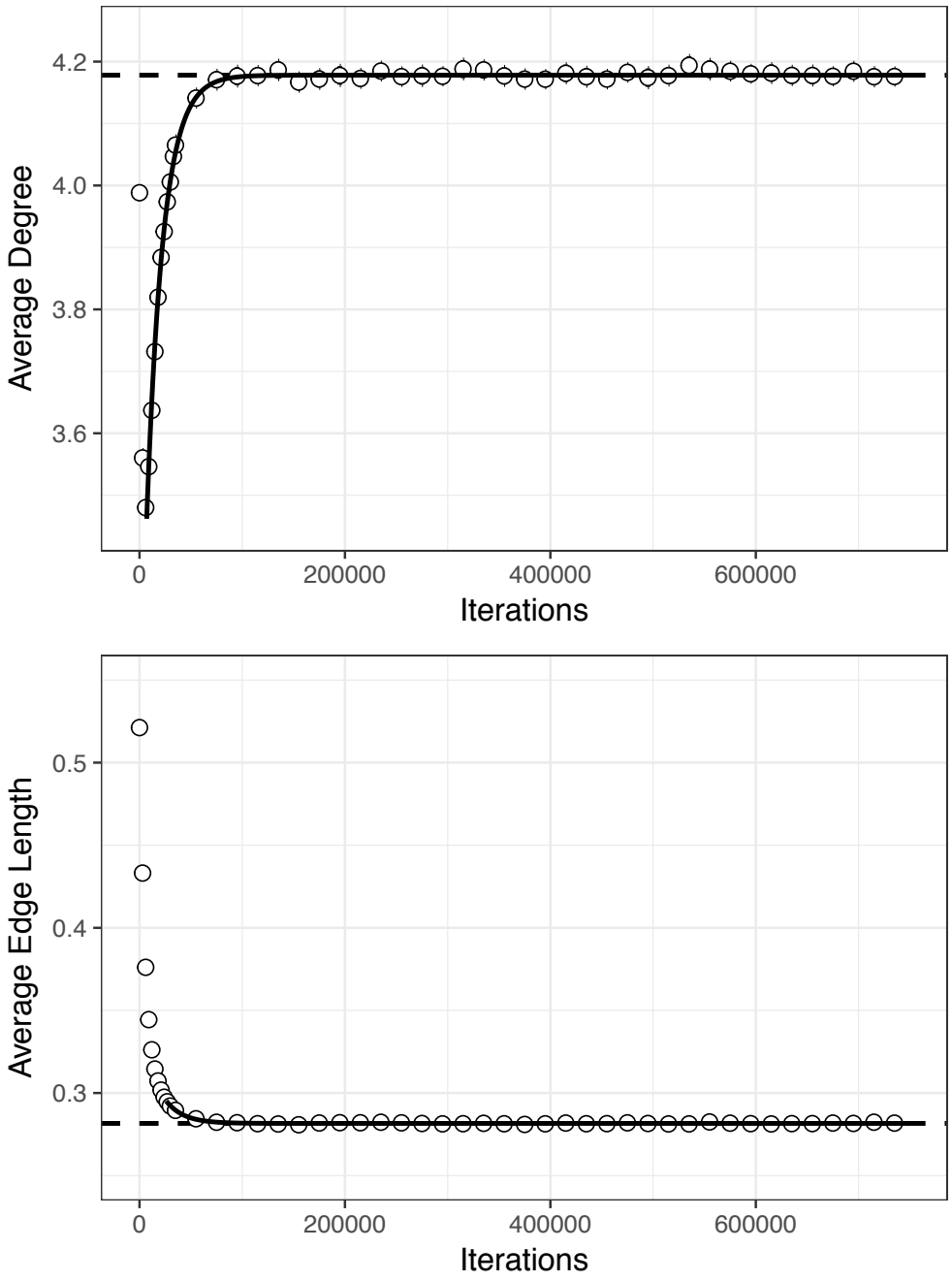


Figure 4.1: Summary statistics over the MCMC process using a TNT sampler for a Waxman network with $n = 1000$ nodes. The range of iterations is chosen to The solid fitted regression curve is shown, and the dashed line represents the fitted asymptote.

of iterations until the curve is within 0.1% of the asymptote, we fit the exponential curve from the point local optima.

For the average edge length in Figure 4.1 (bottom) an exponential curve from iterations = 0 fits poorly. There is a very steep reduction in average edge length (resulting from the removal of the long edges) on a similar time-scale observed in the average degree trace. It is important that the exponential curve fits the data well to approximate when the ensembles are close to the carrying capacity. Therefore we choose fit from the local optima in the average degree trace to ensure a good fit and reliable convergence results. Starting 'closer' to the desired ensemble, e.g., initialising with a Waxman network, would reduce the iterations to convergence; however the scaling would remain the same. It is interesting to note that although starting from an ER graph requires more iterations, it provides less noisy measurements when approximating the value at which the fitted curve is within 0.1% of the asymptote.

As in Chapter 3 we fit the exponential to the curves as described above for various parameter values. Figure 4.2 depicts how the number of iterations until convergence scales with the number of nodes (top) and the average degree (bottom). Recall from Section 3.6 that the scaling for the original proposal was $\mathcal{O}(n^2)$. On a linear scale we see that the average degree convergence scales linearly for constant z. We see that the number of iterations until convergence is also linear in the average degree for fixed n. Combining this data, Figure 4.3 shows the log-log plot of the iterations until convergence against the number of edges with a slope of 1.01. We conclude that the convergence with respect to these parameters is $\mathcal{O}(zn)$, which is equivalent to $\mathcal{O}(e)$. This is a marked improvement to the $\mathcal{O}(n^2)$ complexity from the original sampler in Chapter 3, especially in sparse graphs where $e \sim \mathcal{O}(n)$.

4.1.3 Complexity comparison

In the previous section we established that the convergence complexity of the TNT samplers is significantly lower than that of the original sampler. Here we consider overall complexity, combining the complexity at each step with the convergence complexity. We show that while overall theoretical complexity is equivalent the TNT samplers are superior in the time required to take a sample

Recall that the TNT sampler has $\mathcal{O}(e)$ convergence complexity. However, to achieve this improved convergence we now propose edge removals more often, and therefore require the $\mathcal{O}(n)$ connectivity check every 2 iterations on average. Therefore, Theorem 2 describing constant complexity in n no longer holds.

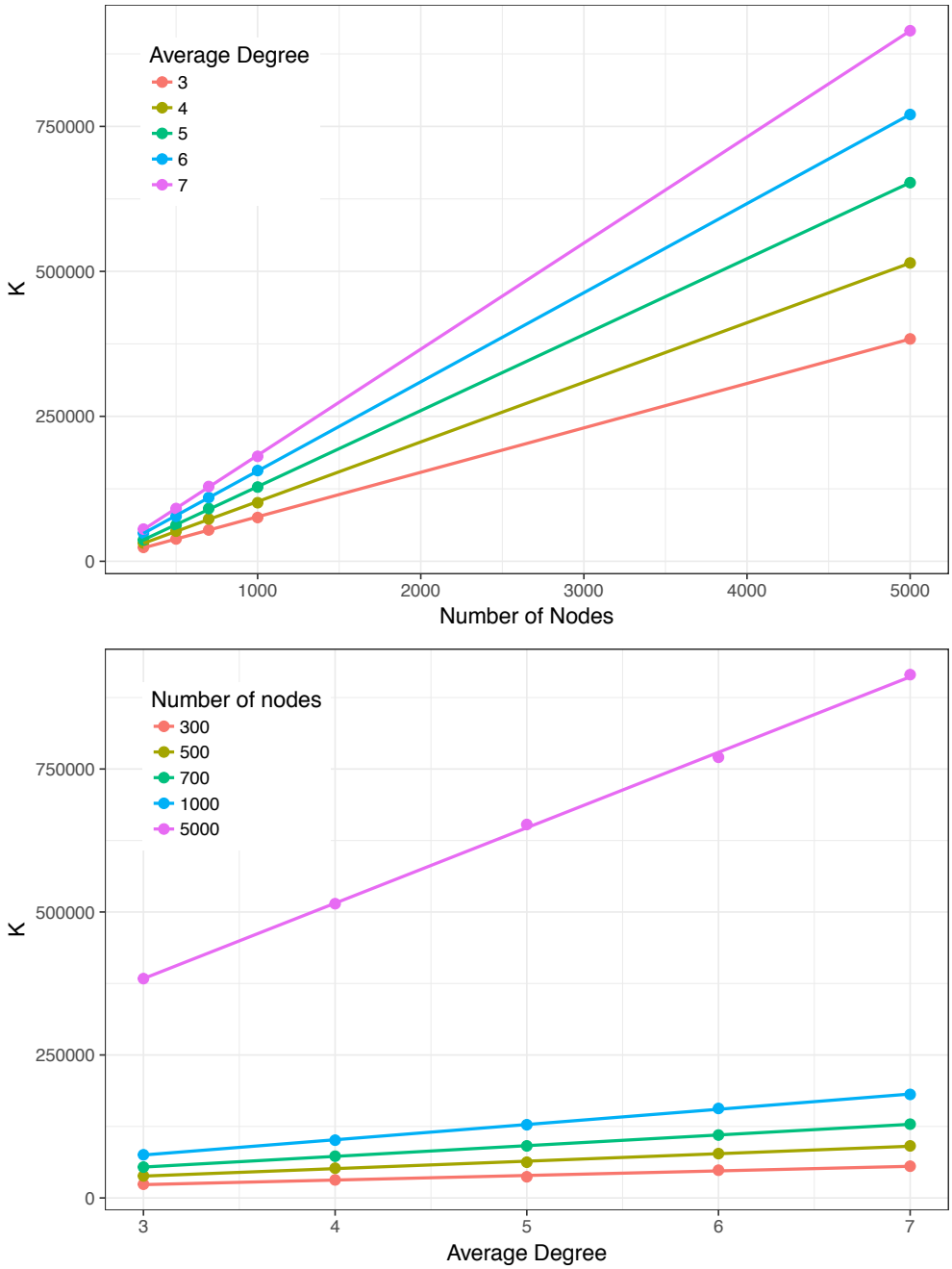


Figure 4.2: Scaling of the number of iterations to convergence with respect to the number of nodes (top) and the input average degree (bottom). We show the results for convergence of average degree (circles) as the summary statistic. Average edge length produces almost indistinguishable results and has been omitted to reduce clutter. We observe the linear relationship with slope that depends (linearly) on the other parameter suggesting convergence depends on both these parameters.

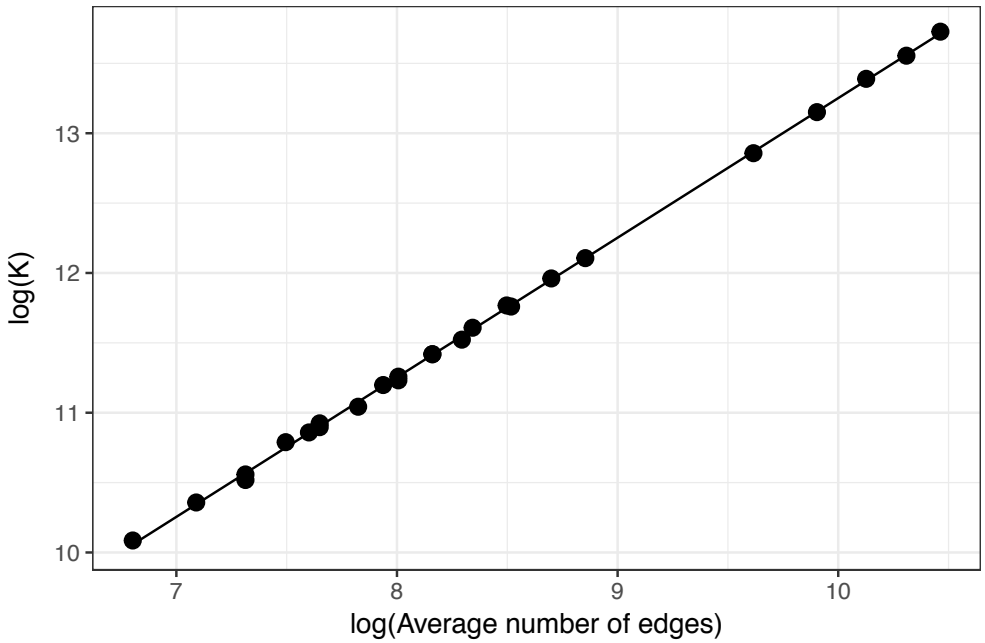


Figure 4.3: Log-log plot of the iterations to convergence of the algorithm for as the average number of edges (zn) changes. We show the results for average degree (circles) as the summary statistic. Average edge length produces almost indistinguishable results and has been omitted to reduce clutter. The slope of the fitted line (solid) using average degree is 1.005 ± 0.005, and for average edge length (not shown) is 1.01 ± 0.003.

Table 4.1: Complexity comparison for different proposals

Step	Network Type	Original proposal	TNT low memory	TNT high memory
Worst case connectivity check	Sparse	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	Dense	$\mathcal{O}(n+e)$	$\mathcal{O}(n+e)$	$\mathcal{O}(n+e)$
Connectivity check required	Sparse	$\mathcal{O}(1/n)$	$\mathcal{O}(1/2n) + \mathcal{O}(1/2)$	$\mathcal{O}(1/2)$
	Dense	$\mathcal{O}(e/n)$	$\mathcal{O}(1/2e) + \mathcal{O}(1/2)$	$\mathcal{O}(1/2)$
Iterations to convergence	Sparse	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	Dense	$\mathcal{O}(n^2)$	$\mathcal{O}(e)$	$\mathcal{O}(e)$
Complexity to convergence	Sparse	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	Dense	$\mathcal{O}(en)$	$\mathcal{O}(en)$	$\mathcal{O}(en)$

Table 4.1 shows the complexity for each of the algorithms at each step of the process. In general, we are interested in the space of sparse graphs, where $e \sim n$, as they are most likely to be unconnected and are often seen in practice as links can be expensive, e.g., social networks and Internet networks. Nonetheless, we show complexity for both the sparse and dense case. Overall, the worst-case complexity to convergence is equivalent in all three algorithms. The 'gain' in complexity of convergence is traded with the requirement to check connectivity at every second step.

With the same overall complexity equivalent in both scenarios, why should we use the TNT sampler at all? Of course, the answer lies in the forgotten coefficients of the scaling; algorithms with the same worst-case complexity can still be orders of magnitude different in run time. So, does the reduction in iterations required to convergence outweigh checking the graph for connectivity more often? In short, yes. To answer this question we look at the actual runtime of the algorithm and how it scales with the number of nodes in the network.

Figure 4.4a shows the average time for 500 iterations for each of the methods. As expected the original set-up is much faster and independent of n. The low and high memory TNT algorithms perform similarly. The higher memory solution requires less computation at each update, in contrast to the worst case $\mathcal{O}(e)$ list search for the low memory sampler, but requires increased memory to store the full edge list. As the n increases the low memory solution outperforms the high memory due to caching requirements of storing the entire edge (and non-edge) list.

More importantly is the time required to take a sample. The quantity of interest is the 'time to convergence'. Figure 4.4b shows the time required until convergence, where convergence is defined by a fixed number of iterations, $K = 10n^2$ for the original sampler and $K = 10e$ for the TNT samplers. It is evident that the improvement in convergence achieved by the TNT samplers outweigh the extra computation by at least an order of magnitude in this set-up.

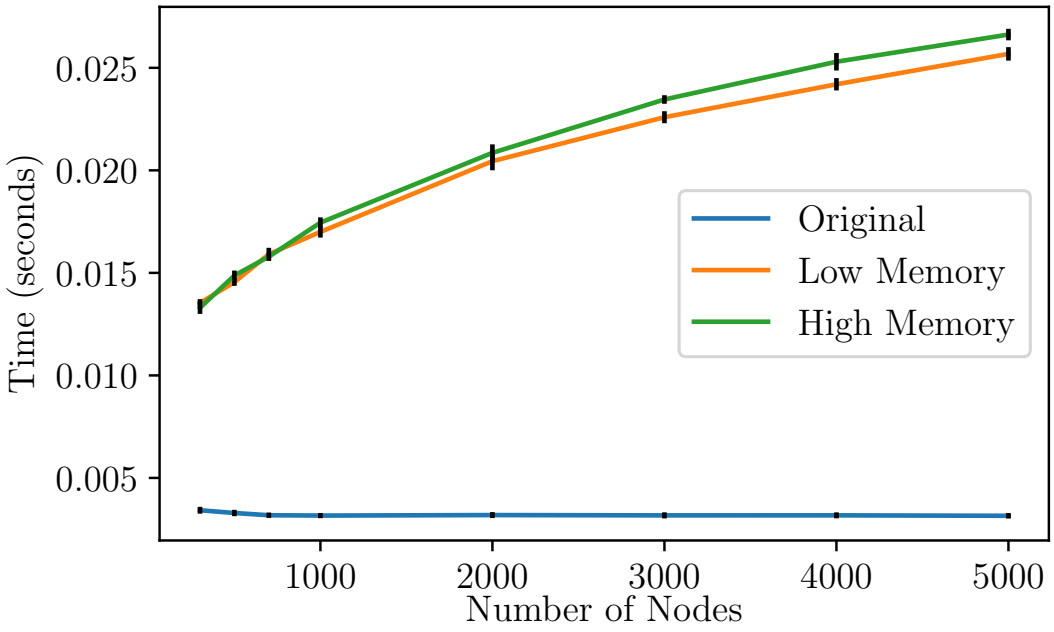
4.2 Sequential Monte Carlo to sample connected graphs

In the previous section we proposed a change to the proposal distribution of the original Metropolis-Hastings algorithm and saw an improvement in computational requirements. Alternatively, we can consider different MCMC methods to provide either computational benefits or flexibility. Well known Bayesian extensions of the MH algorithm include Sequential Monte Carlo (SMC) and Hamiltonian Monte Carlo (HMC) and the many variants/combinations of these. These methods are developed for continuous random variables and our highly dimensional discrete posterior can present issues for many out-of-the-box samplers; however some can be applied to discrete random variables like our graph space. We introduce a likelihood annealed SMC to sample constrained graphs.

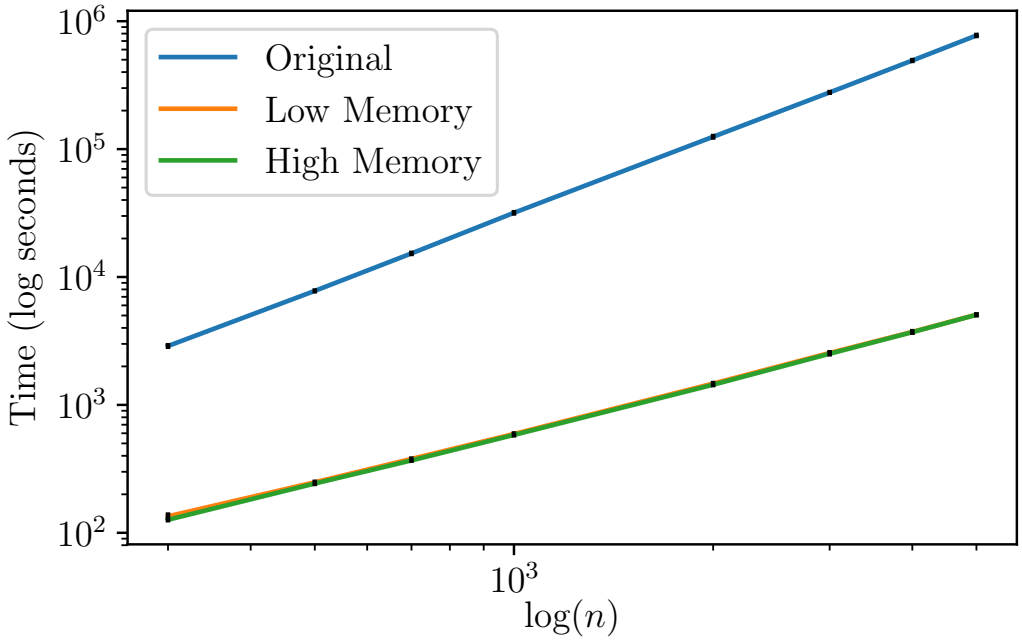
4.2.1 SMC algorithm

We introduced SMC in general in Section 2.3.3. To sample connected Waxman graphs we will utilise likelihood annealing to design a strategy that sequentially samples from different distributions that get us 'closer' to our target. Let X_G be the number of connected components in the graph G and $\gamma_t \in [2, N]$ be an integer. Then,

$$\begin{aligned}
P_0(X) &= \text{Standard Waxman} = P(G) \\
P_1(X) &= \text{Waxman with } X_G < \gamma_1 \\
P_2(X) &= \text{Waxman with } X_G < \gamma_2 \\
&\vdots \qquad\qquad \vdots \\
P_t(X) &= \text{Waxman with } X_G < \gamma_t = P_t(G|X_G < \gamma_t) \\
&\vdots \qquad\qquad \vdots \\
P_T(X) &= \text{Connected Waxman} = P(G|G \text{ connected})
\end{aligned} \tag{4.3}$$



(a) Time for 500 iterations for the three algorithms. Each iteration of the TNT sampler is more expensive.



(b) Plot of the total time until convergence. We define convergence to be $K = 10n^2$ for the original algorithm and $K = 10zn$ for the TNT samplers. On the loglog scale the slope of the original algorithm is ≈ 2 and the TNT samplers are ≈ 1. Moreover, there is a large drop in computation times for the TNT samplers.

Figure 4.4: Results for timing of the original connected graph algorithm and the two implementations of the TNT sampler.

As introduced in Section 2.3.3, at each step the samples are weighted according to the ratio of the important distribution and the current distribution. We assume resampling and moving particles at each step to gives weights $w_{i-1} = 1/S$, so in this case:

$$\begin{aligned} w_i &= w_{i-1}\frac{P_{t+1}(G_i|X_{G_i} < \gamma_{t+1})}{P_t(G_i|X_{G_i} < \gamma_t)} \\ &= w_{i-1}\frac{P_{t+1}(G_i, X_{G_i} < \gamma_{t+1})}{P_{t+1}(X_{G_i} < \gamma_{t+1})}\frac{P_t(X_{G_i} < \gamma_t)}{P_t(G_i|X_{G_i} < \gamma_t)} \\ &\propto \frac{P_{t+1}(G_i, X_{G_i} < \gamma_{t+1})}{P_t(G_i, X_{G_i} < \gamma_t)} \\ &= \begin{cases} \frac{P(G_i)}{P(G_i)} = 1 & \text{if } X_{G_i} < \gamma_{t+1}, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Note we do not need the fraction

$$\frac{P_t(X_{G_i} < \gamma_t)}{P_{t+1}(X_{G_i} < \gamma_{t+1})}, \tag{4.4}$$

as the probability of a graph having a certain number of connected components in the ensemble does not depend on the particular graph. As the weights are 1 or 0 at each step depending on the level of connectedness, we are basically doing a rejection sampling step for networks in our current distribution that are also in the more constrained distribution. This idea is also used in Multi-level splitting and can also be used to determine rare event probabilities [135].

After resampling with the corresponding weights w_i, we do a move step using the TNT sampler introduced in Section 4.1. The number of iterations recommended here varies within the literature. Some literature recommends enough steps that the MCMC sampler is likely to move with probability larger than some value, say 0.9. We have to consider that we are only changing one link at a time and so we may not get sufficient diversity in the samples with only one accepted move. In Section 3.7 we discussed that an appropriate number of iterations for thinning would be n^2 to reduce correlation between samples. While it may not be necessary to run for so many iterations, it presents a good starting estimate. There are heuristics for choosing the number of steps to ensure at least one move as introduced in Section 2.3.3.

At the end of the SMC process we have S samples from the desired distribution $P_T(G) = P(G|G \text{ connected})$.

The efficiency of this method will depend greatly on the annealing schedule that we choose, i.e., choosing the constraint values γ_i. We can look at effective sample size

and rejection rates at each step to tune this. It will depend on network parameters, as n, z and s all impact the number of connected components. At the extreme we can choose $\gamma_i = \gamma_{i-1}+1$ and initialise the original γ_0 to be the maximum number of connected components in the prior samples. However, incrementing in steps larger than 1 will decrease required computation and may not impact the results.

We have introduced the SMC method on networks here as an example. As it is quite easy to find a graph that has non-zero probability in the posterior of interest, i.e., connected graphs, SMC is unlikely to have performance improvements in this context. However, one can imagine networks and associated properties where it is hard to find any graph in the posterior. In these cases, the likelihood annealing can be extremely beneficial and reduce computation of initial graphs for the original MCMC method. Additionally, SMC methods as introduced here, sometimes also called multi-level splitting, can also be used to calculate rare event probabilities. This has been used to determine probabilities of proper graph colourings [135], and could also be used here to estimate connectedness probabilities and the relationship to graph parameters.

4.3 Impact of connectedness on graph properties

The Waxman ensemble has two sufficient summary statistics, the average edge length, $\bar{d}$, and the number of edges in the graph, e [119]. Understanding their relationship to the Waxman parameters, s and q is vital to be able to fit Waxman models to graphs of interest. If a Waxman model is fitted to a network that actually comes from a connected Waxman ensemble the estimator may be biased. Here, we investigate the impact of connectedness on the relationships between the parameters and the sufficient summary statistics.

Figure 4.5 shows the impact of connectedness on the relationship between the expected number of edges of the network, calculated using the parameters of the network, and the resultant number of edges. As highlighted in Chapter 3 there is an increase in the number of edges in the connected case as expected to obtain connectedness. This difference is more extreme in graphs with lower expected number of edges (x-axis), as these sparser graphs require more edges to obtain connectedness. We show only $s = 5$, as the empirical results gave no evidence of s dependence for $n = 100$.

Recall from Section 2.1.2 that the probability of an edge (i, j) in the Waxman

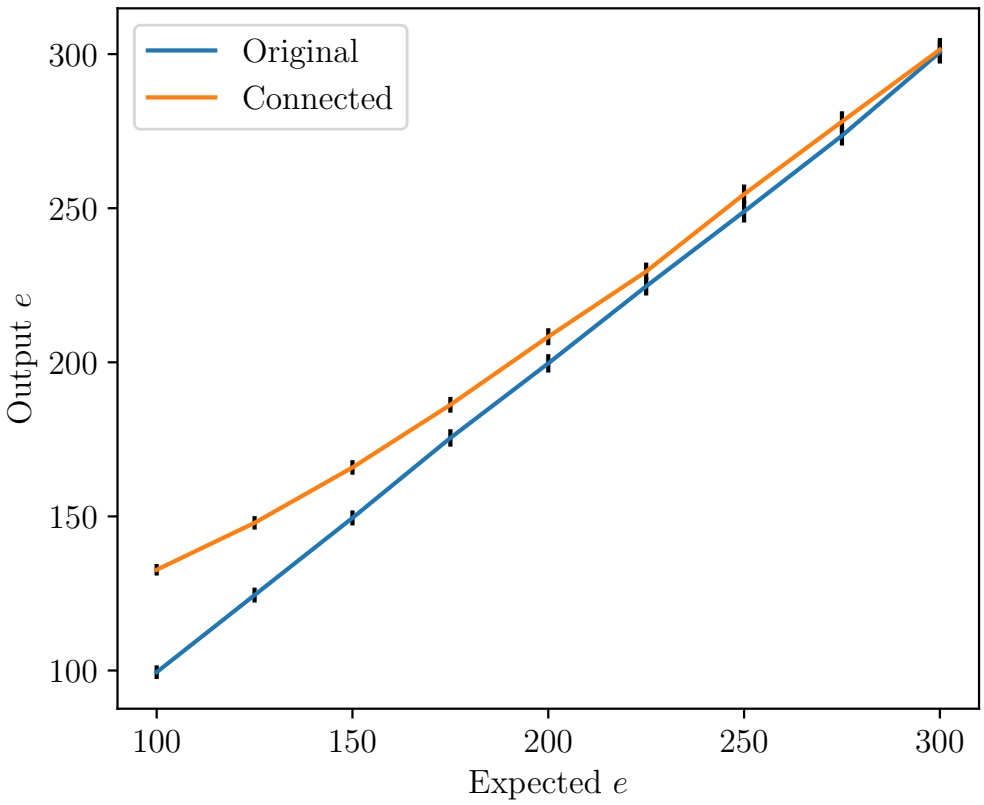


Figure 4.5: Relationship between the expected number of edges in the network and the output e. The expected number of edges is calculated from input parameters: $n = 100$, $s = 5$, $z = 4$. As expected, we see a straight line for the original ensemble (blue). The output e is larger in the connected case (orange) as extra edges are required to obtain connectedness.

ensemble is given by

$$p_{ij} = qe^{-sd_{ij}}, \tag{4.5}$$

where d_{ij} is the distance and s and q are the parameters of the graph. The parameter q is related to the average degree and s parameter by

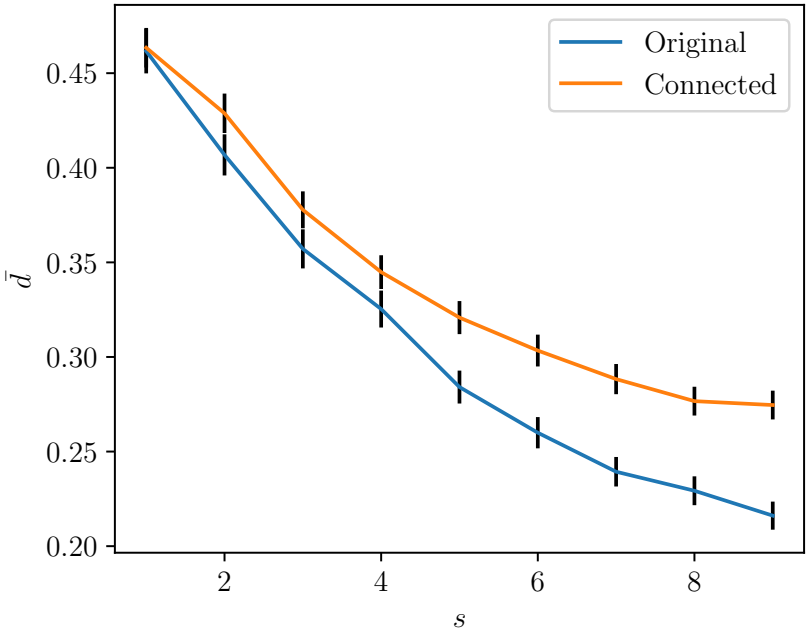
$$q = \frac{z}{(n-1)\tilde{G}(s)} \tag{4.6}$$

where $\tilde{G}(s)$ is the Laplace transform of $g(s)$, the probability distribution of the distance between two points placed randomly in a space. Also, recall $e = zn/2$. As we have seen, the connected Waxman ensemble results in a small increase in e to obtain connectedness. Empirical results suggest the parameter s does not impact this relationship, so the effective parameter q_{eff}, that controls the density of edges in the connected case, is greater than the original q.

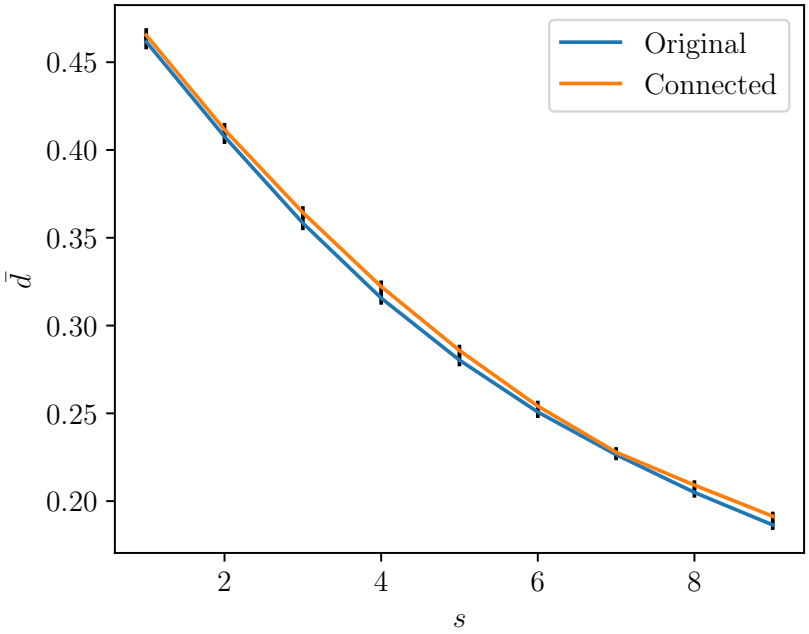
Next, we would like to know if the relationship between s and average edge length $\bar{d}$ is impacted by imposing connectedness on the ensemble.

Figure 4.6 shows the relationship between the average edge length of graphs in the ensemble and the input parameter s in $n = 10$ (Figure 4.6a) and $n = 100$ (Figure 4.6b) networks. For the $n = 10$ case, as s increases there is a discrepancy between the average edge lengths of the connected ensemble and the original. However, for the $n = 100$ graphs the confidence intervals overlap, and there is no significant difference between the original and connected ensembles. This can be explained intuitively by the distances between node pairs available to add as edges. When there are very few nodes on the space there are less edges to choose from to connect the graph, and so we will be forced to choose longer edges. For larger n there are many more nodes on the space and so it is likely there will be short edges that we can use to obtain connectedness.

Figure 4.7 shows the edge length distributions for networks with $n = 10$ and $n = 100$ and for $s = 0, 5$ and 10. As n becomes larger the edge length distribution of the connected ensemble is indistinguishable from the original Waxman. For $n = 10$ there is a significant difference between these distributions at higher s values. Networks with larger s have a stronger dependence on distance and so in general have shorter links. For larger n there are more nodes in the space so shorter edges are available to connect the graph and follow the same edge length distribution. Therefore, we see very similar edge length distributions for both cases in Figure 4.7b. Conversely, in smaller graphs to ensure connectedness the edges required are longer than observed in the original ensemble as there are less short links. Therefore, we see an increase in the average edge length for smaller graphs, as shown in Figure 4.7a.

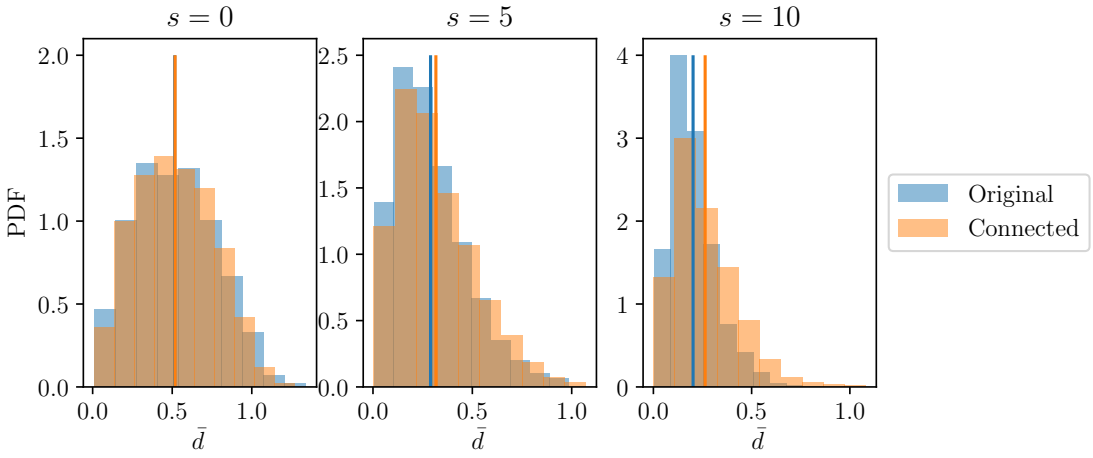


(a) $n = 10$.

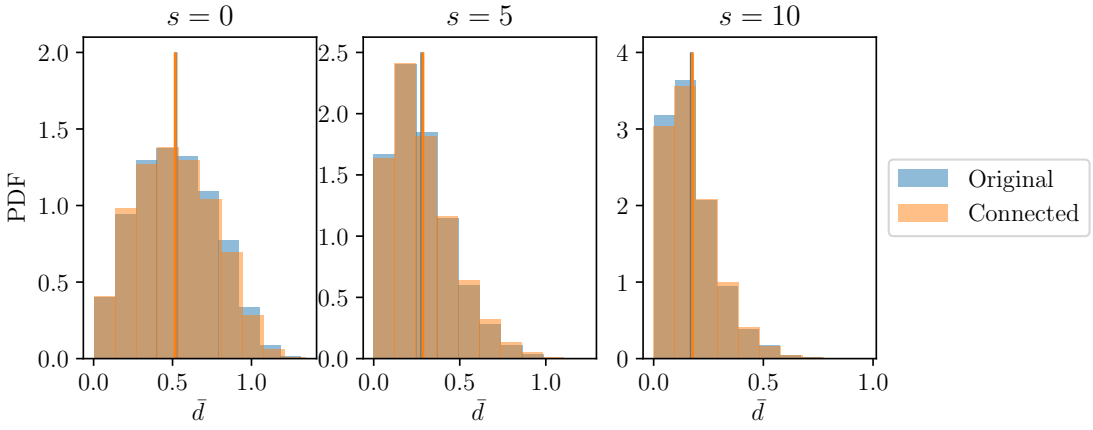


(b) $n = 100$.

Figure 4.6: Relationship between average edge length, $\bar{d}$ and parameter s. $z = 2$ shown. As z increases the probability of connectedness decreases and so the discrepancy decreases for all n (not shown).



(a) $n = 10$.



(b) $n = 100$.

Figure 4.7: Empirical edge length distribution for $s = [0, 5, 10]$. The original ensemble is in blue and the connected ensemble in orange. The average of each ensemble is shown by the solid vertical line.

We consider the difference in the expected value of the edge length $\bar{d}$ theoretically. Let us first consider fixed node positions, so the probability of a distance d_{ij} between two nodes is equivalent to the probability that link exists. i.e., $P(d = d_{ij}) = P(e_{ij} \text{ exists})$. The expected value of the distance in a Waxman graph with parameters s and q is,

$$E[d|s,q] = \sum_{ij} e_{ij} P(e_{ij} \text{ exists}|s,q). \tag{4.7}$$

Using Bayes' rule we can write the expected value of the distance in a connected graph

$$E[d|s,q,c] = \sum_{ij} e_{ij} \frac{P(c|e_{ij} \text{ exists}, s, q) P(e_{ij} \text{ exists}|s,q)}{P(c|s,q)}, \tag{4.8}$$

where c is an indicator of connectedness. Rewriting the fraction gives,

$$E[d|s,q,c] = \sum_{ij} e_{ij} \frac{P(c|e_{ij} \text{ exists}, s, q)}{P(c|s,q)} P(e_{ij} \text{ exists}|s,q). \tag{4.9}$$

The difference between the connected case and the original case is a factor of $P(c|e_{ij} \text{ exists}, s, q)/P(c|s,q)$ for each edge. The numerator here gives us the probability that the graph is connected given a certain edge exists in the graph. The existence of the edge will increase the probability of connectedness. Additionally, the existence of some edges, for example a long edge connecting two clusters of nodes, will increase the probability of connectedness to a greater extent than others. For large n the probability of connectedness will depend less on the existence of a single link, so $P(c|e_{ij} \text{ exists}, s, q) \approx P(c|s,q)$. Hence, we see little difference between the lines in Figure 4.6b. Conversely, for small n the existence of a single link can impact connectedness, and so we see divergence in Figure 4.6a. Similar arguments can be made for the impact of increasing z and s on the expected value of d.

We can make a similar argument using the edge length distributions. In the connected case,

$$f(d|s,q,c) = \frac{f(d|s,q) f(c|d,s,q)}{f(c|s,q)} \tag{4.10}$$

where c is an indicator of connectedness. This differs from the original case by a factor of $f(c|d,s,q)/f(c|s,q)$. We can take the expected value and interpret it as above for general node placements and explain the distributional differences in Figure 4.7.

4.4 Conclusion

This chapter has presented extended approaches for sampling random graphs conditioned on a particular property. Specifically, here, we have highlighted the computational improvements gained in sparse graphs by using an asymmetric TNT proposal that preferentially targets edge removals. We also introduced the possibility of a likelihood annealed sampler that has application when it is difficult to generate a graph from the ensemble of interest. Future work here includes designing an efficient annealing schedule and comparing computation time and complexity to the single chain samplers. Finally, we considered the impact that connectedness has on the Waxman ensemble, particularly the parameters that are estimated when fitting these models to real world networks. We have found that there is very little difference in the relationship between graph parameters in the connected ensemble for larger graphs; however, for modelling small networks one should consider an estimator that accounts for the bias created by conditioning on connectedness.

We have developed methods to sample networks of the form $P(G|G \text{ connected})$, or, more generally, $P(G|\sigma(G))$, where $\sigma(G)$ is some set of constraints on G. A natural question to consider next is what other properties of the graph can we consider and how can we use the samples from the distribution? Using the ideas developed in this work we can consider the inference of networks where the property of interest is some relationship between our graphs of interest and observed data. In the next section we highlight the applicability of Bayesian methods in network inference from the observation of information cascades over a network. In this case we condition on the networks that can facilitate the cascade set C, $P(G|G \text{ facilitates } C)$, and use the samples to infer the overall network structure.

Chapter 5

GraphMCMC: network inference from information cascades

This chapter contains contents from the publication:

> Gray C., Mitchell L., and Roughan M. Bayesian Inference of Network Structure From Information Cascades. *IEEE Transactions on Signal and Information Processing over Networks*, 6 (2020) pp. 371-381.

Please see the appendix for full authorship statements.

Contagion phenomena, whether information flow, cascading power grid failures or disease epidemics, are inherently linked to the network structure on which they propagate. Often, we observe dynamics without knowledge of the underlying network structure; either because it is unobservable or because the dynamics are easier or cheaper to record than the underlying network. For example, it may be difficult to observe social structure because connections, like friendships, may be invisible or ill-defined. However, dynamics on these networks often are observable; for example, people posting the same 'meme', buying the same product or sharing the same link. This spatio-temporal information is inherently linked to the underlying network structure and can be used to reconstruct and predict paths of information flow. Reconstruction of the network from these observations can help to understand how network structure interacts with the dynamics and to predict or control the information flow.

Here, we address the problem of inferring the connectivity structure between nodes (network structure) from observations of information cascades, accounting for the inherent uncertainties that arise in inference and data observations. To achieve

this we utilise Bayesian methods that quantify uncertainty. Here, we focus on information cascades as the dynamic process that is observed on the network. However, the same ideas can be applied to other types of dynamics over networks, e.g., disease outbreaks in epidemiology.

There are many sources of uncertainty in social networks. These networks are often dynamic and difficult to observe, and there has been significant effort to improve network estimates from noisy and missing data [18, 46, 76, 77, 101]. The use of indirect observations, such as dynamics on the network, increases the uncertainty in the network estimates. When we can only achieve an estimate of the network through the lens of the cascade processes it is important to consider the uncertainty in the cascade dynamics, e.g., missing data, incorrect time recordings, as well as heterogeneity in observability.

Probabilistic techniques are ideal in this context as they explicitly include such uncertainties. We develop a probabilistic approach to the network inference problem of learning the underlying network structure from information cascades.

We begin with the well-known independent cascade model of information propagation [74] and develop a Bayesian inference method based on Markov Chain Monte Carlo (MCMC) to infer links between individuals. We seek a distribution of graphs that can facilitate the observed cascades to not only find good estimates of the underlying graph, but also use the knowledge of the underlying distribution to facilitate informed decision making and promote further understanding of the uncertainties in these processes.

The method samples from $P(G|C)$: the distribution of networks $P(G)$ conditioned on the observation of a set of cascades C. Point estimates and associated uncertainty about the existence of edges can then be obtained. By quantifying the uncertainty, estimates can be obtained with small amounts of data, and previously known information about edges can be easily incorporated.

We demonstrate the method using the independent cascade model; however, it can easily be extended to any cascade model where the likelihood of a cascade can be evaluated for a given network. We show that Bayesian techniques can produce estimates and associated uncertainty with a small number of cascades where other methods, such as NETINF [55] can not even produce a result.

The main contribution of this chapter is the development of a probabilistic technique to sample from the distribution of underlying graphs. The resulting inference of the network structure includes uncertainty quantification that can be used to inform decisions and understand the interaction between spreading process and network structures.

5.1 Background and related work

The network inference problem is an active area of research that aims to infer the links or transmission probabilities using both model and model-free inference. Only recently have the uncertainties in these types of estimates begun to be quantified.

There is a significant body of work on inferring information about the underlying network structure from observations of information cascades. These methods are largely based in a maximum likelihood estimation (MLE) or expectation maximisation (EM) frameworks with a variety of optimisation strategies employed to find a good estimate of the network structure. However, most of these models fail to incorporate uncertainty quantification, which can result in misleading results.

A seminal work on network inference is the well known NETINF algorithm that uses submodular optimisation to infer the underlying graph structure. Numerous extensions incorporate prior information about the underlying graph structure such as sparsity, motif frequency, community structure, *etc* [66, 113, 132]. Many of these are ad-hoc extensions to include prior information. Additionally, algorithms have been developed to infer the strength of connections in a network [12, 54], understanding the heterogeneous influence of edges on different topics [66, 86, 140] and derive bounds on the reliability of the methods [98].

We require a general probabilistic framework for inference that incorporates uncertainty. Bayesian methods can provide this framework to infer distributions and quantify uncertainty, as well as include prior information about the network structure and cascade process.

Bayesian inference of networks from observation of network dynamics emerged in the stochastic epidemic literature to recover parameters of underlying network models, e.g., the underlying parameter p of an ER random graph, in addition to the epidemiological model parameters [14, 42]. More recently, the benefits of Bayesian techniques to quantify uncertainty in networks, often due to collection methods, have been identified [101, 107]. This extends naturally to the use of these methods in the estimation of the network structure and its properties from the observation of dynamics on the network.

Embar *et al.* [43] propose a Bayesian framework for estimating properties of the network, e.g., edge strengths as well as cascade properties, e.g., propagation trees. Other Bayesian approaches for network inference in disease prediction aim to infer the underlying network when the exact infection time is unknown in both standard and online algorithms [84, 125, 126]. Additionally, some approaches based on MLE use Bayesian techniques at intermediate steps to improve results [25]. Only

recently have Bayesian techniques been introduced to quantify the uncertainty of network structure inference from observation of dynamics [34,49,108,136]. Ghalebi *et al.* [49] highlight the need for general probabilistic frameworks for inference problems and propose the algorithm DYFERENCE that samples edge and node probabilities in an online algorithm for dynamic network inference. Peixoto [108] recently proposed the most closely related work in an algorithm designed to jointly reconstruct network structure and community labels assuming a stochastic block model structure and highlights the benefits of recovering the full posterior distribution of networks.

It is worth noting that Bayesian methods are extensively used in sampling exponential random graph models (ERGMs) and fitting coefficients of motifs to observed networks [19, 22, 87]. Fitting ERGMs can be extended to the network inference problem by observing information cascades instead of observing an existing network to infer ERGM coefficients.

5.2 GraphMCMC for network inference

The network inference problem aims to learn the structure of an underlying network from the observation of transmissions between individuals (nodes) in the network. A set of these transmissions is a cascade and could be an information cascade, where individuals are transmitting information, or an infection cascade, where individuals are contracting disease from their contact networks. We have an underlying graph $G = (N, E)$ with node set N and edge set E, where the number of nodes is denoted n. On a given graph G, a *cascade* is a set of node time pairs where a node $N_i \in N$ becomes 'activated', by one of its neighbours j, where $(i, j) \in E$ at time t_i. We observe a sequence of activated nodes and their activation times. However, we do not observe the parent of the transmission; that is, we do not observe who infected node i, only the time they were infected.

This leads to the following problem. Given a set of cascades $C = c_1, c_2, ..., c_{|C|}$, where $c_i = \{N_1 : t_1, N_2 : t_2, ...\}_i$ is the infection node/time pairs in cascade i, what is the underlying graph G on which the process was observed?

Here we are interested in the distribution over all graphs that could have produced the observed cascades C. That is, we are interested in recovering $P(G|C)$. Directly recovering this distribution is infeasible for graphs with more than a handful of nodes as the dimension of the space increases exponentially. Instead, we draw samples from this distribution using Bayesian methods and use these to recover the most probable network structures and a measure of their probability.

5.2.1 The independent cascade model

We use the independent cascade (IC) model of information cascades [74] on the edges of G, analogous to the SI model in epidemiology. The IC model assumes that every node u independently infects its neighbour v with some probability β of success. Originally proposed as a discrete time model, we use the continuous time extension [55]. Each node u attempts infection of inactive neighbour v after some time $\Delta_{u,v}$, so if the activation is successful $t_v = t_u + \Delta_{u,v}$. Additionally, upon successful activation node u becomes the parent of v, denoted as $\rho_v = u$.

We define $P(T_v = t_v | c^{(t_u)}, \rho_v = u)$ to be the probability density of T_v, the time node v will be activated, conditioned on the cascade activation times up until t_u, denoted $c^{(t_u)}$, and node u being the parent of v. We assume here that this depends only on the times of activations of the two nodes. However, this can be altered to include other characteristics like the degree of nodes, the importance of friendship (u, v), properties of the node or the content of the message being spread. There is evidence of both exponential and power-law distributed waiting times for contagion spread [3,88,91] and either can be incorporated in the proposed method. We demonstrate the method using the exponential distribution for the time between activations:

$$P(t_v | c^{(t_u)}, \rho_v = u) \propto \exp\left(\frac{-(t_v - t_u)\mathbb{1}(t_v > t_u)}{\lambda}\right), \tag{5.1}$$

for some parameter λ. For simplicity we use $\lambda = 1$ for synthetic datasets. For real world datasets, we can attempt to estimate λ from the observed data.

The model assumes that only one node u activates node v despite the possibility of multiple active neighbours. Therefore, the resulting cascades are trees. We choose the continuous time independent cascade model to demonstrate the MCMC method as it is simple, well studied and captures much of the dynamics of information flow. However, our method can be extended to other transmission models, such as epidemiological models, discrete time IC models, Hawkes process models [65] or other models where the likelihood can be evaluated.

5.2.2 Likelihood

We require the likelihood of the set of observed cascades $C = \{c_1, \ldots, c_{|C|}\}$ given some underlying graph G. We use a similar derivation as in [55]. First let us consider the likelihood of a single cascade c. We assume that cascades propagate via a tree so each node will have a single parent for a given cascade. Each possible

tree is a disjoint outcome for the cascade on the graph, so,

$$P(c|G) = \sum_{T \in \mathcal{T}(G)} P(c, T|G), \tag{5.2}$$

where $\mathcal{T}(G)$ is the set of all possible trees and $P(c, T|G)$ is the probability cascade c travels through tree T on graph G.

Consider the likelihood that a single cascade c in graph $G = (N, E)$ propagated in the tree $T = (N_T, E_T)$. At each edge the cascade propagated with probability β and stopped with probability $1 - \beta$, giving:

$$\begin{aligned} P(c, T|G) &= P(T|G)P(c|T, G) \\ &= \prod_{(u,v)\in E_T} \beta \prod_{u\in V,(u,x)\in E\setminus E_T} (1-\beta) \times \prod_{(u,v)\in E_T} P(t_v|c^{(t_u)}, \rho_v = u), \\ &= \beta^y (1-\beta)^r \times \prod_{(u,v)\in E_T} P(t_v|c^{(t_u)}, \rho_v = u), \end{aligned} \tag{5.3}$$

where y is the number of edges over which the cascade propagated, $y = |E_T| = N_T - 1$, and r is the number of edges in the graph that the cascade did not pass through. Mathematically, $r = \left(\sum_{u\in N_T} z_{out}(u)\right) - y$, where $z_{out}(u)$ is the out-degree of node u. Note the use of the indicator function to ensure that $P(c|T, G) = 0$ when the tree violates the cascade activation order.

Now consider the cascade over all possible trees on the graph G. We substitute (5.3) into (5.2), to give

$$P(c|G) = \sum_{T\in\mathcal{T}(G_c)} \beta^y (1-\beta)^r \times \prod_{(u,v)\in E_T} P(t_v|c^{(t_u)}, \rho_v = u). \tag{5.4}$$

where G_c is the subgraph of G induced by the nodes in cascade c and $\mathcal{T}(G_c)$ is the set of all possible trees on this graph. Note that in general we sum over trees in G, but if the tree is inconsistent with the observed data then $P(c|T, G)$ is zero, and hence we can sum over the trees in the set $\mathcal{T}(G_c)$.

Each tree in $\mathcal{T}_c(G)$ contains exactly the nodes activated in c, so y and r are independent of T, and so

$$P(c|G) = \beta^y (1-\beta)^r \sum_{T\in\mathcal{T}(G_c)} \prod_{(u,v)\in E_T} P(t_v|c^{(t_u)}, \rho_v = u). \tag{5.5}$$

For a set of independent cascades C occurring on G we have

$$P(C|G) = \prod_{c\in C} P(c|G). \tag{5.6}$$

Equation 5.5 would naively require the sum over all possible spanning trees, which can be super-exponential in the size of G. However, we are only interested in the directed trees over G_c, the subgraph of G induced by the set of nodes in cascade c. The links in this subgraph are directed according to increasing times in the cascade.

To calculate the sum required in Equation 5.5, as suggested in [55], we utilise Kirchoff's matrix tree theorem, extended to directed trees by Tutte [134]. We provide the theorem below, which uses the following definition.

Definition 1 (Laplacian of directed multigraphs) *If $H(N, E)$ is a weighted directed graph, we define the Laplacian L of H as an $n \times n$ matrix with entries:*

$$L(H) = \begin{cases} \sum_k w_{k,i} & \text{if } i = j, \\ -w_{i,j} & \text{if } i \neq j \ \& \ (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

where w_{ij} are the edge weights in H — these could be the binary adjacencies.

Theorem 3 (Tutte [134]) *If graph H has Laplacian $L(H)$ as defined in Definition 1, then the sum over the weighted trees of graph G with root at node r is*

$$\sum_{T \in L(H)} \prod_{(k,l) \in T} w_{k,l} = \det\ (L(H)_r), \tag{5.8}$$

where T is each directed spanning tree in H and $L(H)_r$ is created by removing the r-th (root node) row and column from $L(H)$.

In our formulation we apply this theorem to G_c and set $w_{i,j}$ to be the likelihood of that link in the infection tree: $P(t_v | c^{(t_u)}, \rho_v = u)$. As G_c is directed by the direction of potential infection it is a directed acyclic graph. Therefore, the nodes can be ordered such that the Laplacian is upper triangular and the determinant is a product of the diagonals. Conceptually, as the parent of each vertex can be any node activated at a previous time, the sum over trees for each cascade is the product over the nodes of the number of possible parents. This aligns with the theorem used. This simplifies calculation time and storage from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ as we only work with the diagonal elements. This is a convenient property of the continuous time model that allows infection in only one direction, and does not apply for other models that may have cycles. For example, in the discrete time model when infections can happen simultaneously or if nodes have multiple activations, the Laplacian can be non-triangular.

5.2.3 Inference

Given a set of cascades C — a set of lists of activation times of labelled nodes in N we infer a distribution of graphs conditioned on the occurrence of these cascades. Mathematically, given the cascade set C we sample from the distribution $P(G|C)$.

From (5.6), we have the cascade probability from a graph $P(C|G)$. From Bayes' rule we can get the distribution of interest:

$$P(G|C) = \frac{P(C|G)P(G)}{P(C)}. \tag{5.9}$$

We use the basic Metropolis-Hastings MCMC algorithm to sample from this posterior distribution. At each step we propose a new graph G' from the old graph G using the proposal distribution $Q(G'|G)$. The algorithm accepts the move step based on the ratio between the cascade on the new graph and the old graph. That is, with probability:

$$\alpha \;=\; \min\left(1, \frac{Q(G|G')P(G')}{Q(G'|G)P(G)} \prod_{c\in C} \frac{P(c|G')}{P(c|G)}\right). \tag{5.10}$$

For a single cascade c, we use Theorem 3 and (5.5) to get

$$\frac{P(c|G')}{P(c|G)} = \beta^{y'-y}(1-\beta)^{r'-r}\frac{\det(L(G')_{n_1})}{\det(L(G)_{n_1})}. \tag{5.11}$$

Suppose the proposal changes one link (i, j) from G to G'. The number of nodes in the spanning trees $|N_T|$ remains constant, so $y' = y$. Assuming $t_i < t_j$, if node i is in the cascade $z_{out}(i)$ increases when adding a link, and decreases when removing.

$$r' - r = \begin{cases} +1, & \text{if } i \text{ in } c \;\&\; \text{adding edge,} \\ -1, & \text{if } i \text{ in } c \;\&\; \text{removing edge,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

This term penalises the addition of edges in the cascade. This not only promotes the desired sparse networks but also reduces the probability of edges that didn't propagate the cascade as such edges are less likely. The resulting algorithm is described in Algorithm 5.1.

To initialise the algorithm we require an initial graph with $P(G|C) > 0$. Step 1 (in Alg. 5.1) generates the initial graph and associated probability. Here, we generate the initial graph by connecting each time ordered pair in C; however, any graph

Input: $C = c_1, c_2, \cdots c_{|C|}$
Generate $G^{(0)}$
for $t = 1, \ldots, K$ **do**
Generate $G' \sim Q(G'|G^{(t-1)})$
Take $G^{(t)} = \begin{cases} G', & \text{with probability } \alpha \\ G^{(t-1)}, & \text{with probability } 1-\alpha. \end{cases}$
where $\alpha = \min\left(1, \frac{P(C|G')P(G')Q(\theta|\theta')}{P(C|G)P(G)Q(\theta'|\theta)}\right)$
end for

Algorithm 5.1: Bayesian inference of network G given cascade set C where K is the number of iterations of the algorithm.

of non zero probability could be used. The complexity of finding the probability of this initial graph is $\mathcal{O}(e_{init})$ where e_{init} is the number of edges of the initial graph.

A common proposal used when implementing MCMC on networks is to change a random node pair (i.e., creating a new edge or removing old edges) [22, 58]. However, in sparse graphs non-edges are proposed much more often than edges, and the sampler wastes time proposing new edges that are likely to be rejected. In order to improve mixing, the 'tie no-tie' (TNT) sampler is often used [22, 69, 87], as described in Chapter 4. The TNT sampler selects with equal probability the set of edges or set of non-edges, and then changes a random node-pair in that set. This improves convergence and aids mixing by proposing edge removals at a higher frequency, particularly when we begin our inference from a graph denser than our desired ensemble. Additionally, trying to remove edges often will explore different transmission pathways that rely on different links in the network. We utilise the TNT sampler in the implementation of our algorithm. Other proposals, for example edge flips and switches [33] can also be used.

For the subsequent analysis we will assume β is known and we later show that the network recovery is not sensitive to this choice. Alternatively we could include β in the inference by assuming a prior and determining the posterior estimate through a Gibbs' sampling step. Unfortunately, the sparsity of the inferred network has dependence on both β and p, and untangling these two parameters independently is difficult under this model. These parameters could be estimated separately using other data, perhaps by sub-sampling a small section of the network or observing similar cascades in a known area of the network. Nonetheless, we show that, for inference, results are not greatly affected by the chosen parameters.

We place an Erdős-Rényi prior with parameter p with independent edges on G, so $P(G')/P(G)$ is easily calculated. We assume p is known (or can be estimated) and

consider a sensitivity analysis in Section 5.4.1. This prior serves as a regularisation term to promote sparse solutions.

5.2.4 Complexity and convergence

The MCMC algorithm has two main factors contributing to running time and complexity: (i) the computation required to propose and update each iteration and (ii) the number of iterations required to converge to the posterior of interest.

At each iteration we propose a new graph. Each iteration, the method requires $\mathcal{O}(1)$ operations of choosing to add or remove and choosing the associated dyads. Updating the determinant is at worst $\mathcal{O}(|C|)$ (if the node is in all cascades). Overall the complexity is $\mathcal{O}(K|C|)$, where K is the number of iterations required for convergence. To determine an appropriate number of iterations for convergence traces of the parameters are often used. Theoretically, convergence is guaranteed in infinite time and in line with other work, [22, 87], we show that convergence is reasonable in the MCMC method with the TNT sampler. We utilise traces of the log likelihood and a summary statistic of the graph to give evidence for convergence to the posterior. After convergence we expect no change in the distribution of the summary statistics as the networks are being drawn from the same underlying distribution. In general, we find that convergence is $\mathcal{O}(e)$, where e is the number of edges in the graph, as highlighted in Chapter 4. Figure 5.1 shows the trace of the log likelihood (black) and average degree (blue) for a single chain. We show the average degree of the network as this provides evidence of the number of edges converging; however other summaries (e.g., average clustering or average shortest path) show the same trends. Autocorrelation plots also provide evidence for convergence (usually < 0.4 is acceptable) and provide information on whether thinning is required to reduce the impact of correlated samples on the variance of the estimator. Figure 5.1 shows the autocorrelation plots for the log likelihood. In general, we conservatively use $50n^2$ iterations for burn-in.

5.3 Experimental evaluation

We test our algorithm on synthetic data to evaluate the effect the amount of data and the underlying network type has on performance, and extend this to inference of real networks. Next we describe the synthetic experimental setup.

It is known through work on disease and information cascades that the structure of the underlying graph affects the cascades observed, even when using the same

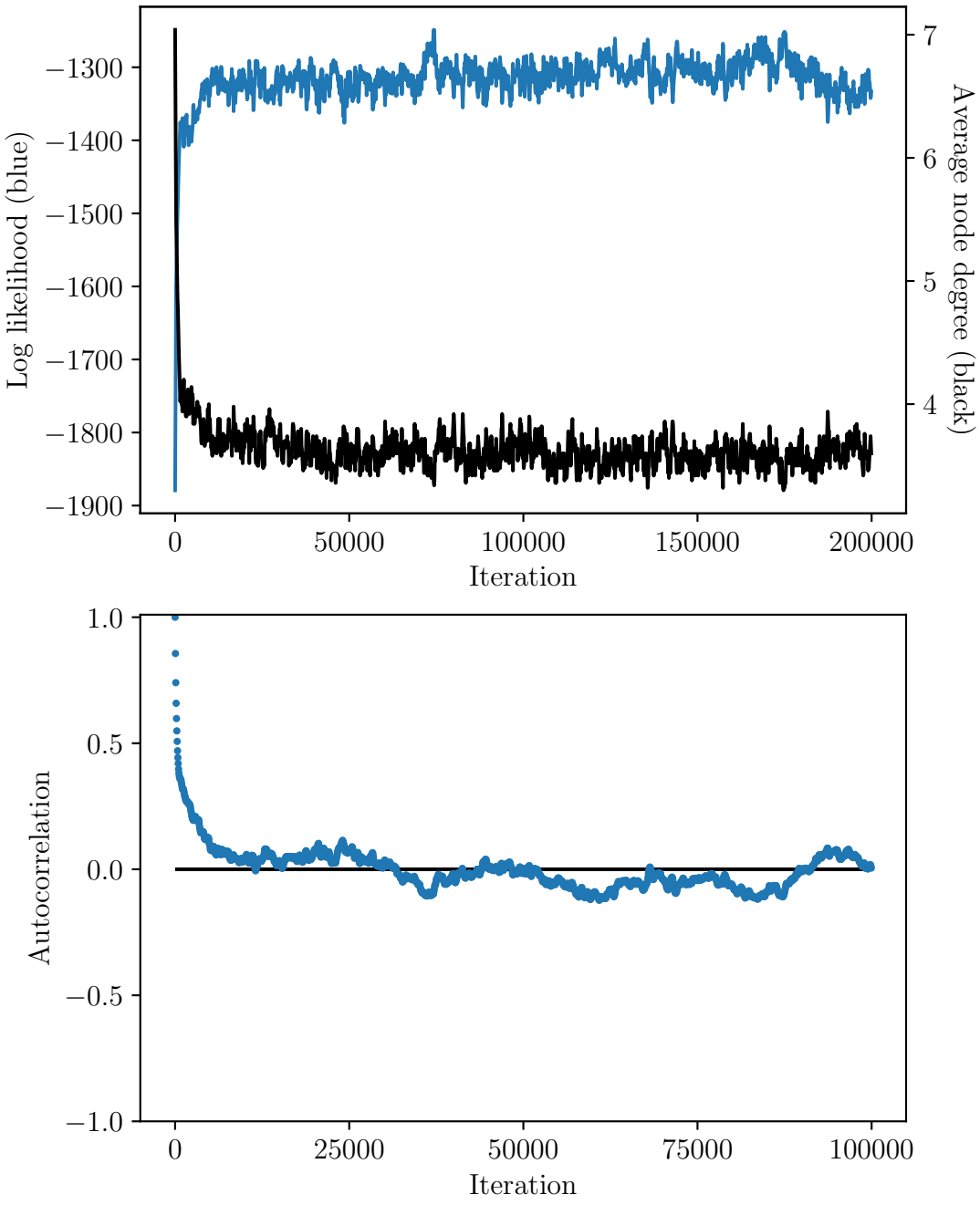


Figure 5.1: Top: Trace of log likelihood (blue) and average node degree (black) during the MCMC process for an $n = 100$ network. Bottom: Autocorrelation of the log likelihood quantities over a single chain.

cascade model [57, 141]. Therefore, we use a variety of undirected random graph models to represent the underlying graph G. We begin with the canonical Erdős-Rényi (ER) networks and then extend to more realistic networks with power-law distributions using the forest fire model [81] and core-periphery and hierarchical structures [30] using Kronecker networks [80].

Cascades are simulated using the IC model with exponential waiting time ($\alpha = 1$). Intuitively, as the amount of data increases, i.e., we observe more transmissions or cascades, the inference problem becomes easier. We quantify the amount of data observed as the fraction, f, of the edges in G that are activated in the cascades, i.e., if all edges are activated at least once $f = 1$. This metric is inherently linked to the number of cascades observed, but ensures we do not over count cascades that do not provide new information. In most simulations we choose $\beta = 0.4$, to align with other works, including NETINF, and to ensure cascades are not too small, resulting in many cascades that have zero or one transmission, and not unrealistically large. We select random seed nodes uniformly over the graph to ensure we have good coverage of the network even when the amount of cascade data is small.

The proposed method provides samples from the distribution of $P(G|C)$, and so a key advantage of this method is the ability to use these to quantify uncertainty and answer questions about the underlying distribution. Other methods provide a point estimate, $\hat{G}$, of the underlying graph. Despite the limitations of a simple point estimate such as this, obtaining an estimate from our posterior is useful, at least for comparison. There are multiple approaches we can use to derive such an estimate.

We would like to be able to get some estimate of the graph as well as the uncertainty and visualise the results in order to gain some understanding about the underlying graphs, and compare to other methods. Therefore, to evaluate the solution quality we recover the marginal probabilities of the network edges, $P((i, j) \in E|C)$. This can both provide important uncertainty quantification, and be used as a measure to determine how well this approximates the true network.

Edges with high marginal probability are more likely to have been part of the true graph G^*. We present these marginals in a weighted adjacency matrix (Figure 5.2) and the most probable edges are used to create an estimate of the graph.[1]

The probabilistic nature of the inference provides the opportunity to use skill scores and probabilistic scoring techniques to compare the recovered distribution to the true network. This is qualitatively different to most other approaches taken

[1]If there is prior knowledge of the underlying network, such as the degree distribution, we could find the most probable edge set satisfying these constraints.

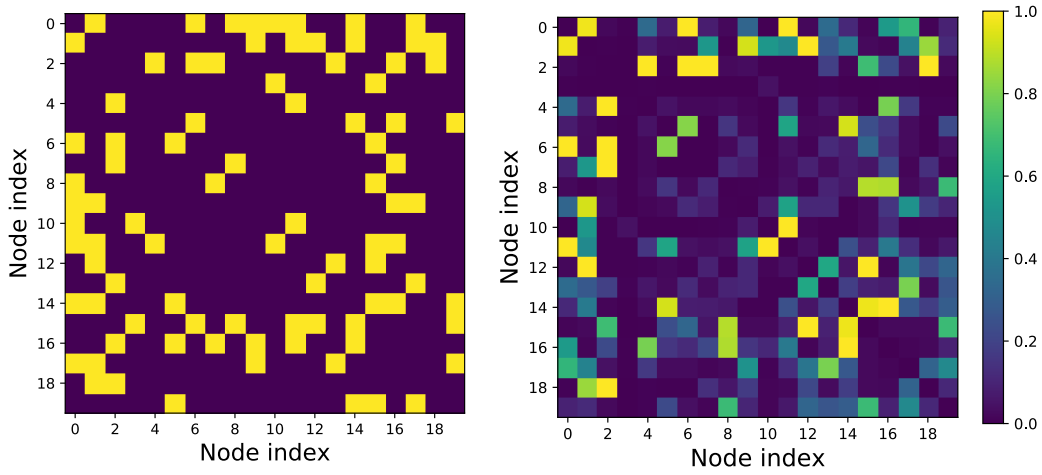


Figure 5.2: Left: Visual representation of the ground truth adjacency matrix where yellow at index (i, j) indicates edge (i, j) exists in the graph. Right: Weighted matrix of estimated posterior edge probabilities.

previously for this problem, which treat every problem as a 'hard' classification of edge existence. Hence, in order to compare with other techniques we use the Receiver Operating Characteristic (ROC) curves that report the true positive rate and false positive rate of the method as the probability threshold varies. For example, Figure 5.3 (left) shows the ROC curve for the posterior marginal probabilities for an undirected 20 node network. The ROC highlights the diagnostic ability of the algorithm when estimating the binary edge presence. Each network estimation will produce a single ROC curve and we would like to compare these curves for different parameters and networks. Therefore, we use two summaries of the ROC curves; the area under the curve (AUC) and we define the 'false positive alarm' to demonstrate the recovery of the true edges in larger scenarios.

The AUC provides an aggregate measure of classification performance across all thresholds, and is measured as the integral of the empirical ROC curve between 0 and 1. The false positive alarm measures the proportion of true edges correctly recovered before there are too many incorrect edges. In many situations false positives are costly, either financially, e.g., the cost of investigating a false claim, or ethically, e.g., incorrect convictions. Understanding the uncertainty in the estimate of the network, or probability of the edge, allows better decision making and application specific requirements on certainty. In this case we determine the true positive rate when 1% of the edges recovered are false positives. In general the AUC provides an overall summary of the ROC curve while the false positive alarm focuses on the bottom left corner of the curve.

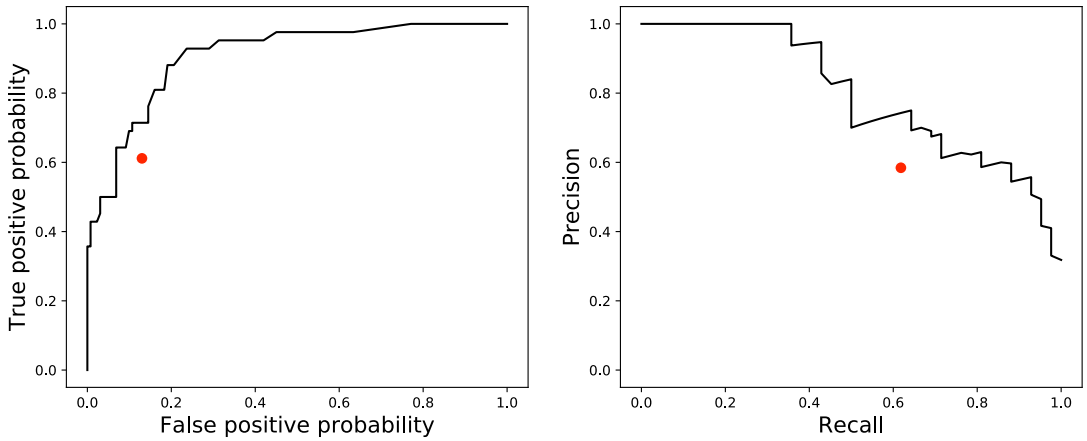


Figure 5.3: Left: ROC curve of recovered edges from the posterior edge marginal distributions. Right: Precision-recall curve of recovered edges from the posterior edge marginal distributions. The MAP estimate point is shown in red.

Figure 5.3 (right) shows the corresponding precision-recall curve. We summarise these curves in a similar way, using the average precision (AP) which is almost always equivalent to the area under the precision-recall curve [37, 106]. The maximum *a posteriori* probability (MAP) is a Bayesian point estimator of the network and can be extracted from the posterior distribution. We show the MAP points on the ROC and PR curves in Figure 5.3. In general, due to the high dimensional posterior, we cannot guarantee that the MAP estimate of the graph in our posterior is similar to the true maximum.

5.4 Synthetic data results

We begin by simulating cascades on synthetic networks generated from a variety of network models to investigate the impact of the amount of data and network type on performance.

Dependence on number of cascades

It is intuitive that inferring the network becomes an easier problem when we have observed more data; however, in many scenarios data is limited. Here we investigate the performance of the MCMC inference methods with varying amounts of data. Recall that the amount of data is quantified as the fraction, f, of the

edges that are activated in the cascades, rather than the number of cascades. This reduces the dependence on the size of individual cascades. We focus on the limited data regime where f is between 0.3 and 1, as this is a more realistic scenario in practice and highlights the importance of uncertainty quantification, in comparison with other methods that consider much more data.

We begin with Erdős-Rényi graphs with $n = 100$ and $n = 1000$ with average degree $z = 4$. We simulate cascades to infer 10 networks and report average and 95% confidence intervals. We take summaries from the ROC curves of the edge probabilities extracted from the posterior samples. NETINF requires the number of edges e in the resulting graph as an input. To create a comparative ROC curve, we report the true and false positive rates (TPR and FPR) for increasing e and determine ROC summaries from this.

Figure 5.4 shows the ROC summaries for the MCMC algorithm and NETINF for varying amounts of observed data for $n = 100$ (top) and $n = 1000$ (bottom). We observe that the MCMC algorithm performs well with relatively small amounts of data. This is beneficial in many cases where data may be scarce. When data is limited, the greedy algorithm used in NETINF does not return the requested number of edges and so performs poorly. Understandably, most algorithms perform poorly under the false positive alarm for small amounts of data; however, when all edges are observed in at least one cascade ($f = 0.99$) the MCMC produces a good false alarm rate of over 40%.

In practice, the underlying network is not known, and metrics like the false alarm rate cannot be observed. Therefore, it is difficult to choose how many edges are reliable in the recovered network. Using the MCMC method, unlike most other methods, we have quantified the uncertainty of each edge to inform such decisions when the underlying network is unknown.

We also report the average precision (AP) of the precision-recall curves in Figure 5.5 for further comparison to NETINF. As expected, the AP is lower than the AUC as it focuses on prediction of the positive class and does not reward the correct prediction of a non-edge. However, our method reports higher precision than NETINF across all data values.

Summarising the samples from the posterior distribution using the marginals in this way is losing some amount of information. One can imagine other ways to utilise the samples from the posterior for various applications that do not include the marginal distributions, and these present interesting areas of further research. For example, extracting most common paths, important nodes, or pathways that are 'over' or 'under' represented in the posterior [79].

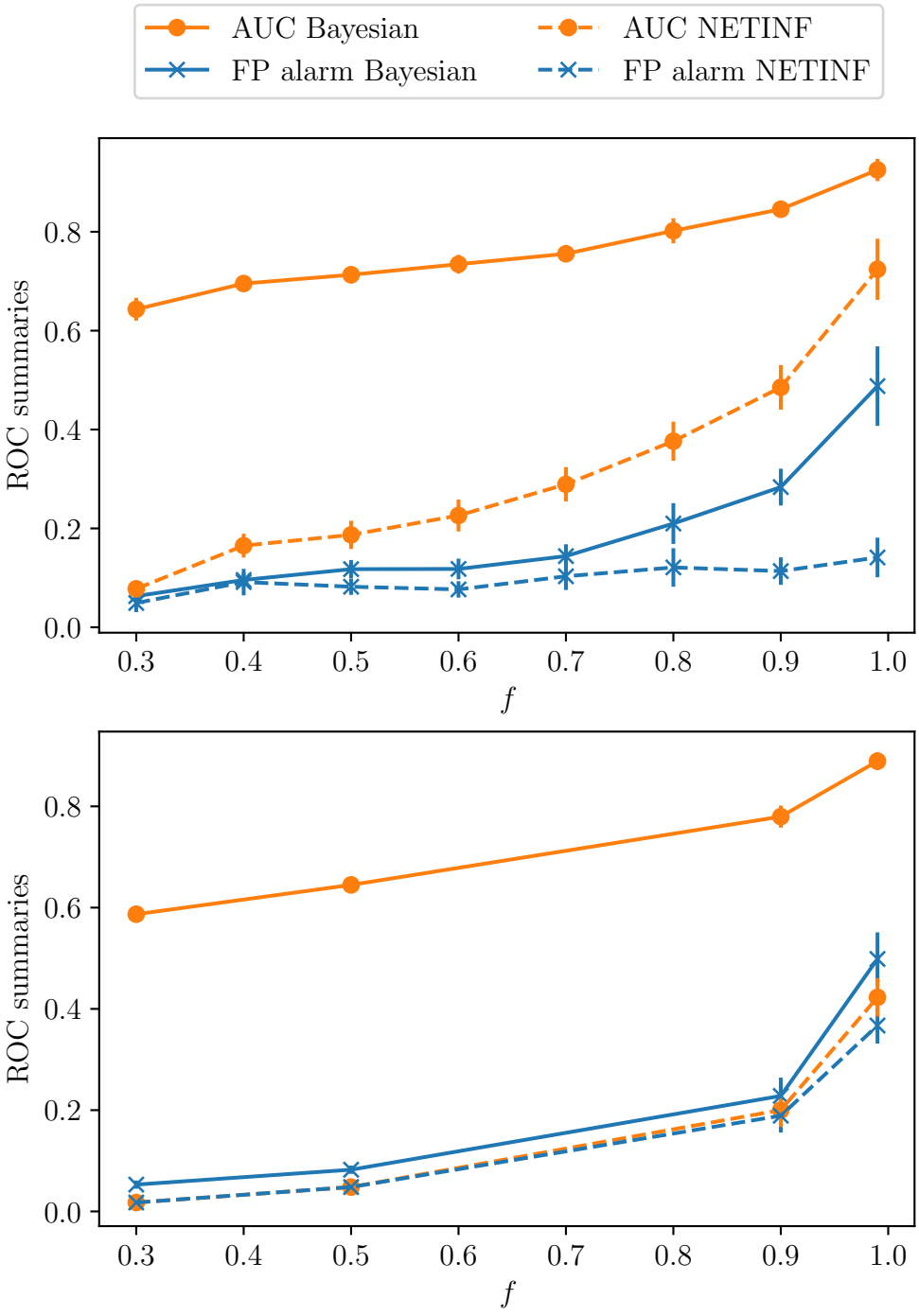


Figure 5.4: ROC summary metrics for varying amounts of data on networks with $n = 100$ (top) and $n = 1000$ (bottom). Our method (solid lines) and NETINF (dashed) after observing cascades that cover a proportion f of the edges. Cascades were simulated until a proportion f of the edges were activated. The orange curves (dots) show AUC and the blue (crosses) show false positive rate (blue crosses). 95% confidence intervals are shown for inference over 10 different networks.

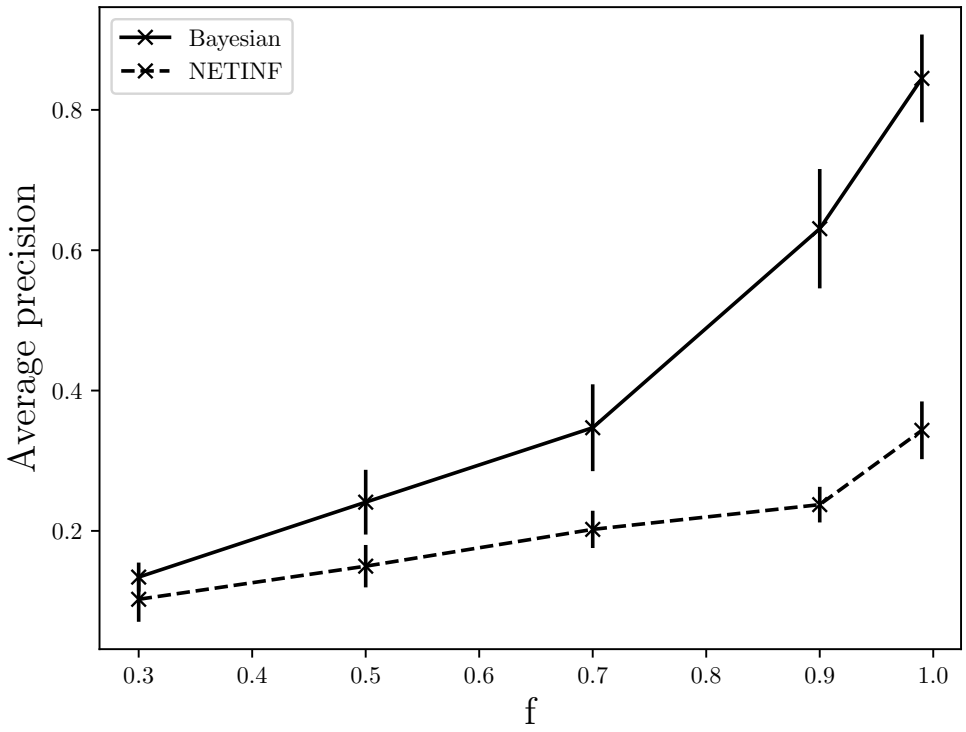


Figure 5.5: Average precision (AP) for varying amounts of data. Our method (solid lines) and NETINF (dashed) after observing cascades that cover a proportion f of the edges. Cascades were simulated until a proportion f of the edges were activated. 95% confidence intervals are shown of inference over 10 different $n = 100$ networks.

Types of networks

We test the performance of our inference scheme on different types of networks that display a variety of properties observed in real world networks. In this work we use an Erdős-Rényi prior that assumes links have an independent probability p to control the sparsity of the solution. We show that despite this the algorithm can infer networks of varying types as the data informs the structure of the underlying graph. We assume some prior knowledge of the edge density to choose p, but we show that the exact value has little effect on the network recovery. In some cases we may have some prior knowledge of the underlying structure, perhaps the degree distribution, density or frequency of some motif, which can be incorporated in an informative prior to improve inference.

There are considerable differences in the recoverability of networks of different types. This is largely due to the nature of the cascades observed. It is well known that the underlying structure impacts the nature of the cascades simulated using various cascade models [57, 74]. The underlying structure of the network has a dramatic impact on the size distribution of cascades we observe and this has a marked impact on the recoverability of the networks. We simulate cascades until we observe a constant fraction $f = 0.9$ of edges activated in each case and show the results in Table 5.1. There is low variance in the recoverability of different realisations of the same graph type. We find that the average cascade size, strongly influenced by the network type and β, can impact the recoverability. The average size and number of cascades required to observe 90% of edges activated changes between networks. For Erdős-Rényi and core-periphery networks it is more likely that we observe fewer larger cascades that each cover many edges; compared to the forest fire and hierarchical networks in which large cascades are unlikely and so we observe many small cascades. Smaller cascades provide more precise information about edges as there are less possible paths the cascade could take, and so we see improved inference on networks that facilitate smaller cascades sizes. The AUC and average precision score (AP) are low using both methods in networks that are dominated by a few large cascades. The signal in the data in these cases is quite low and therefore the inference is dominated by the prior distribution and is less informative.

5.4.1 Sensitivity analysis

There are two parameters used in the inference, the likelihood parameter β and the prior probability p. In the above analysis these parameters are assumed known, but in real implementations these will be estimates, $\hat{\beta}$ and $\hat{p}$ respectively. Therefore,

Table 5.1: Inference results on different network types for $n = 1000$.

	AUC		*AP*	
	Bayesian	NETINF	Bayesian	NETINF
Erdős-Rényi	0.72 ±0.02	0.59 ±0.02	0.32 ±0.04	0.06 ±0.01
Forest Fire	0.97 ±0.01	0.86 ±0.01	0.71 ±0.01	0.56 ±0.05
Core-Periphery	0.82 ±0.01	0.60 ±0.01	0.12 ±0.01	0.05 ±0.01
Hierarchical	0.95 ±0.01	0.85 ±0.01	0.86 ±0.01	0.65 ±0.01

it is useful to understand how sensitive our inference is on the estimates of these parameters. We investigate the impact of choosing values for $\hat{\beta}$ and $\hat{p}$ that vary from the true values on the inference results.

Recall that β penalises the inclusion of links that cascades did not propagate through, while p sets the prior probability of each edge and is used to promote sparse networks. These two parameters are inherently linked to the number of edges in graphs of the posterior distribution.The parameter p is used as a prior edge density and under low data regimes will impact the density of the samples from the method. Despite their impact on the sample density, it is interesting to look at how these parameters impact the network inference metrics (e.g., AUC).

The AUC is highly influenced by the order of edges from their posterior probability. Changing the p and β values does not significantly change the ranking of the edge marginals; therefore, these parameters do not significantly impact the scale invariant AUC significantly. We present the results for the same setting as in Figure 5.4 with a relatively small amount of data, with $f = 0.5$ and $n = 100$. Figure 5.6 shows the AUC as we vary the input $\hat{p}$ and $\hat{\beta}$ independently. Other ROC summaries demonstrate a similar trend. There is a higher sensitivity to $\hat{p}$ as the denser graphs recovered at high $\hat{p}$ values have more impact on the edge marginal rankings and allow multiple transmission pathways, many of which are not observed in the true underlying network.

5.5 Experiments on real networks

Next, we evaluate the proposed algorithm on a real world network of email communications [82] at a European Research Institute. The dataset is an anonymised network containing 986 users from 4 departments (Table 5.2 shows department breakdown) at a research institute, see Reference [81] for collection details. Email

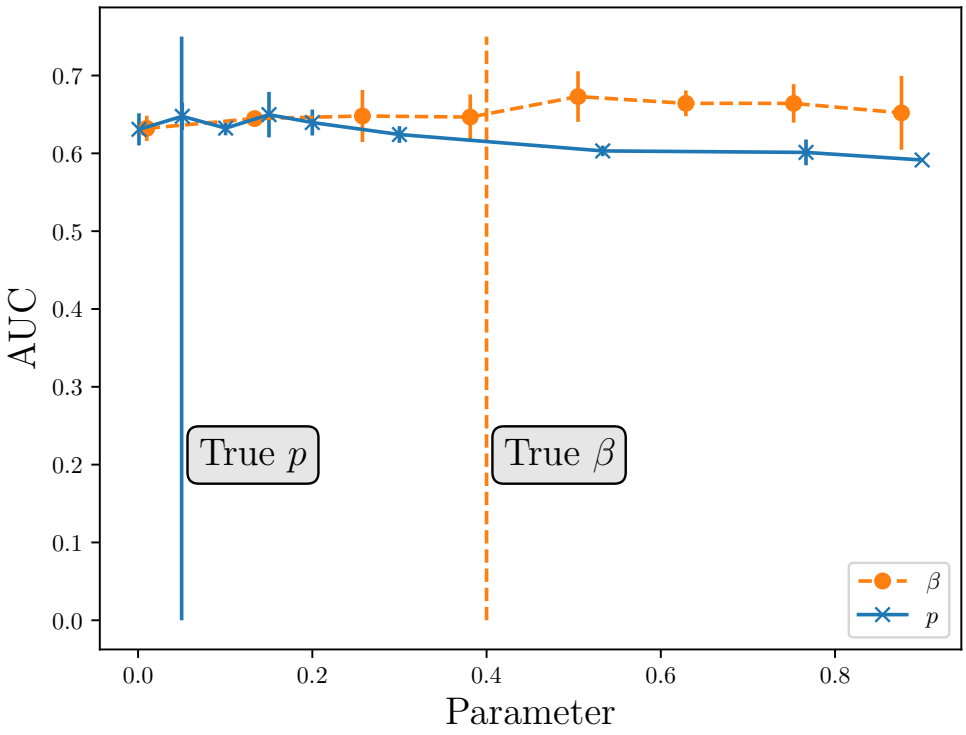


Figure 5.6: The effects of incorrect parameters on network inference ($n = 100$) with cascades that cover half of the edges ($f = 0.5$). We see that incorrect $\hat{\beta}$ values (dashed orange) have very little effect on recovery in terms of AUC. The results are slightly more sensitive to the parameter $\hat{p}$ (solid blue) but only for large deviations. Vertical lines show the true p (blue) and β (orange). 95% confidence intervals are shown for inference over 10 different networks. Other ROC summaries such as false positive alarm (not shown) have the same trend.

Table 5.2: Inference results on Email networks.

Network Type	n	Undirected edges	Number of cascades	AUC	AP
Department 1	309	1938	219	0.89	0.63
Department 2	169	1045	143	0.93	0.73
Department 3	89	973	34	0.74	0.26
Department 4	142	833	129	0.95	0.53
Entire Network	986	16064	234	0.84	0.17

networks commonly observe cascades, for example spam emails, joke emails or event invitations. We use the real underlying networks and simulate cascades, with $\beta = 0.2$, until 90% of links are activated at least once, to be consistent with the previous experiment. This is significantly less information than used in some other works [132]. We see that recovery in terms of AUC value can vary dramatically over the four departments. This is largely due to the different structures in each department that impact the properties of the cascades. Department 3 has one giant component with a high density (~ 0.2), while the other departments are clustered with lower density (< 0.1). Recovering denser networks is a harder inference problem as there are many more possible transmission pathways. These results are significantly better than the NETINF results with AUC ≈ 0.5 for all departments using the same data.

5.5.1 Twitter retweet data

Next, we implement the above algorithm on a real Twitter retweet dataset to highlight one potential application of this work. We use data collected from the Twitter garden hose in the month of October 2010, made available in Reference [67], containing URLs shared during this time as well as the associated network structure. We do some simple data cleaning to keep only unique tweet entries from users that are in the known underlying network. We follow previous work [67, 108] and filter the data to contain only retweet cascades. Cascades in which all retweets occurred at the same time are removed.

In the dataset there are 378 users with 162 cascades, corresponding to 1087 activation events. This is, at most, information about 20% of the edges in the underlying network and highlights the importance of algorithms that can deal with uncertainty. We fit Equation 5.1 to the data to find an appropriate value for α. Note that while we use the exponential form here, the waiting times do not necessarily follow an exponential distribution. To investigate the impact this has on inference, we also simulate cascades on the Twitter network. We simulate random cascades from our model with exponential waiting times until 20% of the edges are active. We also simulate cascades with exponential waiting times where each cascade is simulated over the same nodes as observed in each of the true twitter retweet cascades.

Figure 5.7 (blue solid) shows the precision-recall curve for the recovery of the underlying network. Despite the data violating some assumptions of the model it still performs well. As expected, when cascades follow the exact model we see the best performance. When using the same nodes as the true cascades but with exponential times we see quite similar performance, especially in the region of high

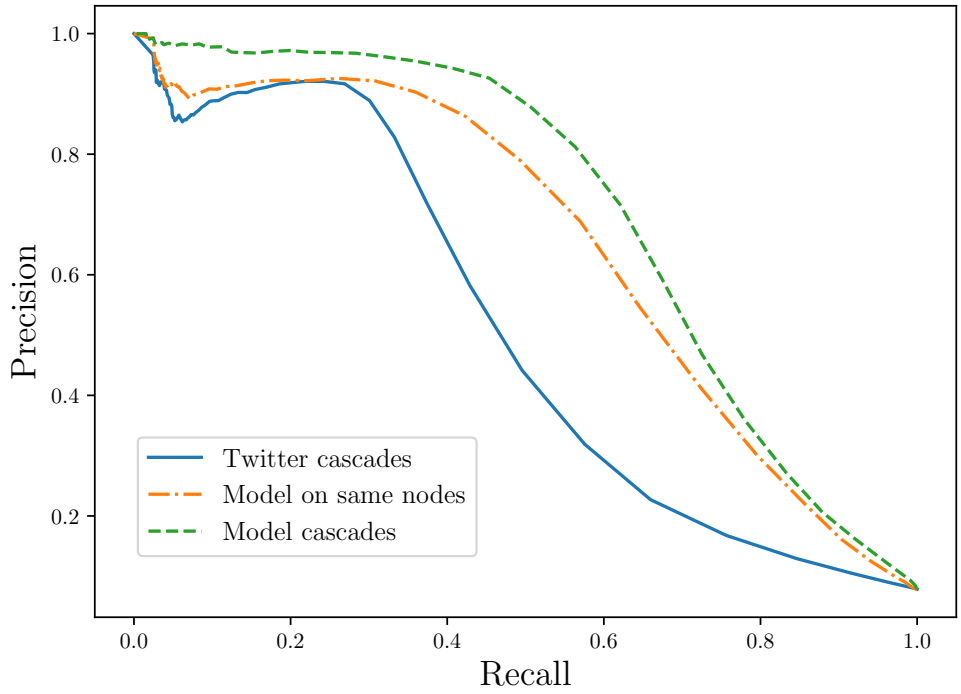


Figure 5.7: Precision-recall curves for inference of a Twitter network with retweet cascades. We show inference using the true cascade data (blue solid) as well as with cascades simulated from the assumed model (green dashed). Cascades simulated with exponential waiting times on the same nodes as the underlying data are shown in orange (dotted).

precision. The precision has a more pronounced drop for the true cascades. It is likely that true cascade times may vary from the exponential in the tail, e.g., users are only on Twitter at various points throughout the day, so it is natural there are longer breaks between retweets. Alternative waiting time distributions, for example a power-law [3], can be included in the method to replace Equation 5.1. Analysis of the data could be used to elucidate a more appropriate form, or other data specific properties that could easily be included in the method. For example, bots often post at the exact same time but are not necessarily connected. Additionally, if it becomes apparent that the cascades are following a different model the method can use the alternative likelihood.

Figure 5.8 depicts a visual representation of the recovered network. The thickness of the edges show the posterior probability, filtered for edges with posterior proba-

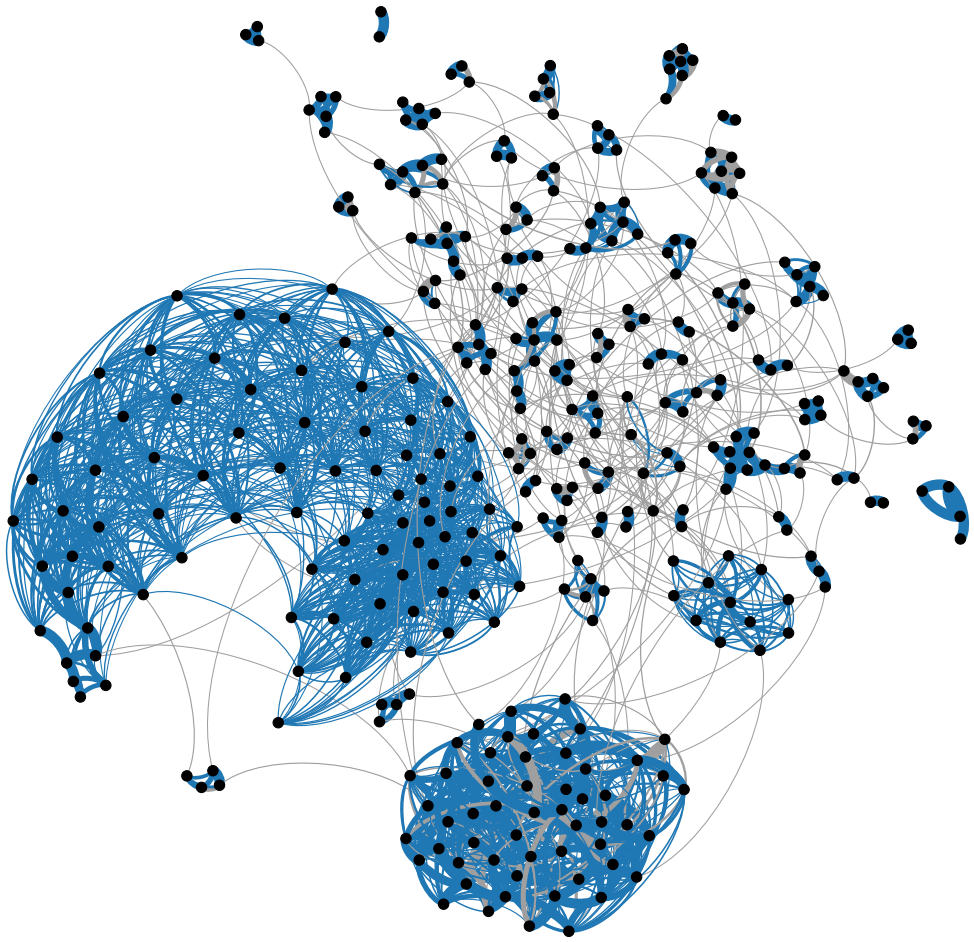


Figure 5.8: Recovered network for retweet data on a Twitter network showing the posterior probability of edges that are in the underlying network (blue) and those that are not (grey).

bility larger than the prior. Blue edges exist in the true Twitter network and grey edges do not. We can clearly see the algorithm recovers edges between clusters in the network well with high certainty, while returning edges between clusters with lower posterior probability. This is due to the lack of data as very few cascades move between clusters.

Some of these clusters may be bot networks, and this highlights other potential applications of this work. For example, posterior samples could be used to consider the probability of clustering between individuals and groups for community detection [108] and bot detection.

5.6 Discussion

Beyond improved performance, particularly when data is limited, there are many benefits and possible extensions to a Bayesian approach to this problem. We will discuss some of these below.

What effect does the prior knowledge give us? A major benefit of Bayesian inference is the ability to incorporate prior knowledge of all or part of the underlying distribution. If some of the edges or edge dependencies in the underlying graph are known, perhaps from prior experiments or partial observations, this can be incorporated to provide a better estimate of the other edges. This allows the above method to be used in missing link inference, when much of the network is known. Alternatively, knowing something about the degrees of each node can be used to sample graphs with the given degree sequences or expected degree distributions.

Noise & missingness There is the potential for inaccuracies in the measurement of the cascade data. In the above analysis we assume that there is no missing data; that is, all activations are observed and recorded correctly. There are three main sources of noise:

- Measurement errors: e.g., missing or noisy measurements — nodes that are activated are not recorded or recorded incorrectly.
- Unobservable components: e.g., outside influence: nodes are activated by information from an external source or node not in our network.
- Model errors: e.g., incubation time does not follow the assumed exponential distribution.

The Bayesian framework used here provides a natural way to deal with uncertainty and we can incorporate known missingness in the model. For instance, it is reasonable to assume noise in time measures, e.g., $t^\star = t_{\text{recorded}} + \epsilon$ for true activation time $t^\star$ and $\epsilon \sim \text{Exp}(\lambda)$. One possible issue here is that the errors may cause the activation order of the cascade to change. This would cause the true graph to have zero probability under the current model. There are two possible ways to address this. We could use another function $P(t_v, t_u)$ instead of the indicator function in Equation 5.1 to place small probability on the activations where the parental relation is incorrect. For example, a logistic function would be a natural extension of the indicator function and allow for small errors in the activation order. A more thorough approach is to infer the true activation times $t^\star$ using MCMC or Gibbs sampling (or other inference methods) jointly with network inference. This method generalises and can be applied to different types of noise or missingness under some assumed model. For example, perhaps only a snapshot of the network is observed at given times over the cascade so $t^\star \in t_i - t_{i-1}$. Alternatively, we can build the observation process into the model directly with an influence model based on partially observed Markov processes [75].

To incorporate unobservable nodes, many other works include an artificial 'source' node in the network that is connected to all nodes with some small probability γ that provides external influence. This parameter could be constant over the network or depend on the node. Higher γ values increase the likelihood of external influence so reduce false negatives but increase false positives. With some prior knowledge we can also estimate γ.

We have modelled the probability a node is activated from its neighbour as a simple exponential decay. However, this can be extended to include other distributions as well as characteristics of the nodes and cascades. For example, node degree as high degree nodes are potentially less likely to be influenced by a low degree node or some node similarity measure (are they 'politically' or 'socially' similar or produce similar textual content). Inferring some of these characteristics of nodes, such as their clustering, from information cascades has recently been addressed in [108] and remains an area of future work. The dependence of social influence and information flow on the content of the information could also be investigated and incorporated. There is also evidence for other distributions of waiting times, such as power-law with or without an exponential cutoff [55, 91]. These can easily be incorporated above.

Inferring β_{uv} parameters that are dependent on the link itself There have been a number of works [43, 54, 108] inferring the link strength between individuals as a measure of social influence or social trust using EM algorithms based on

communication patterns. More recently, Bayesian methods have been applied in this space. Embar *et al.* [43] use a Bayesian framework to analytically and empirically infer link strengths. The method here could be straightforwardly extended to include an inference of link strengths β_{uv}; however, it is likely that much more data is required to infer strengths and strength distributions rather than simple adjacencies.

False positives and negatives The method reports marginal edge probabilities based on the cascades observed. Highly probable edges that do not exist in G^*, or, conversely, edges of low probability that do, can still be informative. Non-edges with high probabilities are indicators that these edges provide alternate pathways through the network or may have been incorrectly measured in the ground truth. Edges of low probability may be due to inactivity of the nodes or the presence of a highly probable edge that promotes the use of an alternate pathway. The presence or absence of edges in the posterior distribution tell us about the social influence pathways present in the network. Even the incorrect classifications the algorithm makes can be informative when coupled with uncertainty quantification.

Time varying networks and data This algorithm assumes the underlying graph is static and all cascades have been observed. It is widely understood that networks are time varying in many cases and often cascades are observed in an online fashion. Online Bayesian inference methods like sequential Monte Carlo present an extension of this algorithm that can handle streaming data. This stream could come in two forms: observing a cascade as it progresses over a network or observing a stream of full cascades when each concludes, or even both. Additionally, online methods are capable of inferring dynamic network structures. As information is collected from these changing networks online algorithms will update posterior estimates of the changing network.

Future work Metropolis Hastings is known for slow convergence in high dimensional posteriors and can be memory intensive. We have shown that, despite this, accurate and useful results can be obtained. Other Bayesian inference methods are designed to combat these shortcomings. Sequential Monte Carlo, combined with importance sampling would not only allow online inference as discussed but also provide scope for more advanced proposals to improve acceptance rates. Additionally, using an augmented model to allow deviations from the distribution of interest will improve mixing time and can be used in conjunction with importance sampling to return to the correct distribution. When inferring directed networks with the analysed model, the neighbourhoods of each node are decoupled and can

therefore be inferred individually. Additionally, other models, such as the survival analysis approach in [54] and generalised linear cascade models [111] can be parallelised over the nodes. The approach described can be easily extended to these models and running these chains in parallel on each node will improve complexity. These extensions provide exciting areas of future research.

5.7 Conclusion

We have presented an algorithm to infer the posterior distribution of networks that facilitate observed cascades. Despite the high dimensional posterior basic MCMC methods can extract useful information and infer network structure based on samples from the distribution. We also show that uncertainty quantification in networks, specifically when inferring networks from observed dynamics, has many benefits in not only the potential to improve informed decision making but also the ability to understand the complex interaction of network structures and the dynamics we observe.

Chapter 6

Improving computation and applicability of GraphMCMC

In Chapter 5 we introduced a Bayesian framework for inferring an undirected network structure from information cascades. We showed that this method provides a unique view of the inferred graph as a posterior distribution over graphs that can facilitate the cascade. However, the nature of the likelihood model required taking samples of the entire graph with limited scope for parallelisation or speed ups. In this chapter we consider two alterations to the model and method in order to improve computation time and consider cascades that are observed in real time as a stream of data.

First, the previous work required consideration of the entire graph at once, preventing parallelisation over each node. Here, we consider the inference of directed graphs, which, with a slight reworking of the likelihood, allows for disjoint neighbourhoods of each node to be inferred. We show that while inferring directed graphs doubles the number of edges to infer, inferring neighbourhoods separately improves computation time and also becomes substantially easier to apply to real data.

Second, in reality, we often observe cascades as a stream of data, and we may need to assimilate this data in real time, but it is computationally expensive to recompute the entire posterior distribution. There are two possible streaming setups:

- **Sequential cascades**: we observe some set of cascades on a network, and then subsequently observe another cascade.
- **Concurrent cascades**: we observe many cascades occurring concurrently

Sequential Monte Carlo (SMC) is designed to consider streaming data. We initially design and implement a data-annealed SMC sampler to infer the posterior distribution when cascades are occurring sequentially. We find that as the posterior support for each cascade is significantly different SMC performs poorly using the likelihood derived in the previous chapter. We address this by incorporating external influence which results in posterior support for all graphs. Finally, we combine the new neighbourhood inference likelihood with SMC to introduce a sampler that can infer networks where cascades are streaming concurrently.

6.1 Parallelisation through node neighbourhood inference

In the previous chapter the cascade model likelihood, Equation 5.5, required the sum over all possible trees in the graph and so inference required taking samples of the entire graph. This can be computationally expensive, especially as the graphs become larger. Parallelisation is a natural way to speed up an algorithm; however the likelihood as it stands cannot infer each neighbourhood in parallel. As we are inferring an undirected network both the influence of i on j and conversely j on i must be considered when determining the probability of edge (i, j). To parallelise the algorithm we instead look at inferring directed networks. Then, each node is only influenced by its in-neighbours, who in turn were influenced by their in-neighbours. Redefining the likelihood in terms of the cascade progression and directed neighbourhoods allows us to solve the problem in parallel.

First, we recall and define some notation. Let the cascade be an ordered set of node time pairs for each node in the network. i.e.,

$$c_k = \{N_1 : t_1, N_2 : t_2, ..., N_n : t_n\}_k, \tag{6.1}$$

where $t_i < t_j$ for $i < j$. If N_k was not activated in the cascade then implicitly $t_k = \infty$. Let $\mathcal{N}_j$ be the set of edges connecting j with its in-neighbourhood, i.e.,

$$\mathcal{N}_j = \{(N_i, N_j) \forall N_i \in N | (N_i, N_j) \in E\}. \tag{6.2}$$

We refer to an *in-neighbourhood* to be the nodes that are influencing j.

Now, define the set of edges connecting j to nodes that have already been activated in the cascade as $\mathcal{N}_j^A$. i.e., edges connecting j to its *active neighbourhood*. As the cascade is ordered, $\mathcal{N}_j^A = \{(N_j, N_i) \in E | (N_j, N_i) \in E, t_i < t_j\}$. Intuitively, only already active nodes could influence the current node j, so instead of considering

the whole graph we only need to look at each $\mathcal{N}_j^A$. All the edge sets, $\mathcal{N}_j$, are disjoint, and so the underlying edge set of the graph G is the union of all $\mathcal{N}_j$.

Now consider the probability of a cascade given the underlying graph G. First, consider an arbitrary activation (N_j, t_j), where $t_j < \infty$. This activation depends only on the cascade up until immediately prior to t_j, denoted $c^{(t_j)}$, and the connections to the active neighbourhood of N_j. The probability of an entire cascade c is the product of each activation. This gives,

$$P(c|G) = P(T_1 = t_1|t_0, \mathcal{N}_1^A)P(T_2 = t_2|t_0, t_1, \mathcal{N}_2^A)P(T_3 = t_3|c^{(t_3)}, \mathcal{N}_3^A)\dots \\ P(T_j = t_j|c^{(t_j)}, \mathcal{N}_j^A)\dots P(T_N = t_N|c, \mathcal{N}_N^A). \quad (6.3)$$

If the cascade is not global, some nodes are not active so their activation time is recorded as $t_k = \infty$.

From the independent cascade model [74], an active node could have been activated by any one of its active neighbours. So, for nodes where $t_j < \infty$ we have

$$\begin{aligned} P(T_j = t_j|c^{(t_j)}, \mathcal{N}_j^A) &= \sum_{k \in \mathcal{N}_j^A} P(T_j = t_j|c^{(t_j)}, \rho_j = k) \\ &= \sum_{k \in \mathcal{N}_j^A} \beta(1-\beta)^{|\mathcal{N}_j^A|-1} \exp(-(t_j - t_k)) \\ &= \beta(1-\beta)^{|\mathcal{N}_j^A|-1} \sum_{k \in \mathcal{N}_j^A} \exp(-(t_j - t_k)), \end{aligned}$$

where ρ_j is the parent of N_j. For nodes that were not activated all previous activations failed so $P(T_j = \infty|c, \mathcal{N}_j^A) = (1-\beta)^{|\mathcal{N}_j^A|}$. Together,

$$P(c|G) = \prod_{j \in V} \left(\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1} \sum_{k \in \mathcal{N}_j^A} \exp(-(t_j - t_k)) \right]^{\mathbb{1}(t_j < \infty)} \left[(1-\beta)^{|\mathcal{N}_j^A|} \right]^{\mathbb{1}(t_j \not< \infty)} \right). \quad (6.4)$$

For a set of independent cascades C we have

$$P(C|G) = \prod_{c \in C} P(c|G). \quad (6.5)$$

Recall that we are interested in using this likelihood to infer the graph G, which is the union of all the directed $\mathcal{N}_j$. In the product above, as all $\mathcal{N}_j^A$ are disjoint, we can solve for each $\mathcal{N}_j^A$ in parallel.

Of course, it is possible that there are edges that we have no information about. If node N_i was never infected before node N_j we can not say anything about the

influence of N_i over N_j. Our Bayesian framework will allow these edges to have the prior probability.

We use Bayes' rule and a basic Metropolis-Hastings algorithm to infer the connections to each node, $\mathcal{N}_j^A$.

$$P(\mathcal{N}_j^A|c^{(t_j)}, T_j = t_j) \propto P(T_j = t_j|c^{(t_j)}, \mathcal{N}_j^A)P(\mathcal{N}_j^A) \tag{6.6}$$

The proposed $\mathcal{N}_j^{A'}$, differs from $\mathcal{N}_j^A$ by one link. So,

$$\alpha_j \;=\; \min\left(1, \frac{Q(\mathcal{N}_j^A|\mathcal{N}_j^{A'})P(\mathcal{N}_j^{A'})}{Q(\mathcal{N}_j^{A'}|\mathcal{N}_j^A)P(\mathcal{N}_j^A)} \prod_{c\in C} \frac{P(c|\mathcal{N}_j^{A'})}{P(c|\mathcal{N}_j^A)}\right). \tag{6.7}$$

We can now proceed in taking samples from, and inferring marginal probabilities for, each $\mathcal{N}_j$ in parallel. We use the same TNT-sampler as in Chapter 5.

When inferring undirected networks, information about (i, j) informs (j, i); however, as we are now inferring a directed network we have double the number of edges to consider. Therefore, we will need more data to get reliable estimates over the network. However, this is often a more realistic scenario as edges are not automatically reciprocal in many real-world networks, e.g., Twitter. There are at most n possible edges for each node, so convergence is at worst $\mathcal{O}(n)$ for each node.

6.1.1 Results

To confirm that this method is an improvement on the original, we consider the accuracy of the results and show that there is no decrease in accuracy while computation time is dramatically reduced.

Figure 6.1 shows the area under the ROC curve (AUC) and average precision (AP) for both the original algorithm and the parallelised algorithm. We notice that both algorithms perform similarly, with the new neighbourhood inference slightly outperforming the original.

The slight improvement is likely caused by the use of the Independent Cascade model on undirected graphs compared to directed graphs. We show the results for the same f value, i.e., the proportion of edges activated at least once. However, at comparable f values, the average number of activations per edge is larger for cascades on directed networks than for undirected networks. The increased data is likely the cause of the slight improved performance of the directed network algorithm over the original.

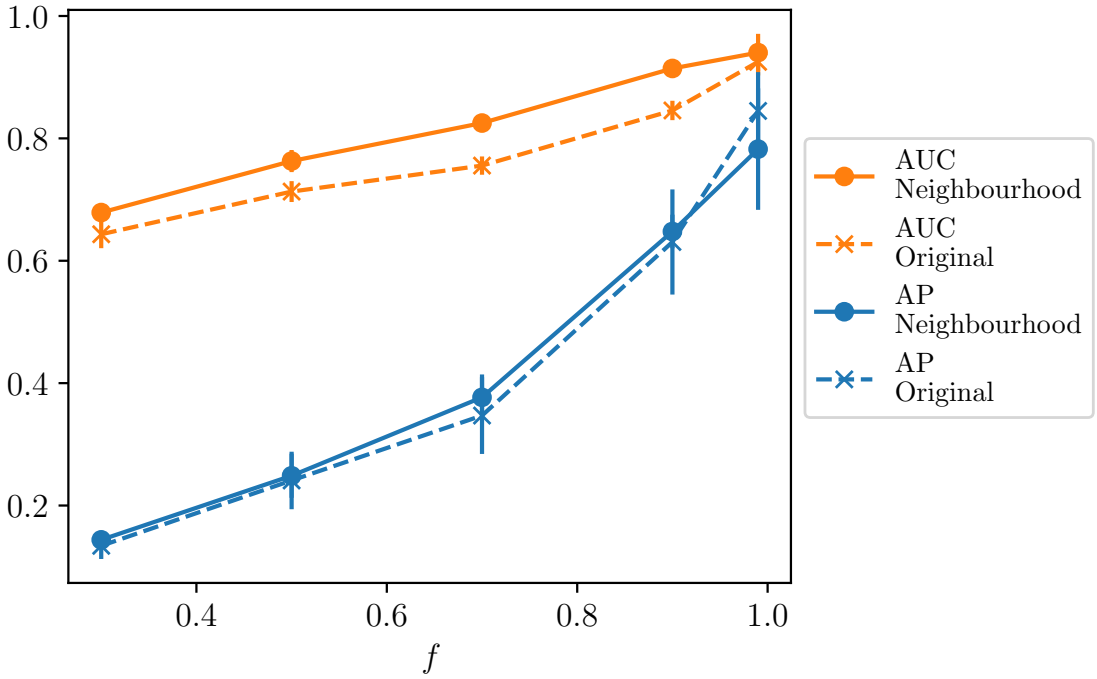


Figure 6.1: Area under the ROC curve (AUC, orange) and average precision (AP, blue) for the parallelised algorithm that infers edges in directed neighbourhoods (solid) and the original algorithm, see Chapter 5, to infer the undirected graph (dashed).

In terms of performance the new parallelised model significantly outperforms the original. In the original model iterations until convergence scaled like $\mathcal{O}(n^2)$, in the number of nodes, see Chapter 3 for further details. When we infer only the edges to node neighbourhoods, each neighbourhood requires inference of at most n links. Therefore, the computational complexity of each neighbourhood is $\mathcal{O}(n)$. Additionally, we only infer edges to the active neighbourhood, $\mathcal{N}_j^A$, of a node j. It is common that not all nodes have been activated prior to j. This provides another speed up and so even if we run this algorithm in series (no parallelisation), we still see improvement over the original algorithm. Each inference is also at worst $\mathcal{O}(|C|)$ as it is linear in the number of cascades in which node j participated.

6.1.2 Discussion

Inferring neighbourhoods rather than the entire graph is beneficial in many ways. The inference of one node is only impacted by the activation times of earlier nodes, not the entire network. As we have seen, this allows for parallel inference and improved efficiency, and it allows us to use data more effectively. Additionally, inferring directed graphs provides useful information about the direction of influence. In the previous set-up to infer undirected graphs we must consider the entire graph and nodes cannot change between cascades. Therefore, they must be collected over the same node set. In the new set-up, it is easy to truncate cascades and infer specific neighbourhoods of interest. We could also extend this set-up to networks that change with time, either the edge set or the node set, which is a current research area [49] and this leads to an exciting area of future research.

In conjunction with this work we considered the inference of neighbourhoods using the Twitter data from Chapter 5. The ability to parallelise the nodes meant we could infer larger network structures. However, we found that the increased size of the inference due to both the increased number of nodes inferring the directed network, resulted in extremely sparse data with very little signal. We also attempted neighbourhood level inference, to infer just the neighbourhoods of the most active nodes. The results here were varied and many neighbourhoods suffered from a similar sparsity issue. Further work to collect more data, perhaps retweet cascades rather than link sharing cascades, is needed to apply this algorithm and gain insights on certain networks. See Section 6.3 for further discussion on future work.

6.2 Online network inference with Sequential Monte Carlo

In Chapter 4 we saw that likelihood annealed SMC can sample from graph ensembles, and in Chapter 5 we highlighted the possibility of using SMC for online inference in the network inference problem. Modern data is continuously arriving, and is often called *streaming* data. This gives rise to the need to process data efficiently as it arrives; known as *online* inference.

As information cascades evolve with time, naturally, it is natural to receive information in a streaming fashion, i.e., we observe the activations as they occur. A good example is the COVID19 spread as we observe activations sequentially. Perhaps we observed information cascades on the nodes yesterday and today we observe another. The information from the old data has not changed, and the underlying network has likely not changed substantially, so we prefer to update our inference rather than reevaluate completely. SMC is useful in this situation when online inference is required. Here, we will use *data annealed* SMC where the likelihood remains constant but the data is introduced in batches. We consider two types of streaming situations: *(i)* entire cascades are observed sequentially — one cascade then another and *(ii)* cascades are being observed concurrently.

6.2.1 SMC for sequentially streamed cascades

Consider a cascade set $C = \{c_1, c_2, \ldots, c_{|C|}\}$. In Chapter 5 we used the entire set and a Metropolis-Hastings sampler to get samples from $P(G|C)$. However, we could subsequently observe more cascades to get $C' = \{c_1, c_2, \ldots, c_{|C|}, c_{|C|+1}\}$ after the inference has been performed. In the original MH model we would have to resample from the new ensemble $P(G|C')$. SMC is an online learning method that can update inferences as new data is observed. In this case, a new piece of data is a new cascade in its entirety (or set of cascades). Note that in this section we consider undirected networks, and in Section 6.2.2, we consider the directed case introduced above. Recall that we sample many particles and weight them according to a series of importance distributions. In this case each distribution incorporates more cascades as they are observed. We need to define our *annealing schedule*; what distributions our particles are weighted by to get from our prior distribution to the posterior. In a data annealed SMC each distribution utilises more data as it becomes available. As we observe the set of cascades $C = \{c_1, c_2, \ldots, c_{|C|}\}$ sequentially we choose to incorporate each cascade individually. So, the distribution sequence becomes:

$$
\begin{aligned}
P_0(G) &= P(G) \\
P_1(G) &= P(G|c_1) \\
P_2(G) &= P(G|c_1, c_2) \\
&\vdots \quad \vdots \quad \vdots \\
P_T(G) &= P(G|c_1, c_2, \ldots, c_{|C|}) = P(G|C).
\end{aligned} \tag{6.8}
$$

Now consider the weights for each iteration of the SMC process. Initially the weights for the sample from the prior are equal, $w_0 = 1/S$. Recall from Chapter 2 the weights for each iteration are

$$
w_i = \frac{P_i(G)}{P_{i-1}(G)} w_{i-1}.
$$

Substituting our distributions from Equation 6.8,

$$
\propto \frac{P(G|c_1, c_2, \ldots, c_i)}{P(G|c_1, c_2, \ldots, c_{i-1})}
$$

Using Bayes' rule and dropping constants gives

$$
\begin{aligned}
&= \frac{P(c_1, c_2, \ldots, c_i|G)P(G)}{P(c_1, c_2, \ldots, c_i)} \frac{P(c_1, c_2, \ldots, c_{i-1})}{P(c_1, c_2, \ldots, c_{i-1}|G)P(G)}, \\
&\propto \frac{P(c_1, c_2, \ldots, c_i|G)}{P(c_1, c_2, \ldots, c_{i-1}|G)}.
\end{aligned} \tag{6.9}
$$

Finally, substituting our likelihood and simplifying the fraction:

$$
\begin{aligned}
&= \frac{\prod_{k=1}^{i} P(c_k|G)}{\prod_{k=1}^{i-1} P(c_k|G)}, \\
&= P(c_i|G).
\end{aligned} \tag{6.10}
$$

This is intuitive; the cascades are independent so the weights are proportional to the probabilities of the graph samples given the new cascades. This results in a very simple algorithm given in Algorithm 6.1. Note that we find particles become degenerate very quickly and so we resample and move at every iteration. Therefore, $w_{i-1} = 1/S$, where S is the number of particles.

1: Sample S graphs from the prior $P(G)$
2: **for** i=1 . . . $|C|$ **do**
3: Determine weights w_i
4: Resample S particles according to weights w_i
5: Move each particle within $P(G|c_1, \ldots, c_i)$ using the MH algorithm.
6: **end for**

Algorithm 6.1: Data annealed SMC method for online network inference.

A small network example

We perform the SMC algorithm on a small ($n = 20$) network and simulate 10 cascades with $\beta = 0.3$. Figure 6.2 shows the posterior marginals for each edge at each step as an illustration of the method. Panel 1 shows the prior in which every edge is equally likely. As more cascades are observed, we see recovery of structure of the underlying network. The final panel shows the adjacency matrix for comparison. We find that it is difficult to design an annealing schedule using SMC for networks with larger than 50 nodes and 5 cascades without requiring an unreasonably large number of particles to prevent degeneracy. A large difference in support of subsequent distributions is known to be an issue for implementation of SMC. We will consider including external influence in the model to increase support of the distributions and all sampling of larger graphs.

Results on larger networks

In theory, the results for the SMC algorithm presented above should be equivalent to the standard algorithm given in Chapter 5. The posterior distribution is equivalent in both cases and so inference of larger networks should be possible. Despite this, there are some implementation considerations as n and $|C|$ increase that make working in a sequential setting slightly more difficult.

Sequential algorithms work well when adjacent target distributions are similar so that the space evolves reasonably smoothly. When this is not the case, particularly when many particles have zero mass in the new target, particles quickly become degenerate, and in the extreme all particles have zero mass. This phenomenon is prevalent in our graph space as the absence of a single link can result in a graph that is inconsistent with the cascades. In this application it is quite easy for particles sampled from the current distribution and facilitating the current cascades, to have zero mass in the next iteration because the graphs cannot facilitate the next cascade. Ultimately this is because the space of graphs is not at all smooth.

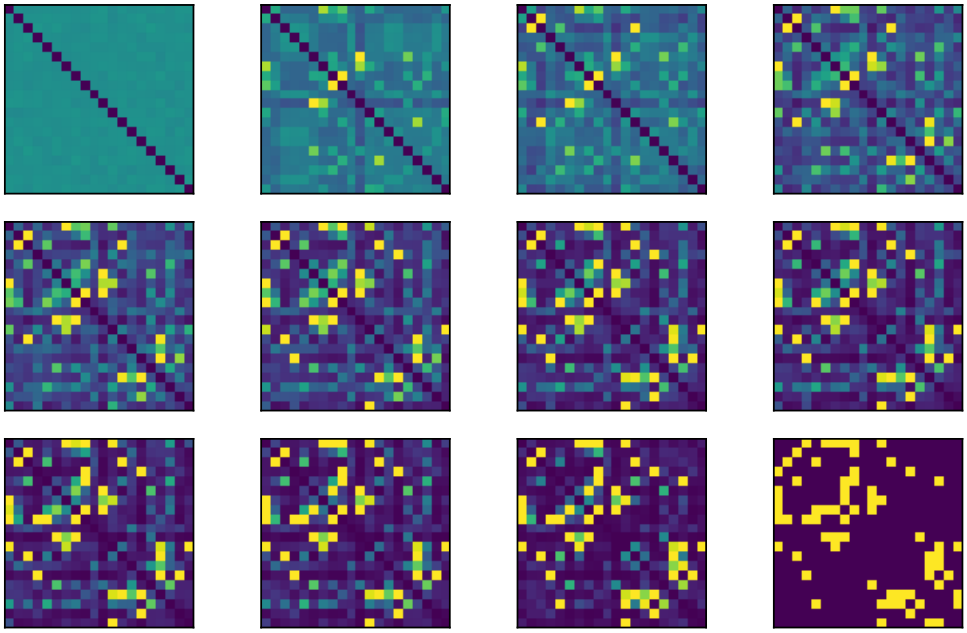


Figure 6.2: Marginal posterior averages for each step of the SMC process. As more cascades are observed, the structure is being elucidated. Panel 1 shows the prior distribution with equally likely edges and the final panel shows the adjacency matrix for comparison.

In Chapter 5 we highlighted the possibility of including external influence in our model. Some nodes may be activated for reasons other than network influence. For example in online social networks not all information propagates over the network, some information can be introduced to the network via mass media [55, 142]. Considering external inference has the added benefit that it adds some small mass to every graph. That is, even an empty graph would have some, very small, probability associated with all activations coming from the external node. We incorporate external influence by adding an additional node to the network that can activate users with probability γ. We assume all nodes are connected to this node and do not infer these connections. This improves the efficiency of SMC and allows sampling of larger graphs with more cascades. We use a final step in the SMC sequence to remove graphs that rely on external influence so we have samples from our posterior of interest.

Figure 6.3 (solid) shows the average precision and area under the ROC curve for varying amounts of data. Both metrics are very similar to the original results (dashed), with no significant difference in most cases. This shows that SMC can be used effectively to infer the networks. The use of the external node does not impact performance and employing it is beneficial to provide intermediate distributions with larger support over graphs in the posterior when cascades are observed in a stream.

The major benefit of the SMC algorithm is the ability to deal with streaming data where rerunning an entire MH model would be computationally expensive. We have shown that in a simulation setting, or in cases where we have the entire dataset, then there are intricacies here that make the implementation of SMC harder than MH. In cases when it is not difficult to find an initial graph of non-zero mass, using MH is likely to be just as effective. We showed that by including external influence we allow small deviations from the space of interest which, in the same vein as importance sampling, allows faster mixing and enables results for larger graphs. The model including external influence could also be used with the methods described in Chapter 5 to improve mixing or expand the application to cascades that include external influence.

6.2.2 SMC for concurrent real-time cascades

Above we considered the case where cascades are observed sequentially. That is, one cascade then another. Here, we consider updating the inference as new activations occur with cascades that occur in parallel.

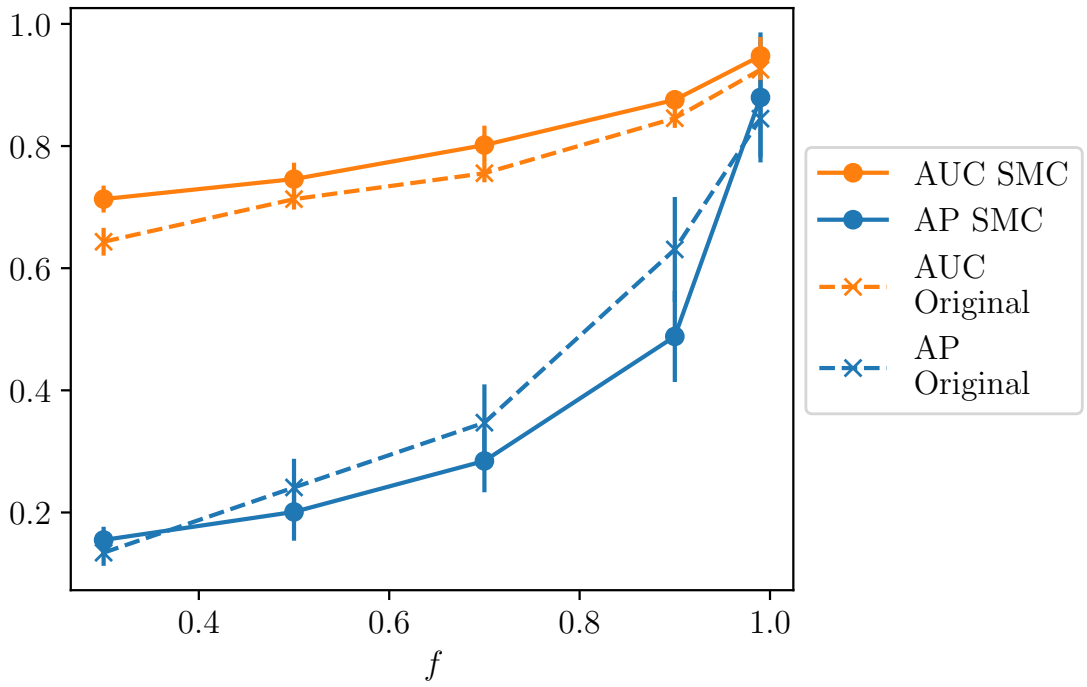


Figure 6.3: Average precision (blue) and Area under the ROC curve (orange) for the SMC algorithm (solid lines). The original results from Figure 5.3 are shown in dashed lines. 95% confidence intervals are shown by vertical bars.

A single streaming cascade: Consider again a single cascade that is ordered by time where node N_u is the node activated at t_u. Let N be the time ordered set of nodes. If the node is not activated then $t_u = \infty$. Let $c^{(t_u)}$ be the cascade up until time t_u.

Recall from the node parallelisation our likelihood can be written as:

$$P(c|G) = \prod_{u \in V} P(T_u = t_u | c^{(t_u)}, \mathcal{N}_u^A), \tag{6.11}$$

where $\mathcal{N}_u^A$ is the neighbourhood of u already activated.

We observe the cascade in an online fashion. In a similar manner as before we define our data annealing procedure to update as each new activation occurs.

$$\begin{aligned}
P_0(G) &= P(G) \\
P_1(G) &= P(G|c^{(t_1)}) \quad : \text{ Prob. of G given the cascade up to time } t_1, \\
P_2(G) &= P(G|c^{(t_2)}) \quad : \text{ Prob. of G given the cascade up to time } t_2, \\
&\vdots \quad \vdots \quad \vdots \\
P_T(G) &= P(G|c^{(t_N)}) = P(G|c) : \text{ Prob. of G given the entire cascade.}
\end{aligned} \tag{6.12}$$

After substituting our distribution sequence the weights in the SMC algorithm become,

$$\begin{aligned}
w_u &= \frac{P_u(G)}{P_{u-1}(G)}, \\
&= \frac{P(G|c^{(t_u)})}{P(G|c^{(t_{u-1})})}.
\end{aligned}$$

Using Bayes' rule,

$$\begin{aligned}
&= \frac{P(c^{(t_u)}|G)P(G)}{P(c^{(t_u)})} \frac{P(c^{(t_{u-1})})}{P(c^{(t_{u-1})}|G)P(G)}, \\
&\propto \frac{P(c^{(t_u)}|G)}{P(c^{(t_{u-1})}|G)}.
\end{aligned}$$

Substituting the likelihood formula for directed graphs from Equation 6.4 gives,

$$
\begin{aligned}
&= \frac{\prod_{j \in V} \left(\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1} \sum_{k \in \mathcal{N}_j^A} e^{-(t_j - t_k)} \right]^{\mathbb{1}(j \text{ in } c^{(t_u)})} \left[(1-\beta)^{|\mathcal{N}_j^A|-1} \right]^{\mathbb{1}(j \text{ not in } c^{(t_u)})} \right)}{\prod_{j \in V} \left(\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1} \sum_{k \in \mathcal{N}_j^A} e^{-(t_j - t_k)} \right]^{\mathbb{1}(j \text{ in } c^{(t_{u-1})})} \left[(1-\beta)^{|\mathcal{N}_j^A|-1} \right]^{\mathbb{1}(j \text{ not in } c^{(t_{u-1})})} \right)}, \\
&= \frac{\beta(1-\beta)^{|\mathcal{N}_u^A|-1} \sum_{k \in \mathcal{N}_u^A} e^{-(t_j - t_k)}}{(1-\beta)^{|\mathcal{N}_u^A|-1}}, \\
w_u &= \beta \sum_{k \in \mathcal{N}_u^A} e^{-(t_j - t_k)}. \qquad (6.13)
\end{aligned}
$$

The final fraction here simplifies as only the inference of the current activated node neighbourhood is impacted. All terms regarding nodes that were activated in a step before $c^{(t_{u-1})}$ and terms regarding nodes that are still inactive after the $c^{(t_u)}$ will all remain the same. The update only depends on the neighbourhood of the current activation.

Multiple simultaneous cascades If we redefine the annealing schedule slightly then we can use SMC for multiple cascades. Currently we update our samples each time we see a new activation. However, it is possible that many cascades are happening at once. So instead, we take a constant time step, Δt, and infer $P(G|c^{k\Delta t})$ for the k'th step. We consider the weights for each step in the SMC process. In the following we consider the product over the cascades in the set C. When taking the product over each node j, to denote that the times and neighbourhoods of node j refer to those in the cascade c, we put c subscripts on the term in the product (instead of adding a superscript c to every j).

$$
\begin{aligned}
w_k &= \frac{P_k(G)}{P_{k-1}(G)} \\
&\propto \frac{P(C^{k\Delta t}|G)}{P(C^{(k-1)\Delta t}|G)} \\
&\propto \prod_{c \in C} \frac{P(c^{k\Delta t}|G)}{P(c^{(k-1)\Delta t}|G)}
\end{aligned}
$$

$$= \prod_{c\in C}\prod_{j\in V}\left(\frac{\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1}\sum_{k\in\mathcal{N}_j^A}\exp(-(t_j-t_k))\right]^{\mathbb{1}(t_j<k\Delta t)}\left[(1-\beta)^{|\mathcal{N}_j^A|}\right]^{\mathbb{1}(t_j\not<k\Delta t)}}{\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1}\sum_{k\in\mathcal{N}_j^A}\exp(-(t_j-t_k))\right]^{\mathbb{1}(t_j<(k-1)\Delta t)}\left[(1-\beta)^{|\mathcal{N}_j^A|}\right]^{\mathbb{1}(t_j\not<(k-1)\Delta t)}}\right)_c$$

$$= \prod_{j\in V}\prod_{c\in C}\left(\frac{\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1}\sum_{k\in\mathcal{N}_j^A}\exp(-(t_j-t_k))\right]^{\mathbb{1}(t_j<k\Delta t)}\left[(1-\beta)^{|\mathcal{N}_j^A|}\right]^{\mathbb{1}(t_j\not<k\Delta t)}}{\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1}\sum_{k\in\mathcal{N}_j^A}\exp(-(t_j-t_k))\right]^{\mathbb{1}(t_j<(k-1)\Delta t)}\left[(1-\beta)^{|\mathcal{N}_j^A|}\right]^{\mathbb{1}(t_j\not<(k-1)\Delta t)}}\right)_c.$$

In the above equation we have a term for nodes that have been active before the current time $k\Delta t$, using $\mathbb{1}(t_j < k\Delta t)$. In practice $t_j = \infty$ if it is not yet activated in the cascade, so we could also use Equation 6.4 directly. This simplifies slightly, to

$$w_i \;=\; \prod_{j\in V}\prod_{c\in C}\left(\frac{\left[\beta(1-\beta)^{|\mathcal{N}_j^A|-1}\sum_{k\in\mathcal{N}_j^A}\exp(-(t_j-t_k))\right]^{\mathbb{1}(t_j>(k-1)\Delta t,t_j<k\Delta t)}}{\left[(1-\beta)^{|\mathcal{N}_j^A|}\right]^{\mathbb{1}(t_j<(k-1)\Delta t)}}\right)_c.$$

Considering the SMC algorithm in this way allows for the parallelised inference of concurrent real-time cascades. This extends the applicability of this work to cascades that are occurring in parallel. We expect this method to result in similar performance to the off-line algorithm as we are sampling from the same posterior distribution. Not only could we apply this to data that is collected concurrently, but we could also use applications of this work on real data to provide insight how networks change over time.

6.3 Discussion and future work

The past two chapters have introduced some novel methods for quantifying uncertainty in the network inference space. We have addressed some extensions of the work from Chapter 5 in this chapter, but there are many directions this research could progress.

What network are we inferring? In the above work we have used an underlying ground truth of links to determine the performance of the algorithm.

Recovering this well is obviously a good result. However, the likelihood of these links and the dependence of links on each other are a direct result of information sharing. This can be subtly different to the ground truth in many cases. Consider your favourite social media feed: although in theory you follow (potentially) many accounts, there are few that influence you to share content. So, the network we can infer from these information cascades will be a subset of the actual connections — maybe your most influential connections. This method provides insight into real networks along which information flows.

Extending the model: We have seen that including an external node in the inference can improve mixing and aids our data-annealed SMC. We could straightforwardly use this model in our inference, and this would likely be more realistic to use on real data. There are many other extensions we can consider for our model. Further analysis on real world data/cascades to determine how realistic some of our model assumptions are and where to extend the model would allow further applications on real data.

Fake news and Twitter bots: We have presented methodologies that can provide insight into the likelihood of connections between individuals from information flow. Particularly, in this chapter we have introduced node parallelisation and sequential algorithms to improve applicability to datasets one may come across. Using these methods to gain insights into systems presents some interesting areas of further research. Recently, there has been a focus on understanding 'fake news' and how it propagates over the Internet and influences both individuals and the collective. The differences between regular news and 'fake news' as well as the network structure between actors that participate in each type of network is particularly important [149]. The methods introduced here could be used to estimate the influence networks using cascades of both real and fake news, and investigate the structural differences between the two. In a similar vein, we can infer bot networks and research how their network structures are impacted by information sharing, especially compared to human users. This could then be used to identify bots and bot networks. We saw an example of this in Section 5.5 where a subset of the network we inferred was a bot network.

Focussing on subsets of nodes: Throughout this work we have considered the inference of all edges to gather a picture of the entire network. However, often data is quite sparse across the entire network and so there is little signal in the data to infer the entire graph. The work presented in this chapter allows for the inference of neighbourhoods individually. This gives much greater flexibility to

infer neighbourhoods of only a subset of 'interesting' nodes, or nodes with 'enough' data.

Structures beyond edges: Currently, we evaluate our models using the edge marginal probabilities in the posterior. This is a natural simplification of the very high dimensional posterior distribution and allows us to compare to underlying ground truth. However, the inference of a posterior graph space gives rise to the ability to consider more advanced structures. For example, transitivity is often considered an integral part of social networks as friends of friends are likely to meet and be acquaintances or friends. It would be interesting to consider whether this extends to influence networks. Additionally, understanding the pathways through which information flows is just as, if not more, important than specific edges; particularly if we are interested in altering the network to improve or hinder movement. The posterior distribution can be used to identify dominant pathways. Or to identify pathways that are more or less dominant than we expect under some null model. There are interesting implications in this area on message passing and how network structures develop to promote or hinder dynamics propagating over the network.

6.4 Conclusion

We have introduced extensions to the basic method described in Chapter 5 to provide wider applicability of these methods. The ability to parallelise the algorithm expands the use of this method to much larger networks. It also allows inference of directed networks and neighbourhoods individually when inferring the whole network is expensive or unnecessary. The introduction of an SMC algorithm provides a more efficient algorithm for streaming data. In conjunction with this method we highlighted the use of an 'external node' to account for external influence in the cascade and provide wider support in the posterior to improve mixing.

In the past two chapters we have been focussing on inferring graph adjacencies from data that has been observed over the network. In the next chapter we continue to use data that is collected by observing processes over networks, but move from social networks to the Internet. We also consider inferring distributions of node and link properties, rather than just adjacencies.

Chapter 7

BeCAUSe: An algorithmic framework for network tomography

This chapter is based on C. Gray's contributions to the manuscript:

> Mosig, C., Gray, C., Roughan, M., Waehlisch, M., Pelsser, C., Bush, R., and Schmidt, T. BGP beacons, network tomography, and Bayesian computation to locate route flap damping. *Proceedings of the ACM Internet Measurement Conference*, (2020) 492-505.

In the previous chapters we have been considering sampling entire graphs from conditional distributions given some data observed on the network. This can be thought of as inferring edge properties — in our cases, edge existence. In this chapter we focus on the inference of node properties from the observation of data on a network. Specifically, we address the network tomography problem which requires inference of node properties when only data about paths over the network is available.

The motivation for this work stems from an application in Internet routing, specifically a phenomenon called Route Flap Damping (RFD). We provide a brief overview of this, and other application specific terminology, in the following section. The work from this chapter contributed to the publication under submission: "BGP Beacons, Network Tomography, and Bayesian Computation to Locate Route Flap Damping" [59]. This paper includes a detailed measurement setup to collect data, heuristic methods and a Bayesian computation method for pinpointing which nodes are displaying the property from the path data. This chapter

is an expansion of the Bayesian computation contribution and focuses on how to use Bayesian computation to pinpoint nodes that are displaying certain properties from the path data. Contributions belonging to other authors of the paper are noted if required.

We introduce the method in the context of a specific Internet application to highlight the novel insights that can be obtained. However, the method can be applied to network tomography problems in general and so a deep understanding of the application area is not essential to understand the method in general.

At a high level, the Internet is a network of Autonomous Systems (ASes) that, among many other things, transmit packets between source and destination computers to fulfil browsing requests. We will refer to ASes as nodes in the Internet network[1]. On top of this, ASes need to know where to transmit packets to arrive at the desired destination. Therefore, the ASes share routing information between each other by announcing which IP addresses they can access. Of course, there are many layers of the Internet; here we focus on the routing information layer and refer the interested reader to review papers by Willinger, W. et al. [120,144].

A path is a series of ASes through which information has passed. Mant experiments on the Internet collect path based data — i.e., we can collect data about paths but not specifically about individual nodes. However, often information is required at the node level. This motivates the network tomography problem, coined by Vardi [138]. The aim is to determine node or link level information from path data.

In this chapter we develop a general Bayesian framework for the network tomography problem to pinpoint nodes displaying a particular property given labelled path data. Solving the problem in a probabilistic framework can allow for measurement error in the data and nodes that behave inconsistently. We then apply our method to a specific Internet measurement problem: identifying ASes that are showing a property called route flap damping (RFD). We highlight the general applicability of the framework by inferring nodes displaying another property, route origin validation (ROV), and highlight extensions to the method that can improve applicability and robustness of the method.

In Section 7.1 we give a brief overview of the required knowledge to understand the Internet application of the chapter, and introduce the network tomography more explicitly. In Section 7.2 we introduce the probabilistic view of the network tomography problem and the algorithm we use to find solutions: BeCAUSe (BayEsian Computation of AUtonomous SystEms). We give an overview of the data collected by our collaborators in Section 7.3 and then use this data to infer nodes displaying

[1]They are not actually nodes but this approximation is often applied [120]

the property RFD in Section 7.4 and 7.5. We also implement model extensions to incorporate measurement error and deal with inconsistently behaving ASes.

7.1 Background

The Internet is a network of networks. The nodes of this network are Autonomous Systems (ASes) and each AS itself is a network of routers controlled by the same administrator. The ASes are connected by the Internet Protocol (IP) [145] which allows communication between them, and subsequently the transmission of packets along paths of the network. Here we will focus on the top level network of the interaction between distinct ASes.

7.1.1 BGP and Route Flap Damping

BGP, the Border Gateway Protocol [130], is an Internet protocol that is designed to exchange reachability and routing information among ASes. The ASes announce information to each other regarding which other nodes it can reach. It is often said to be the 'glue' between the ASes and is essential to the functioning of the Internet.

Route Flapping To exchange information ASes send routing information between each other. Each AS will *announce* its best route for certain IP prefixes to its neighbours. Additionally, if the routing information is no longer known then the AS will *withdraw* the information. This process is essential to address the constantly changing Internet. However, errors in hardware, software, configurations *etc.* can cause announcements and withdrawals to occur in quick succession. This causes a phenomenon called *route flapping*.

Route Flap Damping In the early 90s constant BGP 'churn', caused by continuous route flapping, overloaded many routers. To overcome this, core network operators and vendors met to develop Route Flap Damping (RFD) to damp/suppress routes that were flapping too frequently. Parameters can be chosen to determine how frequently 'too frequently' is, and suppressing these announcements reduced strain on the system. When a node observes a flapping announcement it damps the signal and does not pass this routing information to its neighbours. Therefore, if a flapping signal is passing over a path in the network, if any of the nodes damp the signal then the suppression will be observed at the destination.

There is some debate as to the impact of RFD on Internet routing. In the early 2000's it was shown that RFD had negative effects on Internet routing. So, as computation power had increased dramatically, it was assumed that most vendors removed RFD from their protocols. Subsequently, it has been shown that some settings of RFD are helpful [109], and so there is some level of confusion of best practice among the community. Despite its importance and impact on the Internet, there has been no measurement of the deployment of RFD across the Internet. Understanding the deployment and parameters is important because RFD can cause reachability issues [24, 89] and can obscure measurements for researchers. Additionally, it is a canonical example of a network tomography problem where we localise routing properties from external measurements.

Active measurement We have already highlighted that the Internet sends information between neighbours. This can be measured by sending artificial information over a path from the initial sender, we call this a beacon, and the receiver is called a vantage point. Often researchers implement active experiments by using well defined input signals to provoke observable events at the vantage point. For example, generating frequent announcements and withdrawals at a beacon will cause damping to be observed at vantage points if a node along the path employs RFD. The problem we address here is the binary network tomography problem — identifying which AS on the path caused the signal at the vantage point. The left hand side of Figure 7.1 depicts the active measurement of RFD through an example network.

7.1.2 Binary network tomography

Commonly in Internet network measurement, it is either impractical or infeasible to measure the characteristics at nodes directly. ASes have diverse owners and are often competitors, so querying the 'nodes' directly is unlikely to yield large scale results. Hence, our aim is to identify ASes deploying RFD using path data collected through active measurement, and this requires finding solutions to a binary network tomography problem. The right hand side of Figure 7.1 summarises the data available from the path data and the objective is to identify which node/s are RFD-enabled.

The essential nature of network tomography problems is that they map properties of paths to nodes or visa versa. For example, here we observe path properties, and wish to infer node properties from these. Typical tomography problems observe volumes, e.g., volume of traffic, or size of delays, and the aim is to allocate these

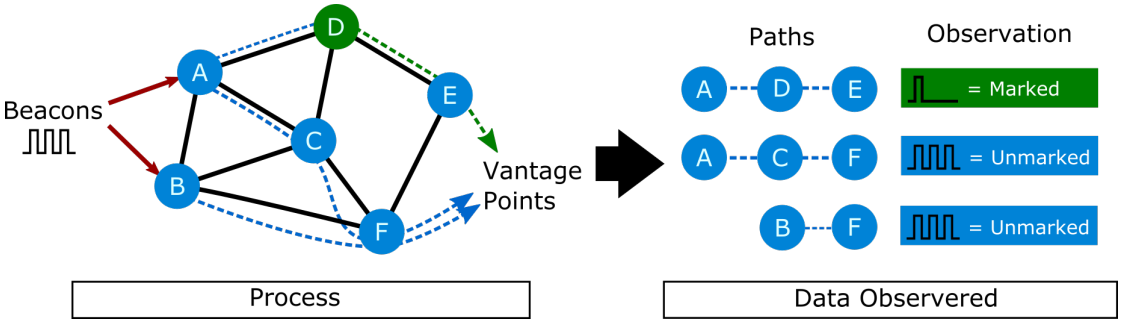


Figure 7.1: Depiction of the active measurement process for network tomography and resulting data. The left hand side shows the underlying (unknown) network with one node, D (also unknown), displaying a property of interest. The signal is sent from beacons and collected at vantage points. Paths that pass through D have a signal indicative of the property of interest (in this case route flap damping). The right hand side shows the corresponding data collected. The resulting signals are converted into labels (in this case binary markings) and paths are also collected. The binary tomography problem is to use this data to identify D as the node displaying our property of interest.

to nodes on the path. In these cases the relationship between node and path properties are represented as linear equations. In contrast, here the relationship is binary. That is, each node is considered to either have property A or not. Paths have property A if at least one node on the path has property A, and if no nodes have the property, then the path will not.

We express this mathematically by defining variable x_i where,

$$x_i = \begin{cases} 0 \text{ if node } i \text{ has property A} \\ 1 \text{ if node } i \textbf{ does not} \text{ have property A.} \end{cases} \tag{7.1}$$

Then, a path J consisting of a set of nodes ideally satisfies

$$y_J = \prod_{i \in J} x_i \tag{7.2}$$

where,

$$y_J = \begin{cases} 0 \text{ if path } J \text{ has property A} \\ 1 \text{ if path } J \textbf{ does not} \text{ have property A.} \end{cases} \tag{7.3}$$

From this, if $y_J = 1$ then all $x_i \in J$ must be 1, and if $y_J = 0$, the path is displaying property A, then at least one $x_i \in J$ must be 0 and displaying property A.

From this, one can construct a set of equations, one for each measurement, and subsequently a likelihood for y assuming independent paths. It is worth noting that path changes may result in more than one measurement for each beacon-vantage point pair. Solving this set of equations would ideally solve the localisation problem. This problem is equivalent to the Boolean satisfiability problem (SAT), which is a well known NP-complete problem. Good heuristic tools exist to solve this type of problem and indeed these types of approaches have been used to detect ASes that censor [26].

This binary tomography problem to infer properties of nodes is similar to the inference of binary edge adjacencies we studied in Chapter 5 & 6. There is scope to use the same tools as above and take graph samples where we propose which display property A in each step. From here we could identify the nodes that are most likely to display the property. However, there are a few common problems in solving tomography using real data that present issues for heuristic methods as well as a direct binary inference using MCMC methods.

1. There is often not enough path measurements to obtain a unique solution, and so some side information is needed to help refine solutions. For example in Figure 7.1 there is insufficient data to determine if D or E is RFD-enabled to produce the observed data. Sparsity is often used here, but may not be enough.

2. The noise inherent in any set of measurements often means the equations are inconsistent, and hence have no solution. This is equivalent to the posterior being uniformly zero.

3. The assumption that each node displays the property consistently is not universally true. Similarly to measurement noise, this can result in no solutions.

Related Work There have been many attempts to solve specific instances of the network tomography problems that occur in Internet measurement. Castro et al. provide a survey of the early work, and more recent work involves maximum likelihood estimators using various data collection techniques [20, 23, 31, 40, 41]. Very few works consider Bayesian inference approaches. Preliminary studies use Metropolis-within-Gibbs, with a few additional conditions [103], including sparsity (i.e., that the condition we are interested in is rare), to locate the loss probability of packets on trees. They focus on the simpler task of inferring loss over trees, require an extra expensive Metropolis-Hastings step to sample from the conditional distributions, and use simulated data. Binary network tomography is often used to determine where 'problems' are occuring on the network, e.g., anomaly detection [5], and hence sparsity is a reasonable assumption, but we do not know if it is in

our cases [26]. Moreover, our network is not a tree. There are also attempts to use heuristics that use specific knowledge of the property of interest to pinpoint the culprit ASes [21, 24, 51, 133]. We aim to create a generic method that can work on many problems.

Our approach uses a probabilistic approach to determine if an AS has a particular property that is not necessarily bad or rare. We aim to allow for the possibility of partial properties (a relaxation of the binary constraint), and to provide uncertainty quantification of the inference that current approaches to binary tomography do not typically allow.

7.2 BeCAUSe: Bayesian Computation for AUtonomous Systems

We reframe the binary tomography problem into a probabilistic setting to introduce an algorithmic framework that can find solutions on general networks while accounting for sparse and noisy data. We introduce a general framework that uses computational Bayesian inference and apply it specifically to RFD in Section 7.4. We demonstrate the applicability of this algorithm to more general tomography problems by inferring Route Origin Validation (ROV) in Section 7.6 and highlighting an alternate likelihood for incorporating measurement error.

Let each AS have some probability p_i that it has property A of interest. We also define $q_i = 1 - p_i$, to be the probability that the AS **does not** have property A. Throughout this chapter we will refer to marked paths as those that show property A and unmarked paths as those that do not show property A. We assume that each AS contributes to the likelihood of any single path showing A independently[2]. So, the probability that a single path J is unmarked is the product of each AS q_i along the path. Hence,

$$\text{Prob}(J \text{ does not show A}) = \prod_{i \in J} q_i. \tag{7.4}$$

[2]We are not presuming that ASes are independent. We are well aware that, for instance, sibling ASes may have correlated policies. We assume that nodes on the path contribute independently to the property on that path. That is, the decision to demonstrate property A is not based on knowledge of the choices of their neighbours. This assumption is potentially false in some places, but as we noted earlier, ASes share little of their internal business decision making with competitors, and so is a reasonable assumption. Moreover the assumption is not needed at all for ASes that have a uniform policy (e.g., display property A at all times).

On the other hand, if **any** AS displays property A then the path will show A. These two cases give the likelihood model for some dataset D given the probability vector $\mathbf{q}$, containing q_i for each AS in the dataset. That is, the likelihood that we observe our data D given some set of values for our ASes q_i.

$$\mathbf{P}(J|\mathbf{q}) = \begin{cases} \prod_{i \in J} q_i, & \text{if path does not show A,} \\ 1 - \prod_{i \in J} q_i, & \text{if path shows A.} \end{cases} \tag{7.5}$$

Then,

$$\mathbf{P}(D|\mathbf{q}) = \prod_{J \in D} \mathbf{P}(J|\mathbf{q}), \tag{7.6}$$

describes the likelihood that we observe our data D given the set of values q_i for the ASes under consideration.

While it is easier to write the likelihood in terms of q_i to mirror the well established binary tomography problem, for many cases we are interested in p_i, the probability of an AS displaying a certain property. The rest of this chapter will look at inferring p_i, although the same applies to q_i using the simple relationship $p_i = 1 - q_i$.

It may be easier to think of p_i as a *proportion* of routes through AS i that will be observed to show property A. This might not be 1 or 0 because the AS is heterogeneous[3] or because it uses parameters that only trigger property A under unusual circumstances or at certain times. It also allows for measurement errors and sparsity.

In a frequentist approach, one might seek a $\hat{\mathbf{p}}$ to maximise the likelihood given in Equation 7.6; however, as we are in a probabilistic setting there is scope to instead infer the distribution of p_i given the observed data, denoted $P(\mathbf{p}|D)$.

The basic probabilistic model introduced in Equation 7.6 mirrors the binary tomography problem in Section 7.1.2. However, the procedure can be generalised to other likelihood models. For example, we incorporate specific types of errors in the measurements in Section 7.6.

The likelihood in Equation 7.6 is a variant of the Poisson binomial distribution, and so we cannot find analytical results for $P(\mathbf{p}|D)$. Computational Bayesian methods are required to sample from the posterior distribution.

MCMC methods are a natural choice in this context as they are ideal when it is easy to simulate potential solutions and calculate their likelihood, but difficult to find the optimal solution.

[3]Heterogeneous ASes implement different routing policies to different neighbours.

Here we will use MCMC to infer the posterior $P(\mathbf{p}|D)$. The dataset will be the set of all measured paths and their corresponding property, explained in more detail in Section 7.3. As always, Bayes' rule converts this into the form required for inference.

$$P(\mathbf{p}|D) \propto P(D|\mathbf{p})P(\mathbf{p}), \tag{7.7}$$

where $P(D|\mathbf{p})$ is a likelihood model associated with the data as described in Equation 7.6 and $P(\mathbf{p})$ is the prior knowledge of the parameters.

Here we use two well known MCMC methods, Metropolis-Hastings [64, 92] and Hamiltonian Monte Carlo [39], introduced in Section 2.3, and summarised here.

The Metropolis-Hastings (MH) algorithm creates a Markov chain to explore the space of interest $P(\mathbf{p}|D)$. The proposal distribution $Q(\mathbf{p}'|\mathbf{p})$ proposes the next candidate for the probability vector $\mathbf{p}'$ from the current state $\mathbf{p}$, which is accepted or rejected in a Metropolis update step. We use component-wise MH and update a single parameter in the vector $\mathbf{p}'$ at each step.

The proposed parameter value $\mathbf{p}'$ is accepted with some probability given by, in the case of M-H, the acceptance probability

$$\alpha = \min\left(1, \frac{P(\mathbf{p}'|D)Q(\mathbf{p}|\mathbf{p}')}{P(\mathbf{p}|D)Q(\mathbf{p}'|\mathbf{p})}\right). \tag{7.8}$$

Substituting Equation 7.7, the α acceptance probability is easily calculable.

Hamiltonian Monte Carlo (HMC) is closely related to Metropolis Hastings and uses Hamiltonian dynamics to explore the space by translating the density function of interest into a potential energy function and including a momentum variable [16]. The method uses a Markov Chain as in MH but new candidates are proposed by propagating the current state along a Hamiltonian trajectory using a Gaussian distributed momentum parameter. To obtain the samples of interest the auxiliary momentum parameters are ignored (marginalised over). This allows for multidimensional updates and allows the sampler to escape from local optima. HMC also uses a Metropolis update, and the acceptance probability uses the ratio of the auxiliary distribution of both the parameters of interest and the momentum.

In both MH and HMC the chain is generated from the proposed parameter $\mathbf{p}''$ as follows

$$\mathbf{p}^{(t+1)} = \begin{cases} \mathbf{p}', & \text{with probability } \alpha, \\ \mathbf{p}^{(t)}, & \text{otherwise,} \end{cases} \tag{7.9}$$

where $\mathbf{p}'$ is generated from either the MH or HMC proposal and α is the corresponding update probability. We implement both these algorithms in Python and utilise the well regarded PyStan [129] package to implement our HMC sampler.

We have introduced this algorithm with both the simple MH sampler as well as HMC to improve mixing in the high dimensional space. As expected, we find that both MH and HMC samplers give the same results. As HMC can provide updates in many dimensions, is less likely to get stuck in local optima, and is faster, most of the results are demonstrated with the HMC posterior samples. There is also scope to extend this method to use other MCMC techniques such as Sequential Monte Carlo, as discussed in Chapter 4 for online inference or other likelihood models.

To finalise the algorithm we must also decide on a prior distribution $P(\mathbf{p})$ for Equation 7.7. This provides some flexibility of the method to incorporate what prior knowledge we have about the system. In the RFD case, for example, the nodes that initiate the measurement are known not to display RFD. We present results for a uniform prior distribution on $[0, 1]$ in each dimension for MH inference. Additionally, as this is a relaxation of the binary tomography problem, we expect nodes to be either showing the property or not,so we use a symmetric beta prior distribution (with parameters $a = b = 0.2$) with mass at zero and one. We use this prior for our HMC estimates. We find that as there is sufficient data in the BGP setting for most ASes, the prior does not strongly influence the results in this case, but a non-uniform prior can utilise underlying knowledge to provide a good signal where there is insufficient data.

BeCAUse generates samples from our distribution of interest using only a likelihood model, path data measurements and the prior. There is no requirement of ground truth for training, which is useful because there is very little ground truth available. The level of certainty about the inferences are implicit in the distributions and allows for informed decision making based on the desired application. In Section 7.5 we highlight how the samples from $P(\mathbf{p}|D)$ can can be used to identify RFD-enabled ASes from Internet path data.

7.3 Data

In the previous section we introduced a framework to infer node properties from path data. To locate ASes that employ RFD we need path data in which paths that display RFD are labelled as such. The datasets we use were collected using active measurement by Mosig et al. [59]. To collect path data Mosig et al. set up 7 beacons across the globe. These beacons make announcements and withdrawals at certain frequencies. The announcement and withdrawals are passed along paths to the vantage points that measure the resulting pattern. RFD has a distinctive pattern caused by the suppression of the signal. If RFD is employed by any AS

Table 7.1: Excerpt of dataset collected in Mosig et al. for use in BeCAUSe for an update interval of 2 minutes.

Vantage point IP	Prefix	Path	RFD
102.130.50.5	147.28.48.0/24	[6233, 6939, 16735, 22548, 58364]	False
103.102.5.1	147.28.32.0/24	[131477, 57695, 3223, 1299, 2497, 58361]	False
217.29.66.225	147.28.44.0/24	[50877, 37100, 58363]	True
217.29.66.120	147.28.36.0/24	[15605, 12874, 3303, 2914, 3130]	True

Table 7.2: Summary of datasets for each update interval.

Update Interval	Number of ASes	Number of paths	Percentage of RFD paths
1 minute	574	8019	18.3
2 minutes	580	9021	13.7
3 minutes	581	9643	6.3
5 minutes	588	10730	4.1
10 minutes	591	11303	0.8
15 minutes	597	11474	0.4

on this path then this pattern will be observed. The vantage point also receives a record of the path taken from the beacon. Paths that display this pattern are labelled RFD. An excerpt of the dataset can be seen in Table 7.1.

Update intervals ASes can deploy RFD with various parameters to control when damping occurs. One important parameter is the update interval at which nodes begin to employ damping. That is, at what frequency of announcements and withdrawals does the AS begin to damp the signal. It is understood that vendor default RFD parameters will start damping if a router flaps at least every 8 or 9 minutes [59]. The recommended parameters would trigger for flaps in a 2 minutes interval. Understanding the deployment of recommended parameters is very useful to the Internet community. Therefore, the data is measured with update intervals 1, 2, 3, 5, 10 and 15 minutes. We expect that more ASes will employ damping at the lower update intervals.

Full details of the measurement set-up as well as labelling and validation methods can be found in [59]. One of the benefits of the framework introduced here is the general applicability. It is not necessary to know particulars about RFD itself, only that our model in Equation 7.6 is reasonable for this application.

7.4 Inferring RFD nodes

The posterior distribution $P(\mathbf{p}|D)$ of interest is high dimensional, but we are interested in identifying which individual nodes are displaying RFD. Therefore, we look at the marginal distributions of each AS, i.e., distribution of each p_i, to identify RFD-enabled nodes. These distributions provide us with a wealth of information; most notably, how likely AS i is to display RFD as well as a confidence level in our estimate. We will also see that these distributions can identify which ASes employ RFD inconsistently. Ultimately, the desired output for many researchers in this space is Boolean: RFD-enabled or not. In the following sections we describe a framework to summarise the distributions to identify RFD nodes while maintaining some of the uncertainty information from the distributional outputs.

7.4.1 Algorithm output

To identify RFD-enabled ASes using BeCAUSe we must establish marginal distributions that are indicative of RFD. Here, we highlight the diagnostic ability of these distributions, and describe a basic summarisation and classification process to provide automatic insights.

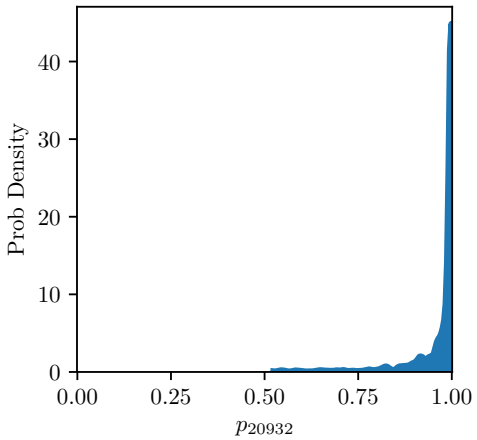
Figure 7.2 depicts the marginal distributions of four ASes that are indicative of behaviours of interest.

a. The distribution is heavily skewed with most mass at 1. There is very little spread suggesting there is a lot of evidence that the node has high probability of damping.

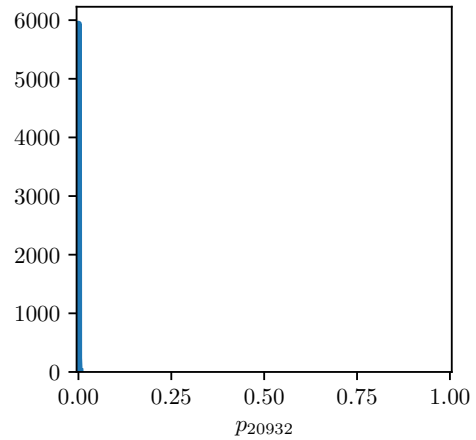
b. The distribution is heavily skewed with most mass at 0. There is very little spread suggesting there is a lot of evidence that the node has low probability of damping.

c. The mass is centred around mean 0.1. This suggests there could be contradictory information about this AS. In fact, AS701 damps inconsistently.

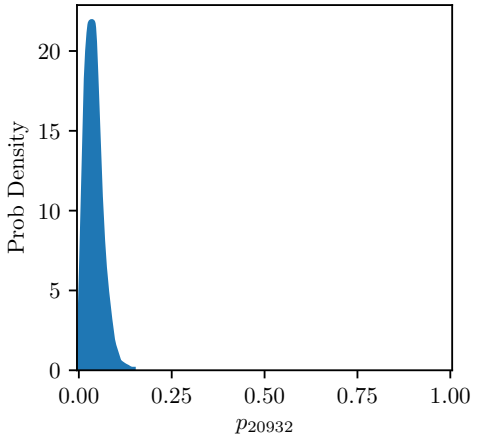
d. The distribution we see here is the prior (i.e., what we thought before we saw any data). Therefore, it is likely that we didn't see any 'meaningful' data about this AS. Interestingly, this AS is on damped paths; however there is already another node on these paths that has a high chance of damping, so we do not have any additional information about this node.



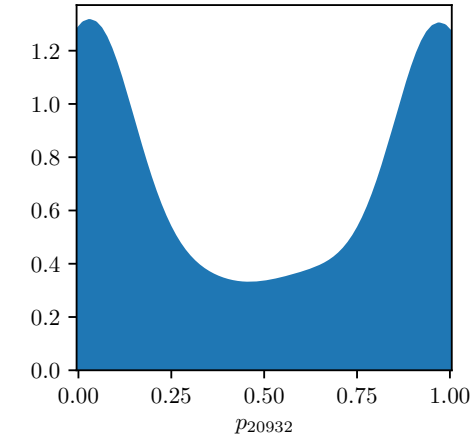
(a) Probability distribution of RFD of AS: 20932. Mass at 1 indicative of RFD.



(b) Probability distribution of RFD of AS: 2497. Mass at 0 indicative of not RFD.



(c) Probability distribution of RFD of AS: 701. Not straightforward, the mean ~ 0.1 and higher spread indicate contradictory data.



(d) Probability distribution of RFD of AS: 12874. Very similar distribution as beta prior suggests low evidence about this AS.

Figure 7.2: Example output distributions demonstrating their clear diagnostic ability.

Summarising the distributions

Inferring the distributions of each AS gives us a great deal of flexibility in identifying ASes that are implementing RFD. However, in many cases, the desired outcome is a binary classification of nodes that are displaying RFD. There are a range of methods, ranging from simple to complex that we could use. The most straightforward is taking summary statistics of the distributions. Thresholding can then be used to determine likely RFD-enabled ASes. We can also preserve some of the information regarding the shape or spread of the distributions to use the implicit level of certainty in our decisions. We will focus here on one possible way of summarising the distributions with two metrics to measure the expected value and the certainty and use these to categorise the ASes and identify RFD.

Metric generation We generate 2 summaries of the distribution for AS from each method:

- The **mean** of the distribution; and
- The **Highest Posterior Density Interval** (HDPI)

The first is just the average $\bar{p}_i$ of the distribution, $P(p_i|D)$, and gives an estimate of the expected value for this AS. The second, HDPI, finds the smallest interval that contains $\gamma = 0.95$ proportion of the mass. Otherwise known as the smallest Bayesian credible interval, it is the interval [A,B] where γ of the mass falls between A and B such that $B - A$ is minimised. The HDPI gives an idea of the 'spread' of the distribution.

Step 1) Categorising While these metrics also provide us with some valuable information we wish to translate these metrics into a 'decision'. We maintain some of the information about certainty by mapping results to a rating from 1-5, where 1 and 2 are highly likely and likely not damping and 4 and 5 are likely and highly likely damping. Category 3 is unsure, either because of contradictory data, or, most often, due to lack of specific data about this AS. Note that not enough data does not necessarily mean the AS is not on many paths. When a node is on paths with another damping node it can be hidden and we may not have any specific information about this particular node.

We assign categories for each AS from the mean and HDPI, see Table 7.3 for assignments. After summarising and categorising both the MH and HMC distributions by mean and HDPI, we use the highest flag.

Table 7.3: Categories based on distribution summaries for each AS. 'Else' indicates the flag if no other category is assigned. The highest category is chosen for each AS.

	Average: $\bar{p}_i$	HDPI: $[A_i, B_i]$
Category 1	$\bar{p}_i \in [0, 0.15)$	$B_i \in [0, 0.15)$
Category 2	$\bar{p}_i \in [0.15, 0.3)$	$B_i \in [0.15, 0.3)$
Category 3	$\bar{p}_i \in [0.3, 0.7)$	else
Category 4	$\bar{p}_i \in [0.7, 0.85)$	$A_i \in [0.7, 0.85)$
Category 5	$\bar{p}_i \in [0.85, 1]$	$A_i \in [0.85, 1]$

We note here that while the intervals were chosen to mimic the diagnostic ability of the output distributions, the requirement to convert these distributions to categories mean these exact values are somewhat arbitrary. We present a data driven explanation for these particular intervals for the mean in Section 7.5. The categories for the HPDI given above are unlikely to be binding as the mean will produce the same, if not a higher, flag. We could change the values depending on the application; however, we refrain from choosing an arbitrary value.

In general we accept Category 4 and 5 to be an RFD enabled AS; however, if higher certainty was required just Category 5 could be used, or categories could be assigned by exclusively using the HDPI.

Step 2) Identifying nodes that RFD inconsistently It is evident from the data, and well known within the network operator community, that some ASes RFD inconsistently. For example, AS701 damps all neighbours except AS2497. The distribution of p_{701} in Figure 7.2c highlighted contradictory data. One way to overcome this would be to infer whether each link is RFD-enabled rather than the nodes. We address this possibility in Section 7.6, but here we use a simple way to identify many of these nodes using the marginal posterior distributions of the parameters.

Recall, that if the path displays RFD then there is at least one node on the path that damps. Therefore, after we have assigned the categories using the previous method there should be at least one path on each RFD path that is labelled in Category 4 or 5. If the path does not contain an inferred RFD node, we use the posterior marginal distributions to determine the node that is most likely causing RFD on this path. Specifically, for each node X on the path we determine the posterior probability that node X is the most likely to be causing RFD on the path. If

$$\text{Prob}\{\min(p_i \text{ for } i \in J) = X\} > 0.8, \tag{7.10}$$

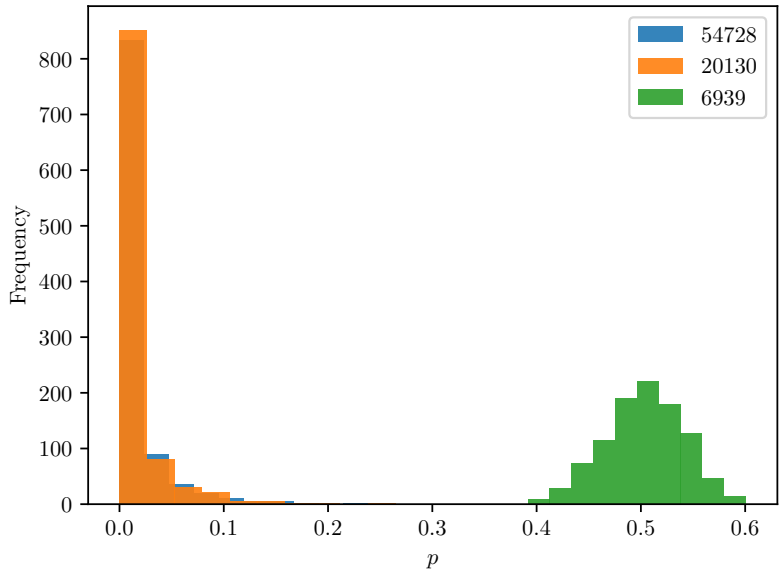


Figure 7.3: Posterior distribution of ASes on the RFD path [54728, 20130, 6939]. Although $\bar{p}_{6939}$ is around 0.5, it is the most likely contributor to the dampening occurring on this path. This is indicative of an AS that is damping inconsistently.

then there is sufficient evidence that X is the damping node on path J. For each damping path J in the data, if there is a node that is most likely damping we label this node as Category 4. Compared to the thresholding step this process is less robust to noise in the data as a single incorrect labelling (e.g., missing an inconsistent node) can lead to another node being labelled to account for the path label.

Figure 7.3 shows the marginal distributions for an RFD path containing ASes 54728, 20130 and 6939. While none of the marginals have any significant mass at 1, AS3303 is the most likely node to cause damping.

$$\text{Prob}(\min(p_i \text{ for } i \in [54728, 20130, 6939]) = p_{6939}) \approx 1. \tag{7.11}$$

Therefore, we label AS6939 as Category 4. This method is also flexible. We could require more certainty and only allocate Category 4 if it is causing damping on more than Y paths; however, this introduces another arbitrary choice.

One might argue that by summarising the distributions we are losing information that we have worked hard to obtain. Indeed this is true, but a decision is still required to solve the initial problem — identifying nodes that display RFD. We

also find that categories are more well received by the Internet community who can easily use these results to understand the network wide deployment of RFD.

Nonetheless, the distributions also provide a high level of flexibility not only in how they can be used to make a final decision on RFD, but also on the insights we can elucidate from them. In the next section we will see that some nodes retain the prior distribution as the posterior. This highlights that data about this node is lacking, and can elucidate where to focus our efforts to collect more data.

7.5 Results

We run the BeCAUSe algorithm to take samples from the posterior $P(\mathbf{p}|D)$, using the path data as described above. Using the summarisation of the marginal distributions each node is assigned to a category, 1-5.

Figure 7.4 summarises and highlights the output of the algorithm. The scatterplot uses two summary metrics of the output distributions — mean and length of HDPI. Specifically, we show 1-HDPI to make higher values more certain. The x-axis is a measure of how likely an AS is to show RFD, and the y-axis is a measure of how sure we are of our x-axis estimate. The nodes are coloured by category determined by the process introduced above. There is a characteristic U shape. On the right we have nodes that are likely to RFD as shown by the high average $\bar{p}_i$. We find that ASes in the top left are on a large number of RFD paths and so have a high certainty, e.g., AS20932 (Figure 7.2a). For nodes where there is less data (but not contradictory), we see our confidence decrease but the average $\bar{p}_i$ still suggests RFD. Conversely, on the top left we have nodes with a lot of evidence for not RFD, e.g., AS2497 (Figure 7.2b). For nodes that are on less paths we have less information, but the average remains high as we see in the Category 1 and 2 ASes. At the bottom of the plot we see ASes that we do not have much information about and recover the prior with high spread, e.g., AS12874 (Figure 7.2d). The category 4 ASes spread over the centre are quite interesting. It suggests we have high evidence that there is inconsistent damping occurring, largely due to contradictory information. Despite this, our flagging method has identified these nodes as sometimes RFD, e.g., AS701 (Figure 7.2a). This plot is indicative of the other update intervals, and we see similar trends, although the strictly higher update intervals have less RFD nodes.

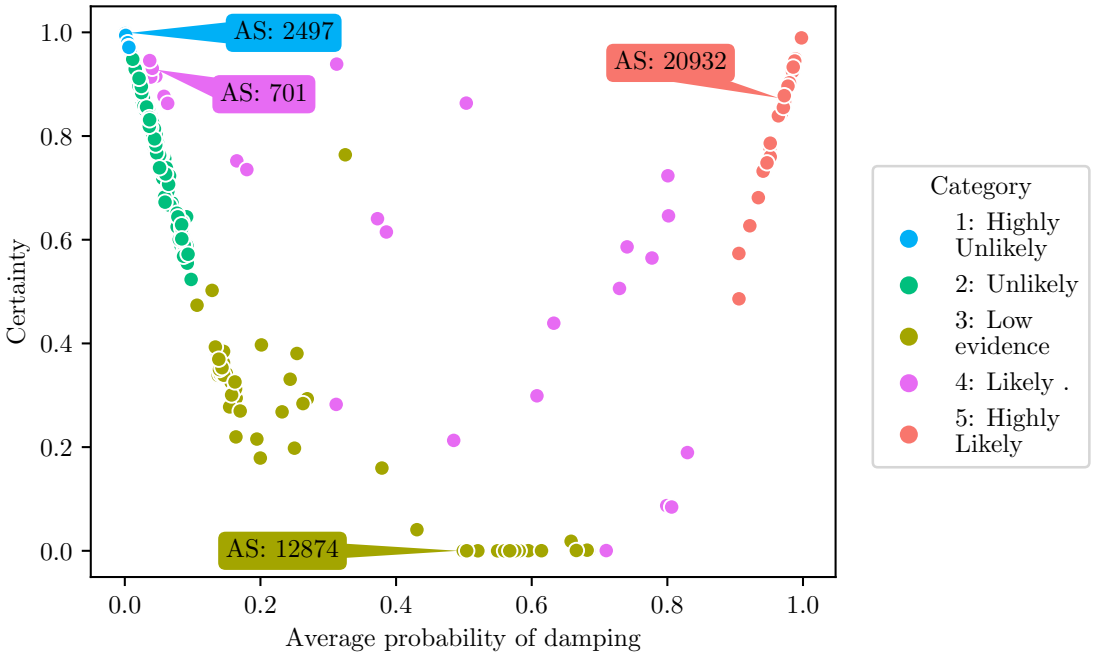


Figure 7.4: Scatter plot of the mean of the marginal posterior distribution for each AS and a measure of certainty about the estimate. Here we use the length of HDPI, and plot 1−HDPI to ensure high values are more certain Coloured by the category. The x-axis is a measure of how likely ASes are to show RFD, and the y-axis is a measure of how sure we are of our x-axis estimate.

7.5.1 Comparison to ground truth

We validate our findings using ground truth from 24 ASes, acquired directly from the network operators. In general, exact configurations were not disclosed and it is difficult to map the provided configurations to exact update intervals from our experiment. Nodes that damp in higher intervals will also damp those at lower intervals, but not necessarily the converse, so we chose to compare our results for the smallest update interval (1 minute) to ground truth. Overall, BeCAUSe performs very well on this dataset with **100% precision**. The **recall is 83%** mainly due to visibility issues in the data that result in some nodes hiding behind other damping nodes.

7.5.2 Different update intervals

As highlighted in Section 7.3, there are 6 datasets for different update intervals. We use the method described above to flag nodes in each dataset as RFD or not. Figure 7.5 shows the number of flagged RFD nodes for each update interval. The orange bars indicate the nodes that are consistently RFD-enabled for which we have non-contradicting data, i.e., damping all neighbours consistently. These ASes have been labelled using only the probability of damping (step (1) Section 7.5). The blue bars include inconsistently damping ASes that were labelled with step (2).

An AS with deprecated default values would start damping at the 5 minute update interval, and as expected, we observe a single sharp incline at 5 minutes. The very few damping ASes at 10 or 15 minutes are likely induced by updates being amplified by anomalies in the network structure. We expect the continuous increase of RFD ASes for the smaller update intervals due to network operators following the current recommendations. However, we observe an unexpected spike at 2 minutes. This is an unexpected impact of inconsistent nodes. Many nodes are heterogeneous, i.e., they damp inconsistently. There is an implicit assumption in our thresholding procedure that nodes with higher posterior probability of damping are more likely to be damping than those with lower posterior probability. However, nodes that damp inconsistently can violate this assumption. For example, AS1299 is a prolific node in all datasets and is on many paths, but only damps a few of its neighbours. For the update interval of 2 minutes it is on many (2,508) undamped paths and only 291 damped paths. This gives AS1299 a very low probability of damping in the posterior. Therefore, our categorisation does not label the node as RFD. This misclassification can lead to other nodes labelled as RFD in step 2 to account for the marked paths in the dataset. Hence the spike

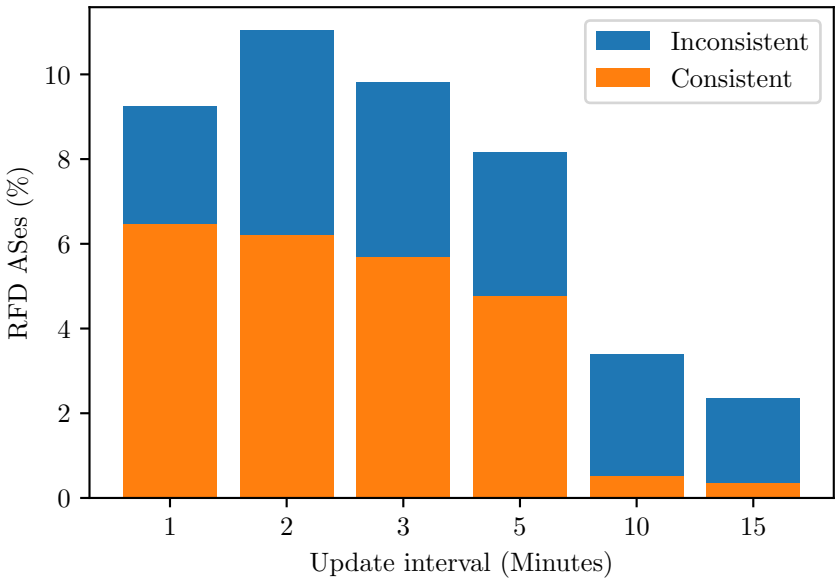


Figure 7.5: Histogram of the number of RFD nodes in each update interval. Orange shows the consistent dampers, blue includes the inconsistent dampers.

at 2 minutes in Figure 7.5. To attempt to address this limitation we introduce a model that includes errors and discuss the possibility of inferring damping links instead of nodes in the upcoming sections.

Currently we infer parameters for each dataset independently. However, as there is dependence between datasets — nodes that damp at a higher update interval will also damp at the lower update intervals — there is scope to use outputs from one dataset as a prior for the next. Additionally, we could create one model that incorporates the relationship between update intervals and utilise all the datasets.

7.6 Extensions

7.6.1 Incorporating errors: a new model

The basic model introduced in Section 7.2 assumes all data recorded to be correct. However, the measurements setup labels data by observing indicative RFD patterns. Paths are labelled RFD only when the indicative pattern is observed. When it does not show this pattern then it becomes an unmarked path. Additionally, if

the pattern is unclear it is left at an unmarked path. We have highlighted that ASes damp inconsistently; therefore, it is possible that there are RFD nodes on paths that are labelled non-RFD.

Unmarked paths have some small probability of having an RFD node. We assume that the converse is not true. Due to the stringent testing criteria for the RFD pattern we are very confident that marked paths have an RFD-enabled node on them. We can model this one way error.

Consider the probability that we observe a path with result O while the 'truth' of the path is T:

$$\begin{aligned} P(O = RFD|T = RFD) &= 1 && (7.12)\\ P(O = RFD|T = nRFD) &= 0 && (7.13)\\ P(O = nRFD|T = nRFD) &= 1 - \epsilon && (7.14)\\ P(O = nRFD|T = RFD) &= \epsilon && (7.15) \end{aligned}$$

Where RFD is a marked path and $nRFD$ is an unmarked path.

By the law of total probability:

$$P(O) = P(O|RFD)P(RFD) + P(O|nRFD)P(nRFD). \tag{7.16}$$

So now the new model for a single observed path in the data

$$\begin{aligned} P(D|\mathbf{p}) &= \prod_{J\in D} \begin{cases} \epsilon P(RFD) + (1-\epsilon)P(nRFD) \text{ if RFD is not observed,} \\ 1 - P(nRFD) \text{ if RFD is observed.} \end{cases} && (7.17)\\ &= \prod_{J\in D} \begin{cases} \epsilon + (1-2\epsilon)\prod_{i\in J} p_i \text{ if path does not show A,} \\ 1 - \prod_{i\in J} p_i \text{ if path shows A.} \end{cases} && (7.18) \end{aligned}$$

In short, this gives some small probability that the observation of the unmarked path was wrong.

The results from this model are similar to the results above, but the marginal posterior mass is shifted right, towards RFD, in most cases. As an example, Figure 7.6 shows some ASes that change category in the new model. We observe that there is a shift in mass towards RFD. Notably, ASes that had contradicting data in the original model display more evidence for RFD. This is exactly the desired outcome. There are some (e.g., AS25548) that have more uncertainty. This is due to another AS on the same paths having an increased probability of RFD which slightly decreases these ASes. As the model incorporates errors it is

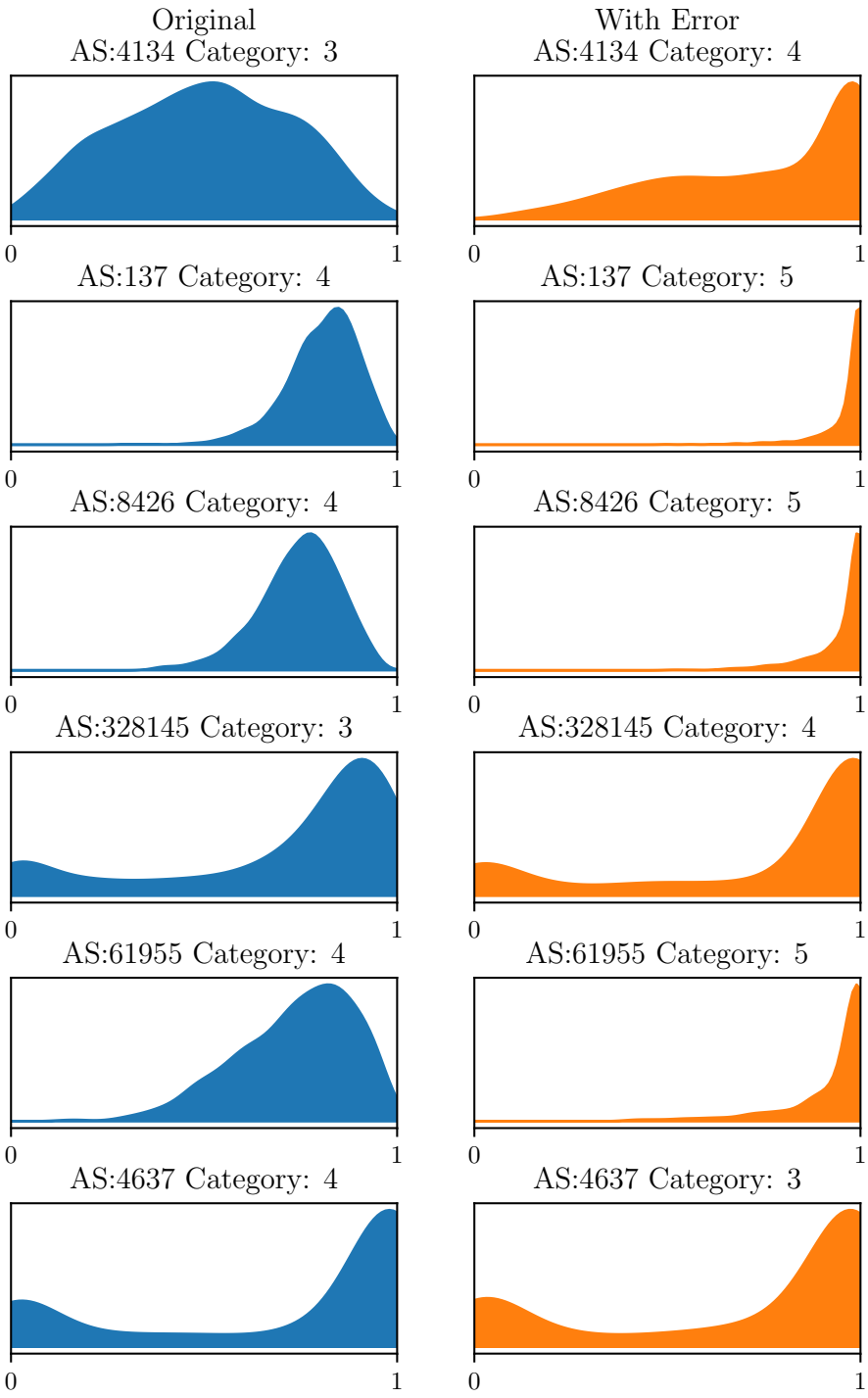


Figure 7.6: Marginal posterior distributions for the original likelihood model (blue) and the model incorporating errors (orange). We choose a sample of nodes that changed category when incorporating errors.

more realistic; however, it does not result in significant differences in our overall inferences.

There is one extra parameter, ϵ, that controls the proportion of observations that are unmarked but contain an RFD node. In the experiments above we used $\epsilon = 0.2$. We attempted to infer ϵ from the data; however, the likelihood is maximised when $\epsilon = 1$. Further work could include finding a data driven estimate for ϵ.

7.6.2 Route Origin Validation

The methodology we have introduced generalises to solve other tomography problems using path data. Route origin validation is a process ASes can employ to prevent unintentional announcements of routes [93]. The exact definition of ROV is not so important for this analysis, only that if at least one node on the path is employing ROV then the entire path is labelled ROV.

To test the method and locate ROV we use a subset of the path data used for RFD and relabel the paths based on ROV nodes. Unlike the RFD dataset described above, whether or not paths show ROV has not been measured. However, we do have ground truth data available for this dataset so we can manually label the paths as ROV. Of course, this is an ideal situation without noise. Nonetheless, it provides an opportunity to demonstrate the algorithm on a different type of network tomography problem. The dataset we have contains ∼2,500 paths with a total of 568 ASes of which 39 are ROV-enabled. There are two key differences in contrast to the RFD dataset: *(i)* 90% of paths are labelled ROV (versus 18% for RFD) and *(ii)* noise is absent. More marked paths are likely to result in more hidden or unidentifiable nodes.

We run the algorithm on the ROV path data, and allocate categories as described above. Figure 7.7 shows the posterior marginal distribution for the set of ground truth ROV-enabled and the set of nodes. The correctly recovered nodes are shaded green and the nodes that were missed are white. There were no false positives in this case, giving **precision of 100%**; the **recall is 64%**. From Figure 7.7 it is evident that the nodes that are not recovered have returned the prior distribution, suggesting we did not see meaningful information about these nodes. In fact, the nodes that were missed were only seen on marked paths that had another ROV node on the path. Therefore, the inability to recover these nodes is a consequence of the dataset not the method. The method highlights the nodes for which we have little information, and so is useful in determining where we may want to collect more data.

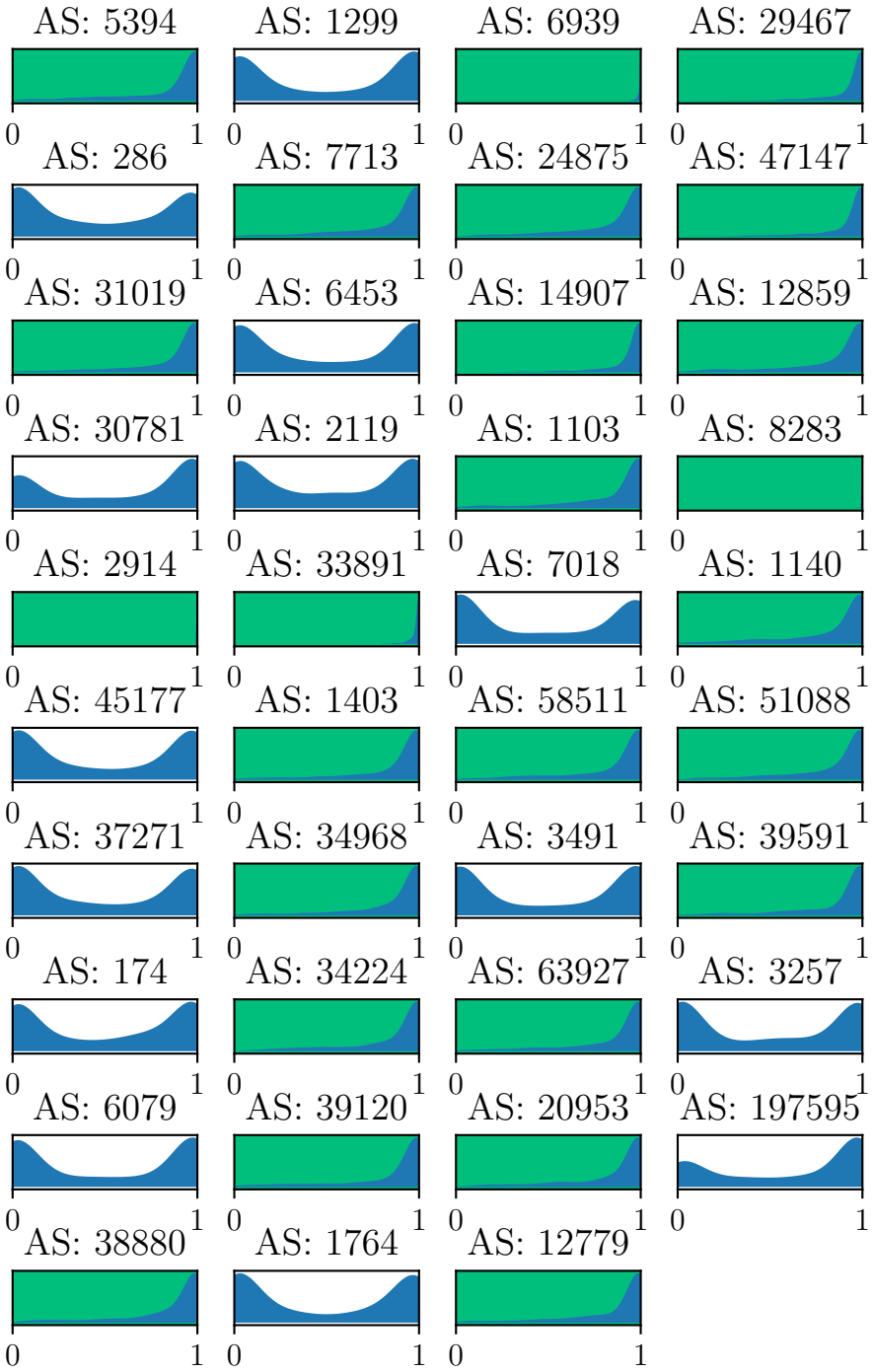


Figure 7.7: Marginal posterior distributions of the nodes that are ROV-enabled. Distributions with a green background were inferred as ROV-enabled as the posterior mass is near 1. Those with a white background were labelled as *unsure* and display the prior distribution as there was insufficient meaningful data.

7.6.3 Towards more robust inference of inconsistent dampers: link inference

Throughout this chapter we have highlighted that, as we assume each AS is a single entity, inconsistent dampers are harder to identify in the current framework. This is largely because some nodes damp only certain neighbours, perhaps because there have been issues with this neighbour in the past. It is also possible that ASes have other more complex configurations that produce inconsistent damping. A natural extension is to model the impact of different neighbours explicitly by inferring link properties rather than node properties. I.e., infer p_{ij} for each AS pair ij in the dataset. In this section we will show basic results for link inference and discuss potential ways to utilise both the rich dataset we have for node inference and the flexibility that link inference provides.

We implement the MH sampler to infer link properties[4]. As using link inference should account for inconsistent dampers, we only use categorisation step 1 using thresholds to give the edges a category of 1-5. Figure 7.8 shows the scatterplot of mean and length of HDPI for the link inference. Again, the nodes are coloured by category. Links with very little data are at the base of the 'V' shape. As more consistent data is observed, the mean shifts and the nodes increase their certainty, moving the nodes 'up' the V. We note only a few nodes do not follow the V shape; these nodes are candidates for inconsistent dampers.

Comparison to ground truth The aim of this project is to identify ASes that employ RFD to any extent. Therefore, we aggregate the link results to nodes by categorising nodes as the highest category of the emanating edges. As this method accounts for inconsistent dampers, even without the second categorisation step, the precision is 100%. For example, AS701 only damps AS2914. When inferring nodes AS701 had low evidence of damping but was identified in our second categorisation step; however in this link inference edge (AS701, AS2914) has strong evidence of damping, while the other neighbours had low damping probability. As one might expect with the increased sparsity of data, the recall is less than for the node inference at 45%.

Relationship between update intervals Earlier we discussed that the inconsistent dampers caused an unexpected spike in the number of nodes labelled as

[4]We attempted to implement HMC; however the sampler did not finish in 3 days. As the MH and HMC algorithms in the original analysis gave indistinguishable results we use just the MH results here. We leave further computation or optimisation of the HMC parameters as future work.

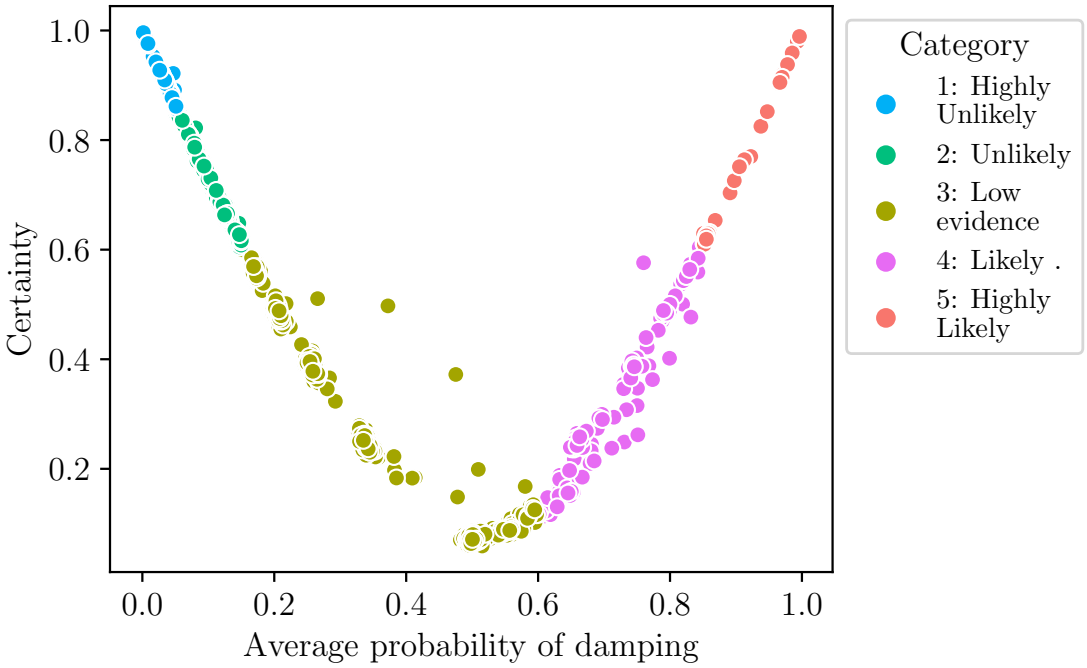


Figure 7.8: Scatter plot of the mean of the marginal posterior distribution for each link and a measure of certainty about the estimate. Here we use the length of HDPI, and plot 1−HDPI to ensure high values are more certain Coloured by the category. The x-axis is a measure of how likely ASes are to show RFD, and the y-axis is a measure of how sure we are of our x-axis estimate. The 'V' shape, in comparison to the 'U' shape observed for the node analysis, is due to the use of a uniform prior rather than a beta prior.

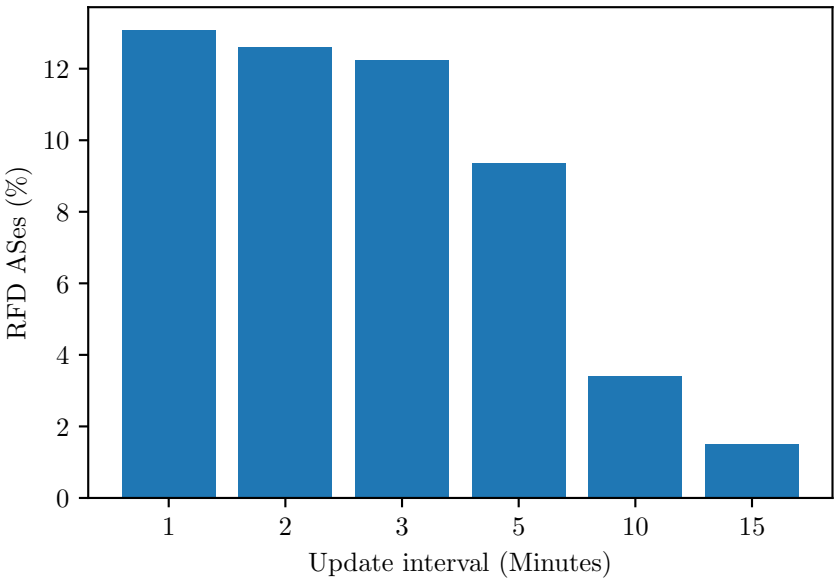


Figure 7.9: Histogram of the number of RFD nodes in each update interval.

damping in the dataset with an update interval of 2 minutes. Figure 7.9 shows the number of nodes flagged as damping using edge inference. We can see the expected trend of monotonically decreasing number of dampers as the update interval increases. Additionally, nodes inferred to be damping at higher intervals are a subset of those at higher intervals (not shown).

Are there still inconsistencies? Interestingly, there are still inconsistencies in the data. Easily seen in Figure 7.8 for an update interval of 1, there are some links that behave inconsistently and appear 'above' the V. This phenomenon becomes more common for larger update intervals. This highlights that the complex nature of AS protocols and RFD implementations cannot be completely explained by the neighbour from which the signal comes. We could again use Step 2 in our categorisation method to flag some of these nodes if required in practice.

7.7 Discussion

Identifiability & hidden nodes Identifiability has a specific meaning in statistics in terms of being able to infer each parameter of some underlying model. Here,

nodes become unidentifiable if there are two nodes that are statistically equivalent. That is, we can swap their values and the observations remain the same. In our case, when two nodes are only ever on the same marked paths either of them could be RFD-enabled without changing the observed path, and we cannot determine which without more data. We expect both of these nodes to return the same distribution, most likely the prior. If two nodes were on unmarked paths then, although they are equivalent, we label them both as non-RFD. In the RFD results we do not see this very often as there are sufficient alternate routes that nodes are not often only observed together. However, we observe this often in the ROV dataset where nodes with the property of interest are more common, and so we see many more nodes with the prior distribution as the posterior.

A more common scenario is to have 'hidden' nodes in a dataset. Commonly, there are nodes that are only on marked paths, but there are other nodes on the path that are more likely RFD-enabled. Naive heuristics may label these nodes as RFD; however, they are not causing the damping. The MCMC methods consider the system as a whole and these nodes return the prior. These hidden nodes highlight a lack of data and are useful to identify where we could collect more data to improve our inferences.

Choosing the prior The prior distribution is our knowledge of the system before we observe any data. In the HMC implementation we use a symmetric beta prior to put mass equally at 0 and 1 and less in between as this problem is a relaxation of the binary tomography problem. We implemented other priors: uniform $U(0, 1)$, where we have no prior information, Beta$(1, 3)$ with mass at 0 to have the prior assumption that no nodes are damping and Beta$(3, 1)$ with mass at 1 to have the prior assume all nodes are damping. We find that, as expected, the prior impacts nodes that are not seen often in the data. There is higher variance in posterior distributions of most nodes when using a uniform prior. Despite this, in the RFD case there is sufficient data for most nodes that the prior distributions does not impact the category assignment of the marginal distributions. As we observed in Section 7.6 many posterior marginals in the ROV results were the prior distribution. Altering the prior distribution in this case, of course, alters the posteriors; however, as we have the extra step of categorising nodes it did not change precision or recall.

Computation We have implemented both MH and HMC to compute the posterior distribution. We use a 2x Intel Gold 6148, 2.4GHz, 384GB memory. HMC requires less computation overall as there is less dependence between samples at each iteration. The PyStan implementation takes $\sim$10 hours per dataset. Despite

this, the MH algorithm can be run in parallel to take independent samples. A single run of the MH algorithm with 1 million iterations takes ∼2 hours. Both methods are at least linear in the number of paths in the data and number of parameters in the dataset. The PyStan tuning algorithm can increase with changes in the shape of the posterior space, we refer the reader to the PyStan manual [129].

Combining node and link inference While the pure link inference suffers from data sparsity, we have shown that the idea has potential for improving the inference of RFD enabled nodes. The node inference we saw in Section 7.5 had a large amount of data and was quite successful; however, inconsistent nodes affected the results. Link inference overcame this problem but suffers from data sparsity and an increase in complexity. Many ASes behave consistently (either RFD or not). Therefore, there is no need to infer each link connected to these nodes individually. Conversely, as we have seen, some nodes behave inconsistently, and in these cases considering the links along which damping occurs would be beneficial for both inference and understanding of the system. Therefore, a hybrid approach could be beneficial. We can first identify potential inconsistent dampers, either from domain knowledge, using raw data or running the inference algorithm. Then, we can infer a combination of node and link parameters using the method above. This may provide a balance between available data and incorporating inconsistent damping protocols. Alternatively, we could infer links for the most prominent nodes, anticipating that there will be sufficient data about these nodes.

7.8 Conclusion

In this chapter we introduced BeCAUSe, an algorithmic framework to infer node properties from Internet path data. In contrast to heuristic methods, which are commonly applied to tackle this challenge, this method does not make restrictions on the topology setup or require specific knowledge about the property of interest. We demonstrated BeCAUSe to pinpoint autonomous systems that deploy route flap damping and route origin validation, and highlighted interesting extensions by including measurement errors and inferring both node and link properties. This chapter highlighted yet another problem in network science — network tomography — that can benefit from the uncertainty quantification and flexibility of Bayesian inference methods.

Chapter 8

Conclusion

This thesis addresses key problems in network science from a computational Bayesian viewpoint, the combination of which provides novel algorithms through which to gain understanding about the systems of interest.

In Chapter 3 we addressed network simulation to take samples from the underlying conditional probability distribution $P(G|G \text{ has some properties})$. We studied the connected Waxman ensemble in detail and proved convergence to the desired ensemble as well as evidence for finite time convergence in $\mathcal{O}(N^2)$. We improved on this convergence complexity using a new proposal method and give evidence for convergence in $\mathcal{O}(E)$ in Chapter 4. This method can be extended to sample from a general ensemble of the form $P(G|G \text{ has some properties})$, and we introduced a likelihood annealed Sequential Monte Carlo set-up to address more complex ensembles.

In Chapter 5 we considered the inference of network structure from observations of information cascades by sampling from $P(G|C)$, where C is our cascade data. Specifically, we considered the inverse problem where the data we observe is some type of dynamic process occurring over the network, rather than information about the network itself. We focused on the inference of social influence networks from the observation of information cascades. Using the well known Independent Cascade model [74] and Metropolis-Hastings, we showed the improved inference of the Bayesian approach over standard optimisation algorithms. We expanded the applicability of this approach in Chapter 6 by inferring directed node neigbourhoods. Not only does this allow the parallelisation of the method to improve computation, it also enables this method to focus on a subset of the individuals that participate in the cascades. We also addressed streaming data by implementing SMC, and show that by extending the model to include external influence we can develop

a data annealing schedule to process data and make estimates in real time. Future work in this area could involve collecting a novel dataset to gain insights into network structure from the dynamics observed. Understanding the differences in network structures that are inferred from different types of cascades is of great interest. For example, networks facilitating real news and fake news, bots and non-bot networks and the difference in networks between users discussing different types of events/topics.

In Chapter 7 we looked at inferring node properties to determine the deployment of RFD in a subset of the internet by sampling from $P(\mathbf{p}|D)$, where D is the measured path data and $\mathbf{p}$ is the vector describing node properties. This novel application of HMC provided a general framework for solving binary network tomography problems. We applied the method to locating route flap damping in the Internet and highlighted that the uncertainty quantification provided by the posterior distribution can provide insights into which nodes are acting consistently and where data should be collected to improve inferences. These methods can be extended to network tomography problems more generally. We showed this on another AS property, route origin validation, and discussed how to include measurement errors in the model to improve inference. We showed that inference of edge properties, rather than node properties, can infer inconsistent dampers more reliably. Further work in this area includes expanding the dataset to include more nodes, applying edge property inference and working with Internet measurement researchers to use the insight gained.

Throughout this thesis we have highlighted the benefits of using Bayesian inference methods in a variety of network science problems. Our methods provide accurate results, have wide applicability and allow for measures of uncertainty. The methods we develop can be used on a wide variety of problems, especially compared to heuristics designed for a single problem or application. We believe there is much scope to continue to apply these methods to gain insightful results and uncertainty estimates in these, and other, network science problems.

Appendix A

Authorship Statements

A.1 Statement of Authorship 1

Title: "Generating Connected Networks"

Publication status: Published

Publication details: Gray, C., Mitchell, L., and Roughan, M. Generating connected random graphs. Journal of Complex Networks 7, 6 (2019), 896-912.

Principal Author

Name of principal author: Caitlin Gray

Contribution to the paper: Wrote the manuscript, implemented the methodology and performed the analysis.

Overall percentage: 70%

Certification: This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.

Signed: ... . Date: 24/7/2020

Secondary Author

Name of principal author: Lewis Mitchell

Contribution to the paper: Project direction & critically reviewed the process and manuscript.

Signed: . Date: 24/7/2020

Secondary Author

Name of principal author: Matthew Roughan

Contribution to the paper: Initiated the concept for the manuscript, project direction & critically reviewed the process and manuscript.

Signed: .... Date: 23rd July 2020

A.2 Statement of Authorship 2

Title: Bayesian Inference of Network Structure From Information Cascades

Publication status: Published

Publication details: Gray, C., Mitchell L., and Roughan M. Bayesian Inference of Network Structure From Information Cascades. *IEEE Transactions on Signal and Information Processing over Networks*, 6 (2020) 371-381.

Principal Author

Name of principal author: Caitlin Gray

Contribution to the paper: Initiated the concept for the manuscript, implemented the methodology, performed the analysis.

Overall percentage: 80%

Certification: This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.

Signed: .... Date: 24/7/2020

Secondary Author

Name of principal author: Lewis Mitchell

Contribution to the paper: Project direction & critically reviewed the process and manuscript.

Signed: Date: 24/7/2020

Secondary Author

Name of principal author: Matthew Roughan

Contribution to the paper: Project direction & critically reviewed the process and manuscript.

Signed: . .... Date: 23rd July 2020

Bibliography

[1] Abbe, E. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research 18*, 1 (2017), 6446–6531.

[2] Artzy-Randrup, Y., and Stone, L. Generating uniformly distributed random networks. *Physical Review E 72* (2005), 056708.

[3] Barabasi, A.-L. The origin of bursts and heavy tails in human dynamics. *Nature 435*, 7039 (2005), 207.

[4] Barabasi, A.-L., and Albert, R. Emergence of scaling in random networks. *Science 286* (1999), 509–511.

[5] Barford, P., Duffield, N., Ron, A., and Sommers, J. Network performance anomaly detection and localization. In *IEEE INFOCOM 2009* (2009), pp. 1377–1385.

[6] Barthélemy, M. Spatial networks. *Physics Reports 499*, 1 (2011), 1 – 101.

[7] Bascompte, J., and Jordano, P. Plant-animal mutualistic networks: The architecture of biodiversity. *Annual Review of Ecology, Evolution, and Systematics 38*, 1 (2007), 567–593.

[8] Blitzstein, J., and Diaconis, P. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Mathematics 6*, 4 (2011), 489–522.

[9] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D. Complex networks: Structure and dynamics. *Physics Reports 424* (2006), 175–308.

[10] Bollobás, B., Janson, S., and Riordan, O. The phase transition in inhomogeneous random graphs. *Random Struct. Algorithms 31*, 1 (2007), 3–122.

[11] Bolstad, W. M., and Curran, J. M. *Introduction to Bayesian statistics.* John Wiley & Sons, 2016.

[12] Braunstein, A., Ingrosso, A., and Muntoni, A. P. Network reconstruction from infection cascades. *Journal of the Royal Society Interface 16*, 151 (2019), 20180844.

[13] Bringmann, K., Keusch, R., and Lengler, J. Geometric inhomogeneous random graphs. *Theoretical Computer Science 760* (2019), 35 – 54.

[14] Britton, T., and O'Neil, P. Bayesian inference for stochastic epidemics in populations with random social structure. *Scandinavian Journal of Statistics 29* (12 2002), 375 – 390.

[15] Broder, A. Z. How hard is it to marry at random? (on the approximation of the permanent). In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1986), STOC '86, ACM, pp. 50–58.

[16] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. *Handbook of Markov chain Monte Carlo.* Chapman and Hall/CRC, New York, 2011.

[17] Butcher, J. C. *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2003.

[18] Butts, C. T. Network inference, error, and informant (in)accuracy: a Bayesian approach. *Social Networks 25*, 2 (2003), 103–140.

[19] Butts, C. T. A perfect sampling method for exponential family random graph models. *The Journal of Mathematical Sociology 42*, 1 (2018), 17–36.

[20] Cáceres, R., Duffield, N., Horowitz, J., and Towsley, D. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions in Information Theory 45*, 7 (1999), 2462–2480.

[21] Caesar, M., Subramanian, L., and Katz, R. H. Towards localizing root causes of BGP dynamics. Tech. rep., EECS Department, University of California, Berkeley, 2003.

[22] Caimo, A., and Friel, N. Bayesian inference for exponential random graph models. *Social Networks 33*, 1 (2011), 41 – 55.

[23] Castro, R., Coates, M., Liang, G., Nowak, R., and Yu, B. Network tomography: Recent developments. *Statistical Science 19*, 3 (2004), 499–517.

[24] Chang, D.-F., Govindan, R., and Heidemann, J. Locating BGP Missing Routes Using Multiple Perspectives. In *Proceedings of the ACM*

SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality (NetT) (New York, NY, USA, 2004), ACM, pp. 301–306.

[25] Chen, L., Crawford, F. W., and Karbasi, A. Submodular variational inference for network reconstruction, 2016. arXiv preprint: 1603.08616.

[26] Cho, S., Nithyanand, R., Razaghpanah, A., and Gill, P. A Churn for the Better: Localizing Censorship using Network-level Path Churn and Network Tomography. In *Proceedings of ACM CoNext* (New York, NY, USA, December 2017), ACM, pp. 81–87.

[27] Chopin, N. A sequential particle filter method for static models. *Biometrika 89*, 3 (2002), 539–552.

[28] Chung, F., and Lu, L. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences 99*, 25 (2002), 15879–15882.

[29] Clarke, D. The Design Principles of the DARPA Internet Protocols. *Computer Communication Review 25*, 1 (January 1995).

[30] Clauset, A., Moore, C., and Newman, M. Hierarchical structure and the prediction of missing links in networks. *Nature 453*, 7191 (2008), 98–101.

[31] Coates, M., and Nowak, R. Network loss inference using unicast end-to-end measurements. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management* (2000), pp. 28–1–28–9.

[32] Cooper, C., Dyer, M., and Greenhill, C. Sampling regular graphs and a peer-to-peer network. *Combinatorics, Probability and Computing 16*, 4 (2007), 557–593.

[33] Cooper, C., Dyer, M., Greenhill, C., and Handley, A. The flip Markov chain for connected regular graphs. *Discrete Applied Mathematics 254* (2019), 56–79.

[34] Crawford, F. W. Hidden network reconstruction from information diffusion. In *2015 18th International Conference on Information Fusion (Fusion)* (2015), IEEE, pp. 180–185.

[35] Crucitti, P., Latora, V., and Marchiori, M. Model for cascading failures in complex networks. *Physical Review E 69* (2004), 045104.

[36] Crucitti, P., Latora, V., and Porta, S. Centrality measures in spatial networks of urban streets. *Physical Review E 73* (2006), 036125.

[37] Davis, J., and Goadrich, M. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning* (2006), pp. 233–240.

[38] Del Moral, P., Doucet, A., and Jasra, A. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68*, 3 (2006), 411–436.

[39] Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. Hybrid Monte Carlo. *Physics Letters B 195*, 2 (1987), 216 – 222.

[40] Duffield, N., Horowitz, J., Presti, F. L., and Towsley, D. Multicast topology inference from measured end-to-end loss. *IEEE Transactions in Information Theory 48*, 1 (2002), 26–45.

[41] Duffield, N., Presti, F. L., Paxson, V., and Towsley, D. Inferring link loss using striped unicast probes. In *Proceedings of IEEE Infocom* (2001), pp. 22–26.

[42] Dutta, R., Mira, A., and Onnela, J.-P. Bayesian inference of spreading processes on networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474*, 2215 (2018), 20180129.

[43] Embar, V. R., Pasumarthi, R. K., and Bhattacharya, I. A Bayesian framework for estimating properties of network diffusions. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2014), KDD '14, ACM, pp. 1216–1225.

[44] Eppstein, D., Galil, Z., Italiano, G. F., and Nissenzweig, A. Sparsification — A technique for speeding up dynamic graph algorithms. *J. ACM 44*, 5 (September 1997), 669–696.

[45] Erdös, P., and Rényi, A. On random graphs, I. *Publicationes Mathematicae (Debrecen) 6* (1959), 290–297.

[46] Farine, D. R., and Strandburg-Peshkin, A. Estimating uncertainty and reliability of social network data using Bayesian inference. *Royal Society Open Science 2*, 9 (2015), 150367.

[47] Feder, T., Guetz, A., Mihail, M., and Saberi, A. A local switch Markov chain on given degree graphs with application in connectivity of peer-to-peer networks. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science* (2006).

[48] Fosdick, B., Larremore, D., Nishimura, J., and Ugander, J. Configuring random graph models with fixed degree sequences. *SIAM Review 60*, 2 (2018), 315–355.

[49] Ghalebi, E., Mirzasoleiman, B., Grosu, R., and Leskovec, J. Dynamic network model from partial observations. In *Advances in Neural Information Processing Systems* (2018), pp. 9862–9872.

[50] Ghosh, B. Random distance within a rectangle and between two rectangles. *Bulletin of the Calcutta Mathematical Society 43*, 1 (1951), 17–24.

[51] Gilad, Y., Cohen, A., Herzberg, A., Schapira, M., and Shulman, H. Are we there yet? On RPKI's deployment and security. In *Proceedings of the Network and Distributed System Security Symposium* (2017).

[52] Gilbert, E. Random graphs. *The Annals of Mathematical Statistics 30* (1959), 1141–1144.

[53] Gkantsidis, C., Mihail, M., and Zegura, E. The Markov chain simulation method for generating connected power law random graphs. In *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments* (2003).

[54] Gomez-Rodriguez, M., Balduzzi, D., and Schölkopf, B. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the 28th International Conference on Machine Learning* (2011).

[55] Gomez-Rodriguez, M., Leskovec, J., and Krause, A. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD) 5*, 4 (2012), 1–37.

[56] Granovetter, M. Threshold models of collective behavior. *American Journal of Sociology 83*, 6 (1978), 1420–1443.

[57] Gray, C., Mitchell, L., and Roughan, M. Super-blockers and the effect of network structure on information cascades. In *Companion Proceedings of the The Web Conference 2018* (2018), pp. 1435–1441.

[58] Gray, C., Mitchell, L., and Roughan, M. Generating connected random graphs. *Journal of Complex Networks 7*, 6 (2019), 896–912.

[59] Gray, C., Mosig, C., Bush, R., Pelsser, C., Roughan, M., Schmidt, T. C., and Wahlisch, M. BGP Beacons, Network Tomography, and Bayesian Computation to locate route flap damping. In *Proceedings of the ACM Internet Measurement Conference* (2020), pp. 492–505.

[60] Green, P. J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*, 4 (1995), 711–732.

[61] Guo, A., and Shakarian, P. A comparison of methods for cascade prediction. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining* (2016), IEEE/ACM, pp. 591–598.

[62] Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference* (Pasadena, CA USA, 2008), G. Varoquaux, T. Vaught, and J. Millman, Eds., pp. 11 – 15.

[63] Hammersley, J. *Monte Carlo methods*. Springer Science & Business Media, 2013.

[64] Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*, 1 (1970), 97–109.

[65] Hawkes, A. G. Spectra of some self-exciting and mutually exciting point processes. *Biometrika 58*, 1 (1971), 83–90.

[66] He, X., and Liu, Y. Not enough data?: Joint inferring multiple diffusion networks via network generation priors. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining* (2017), ACM, pp. 465–474.

[67] Hodas, N. O., and Lerman, K. The simple rules of social contagion. *Scientific Reports 4* (2014), 4343.

[68] Holme, P., and Saramäki, J. Temporal Networks. *Physics Reports 519* (2012), 97–125.

[69] Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., and Morris, M. ERGM: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of Statistical Software 24*, 3 (2008), nihpa54860.

[70] Jerrum, M., and Sinclair, A. Approximating the permanent. *SIAM Journal on Computing 18*, 6 (1989), 1149–1178.

[71] Jerrum, M., and Sinclair, A. Fast uniform generation of regular graphs. *Theoretical Computer Science 73*, 1 (1990), 91 – 100.

[72] Jerrum, M., Sinclair, A., and Vigoda, E. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM 51*, 4 (2004), 671–697.

[73] Karimi, F., and Holme, P. Threshold model of cascades in empirical temporal networks. *Physica A 392*, 16 (2013), 3476–3483.

[74] Kempe, D., Kleinberg, J., and Tardos, E. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2003), pp. 137–146.

[75] King, A. A., Nguyen, D., and Ionides, E. L. Statistical inference for partially observed markov processes via the r package pomp. *Journal of Statistical Software 69* (2016).

[76] Koskinen, J. H., Robins, G. L., Wang, P., and Pattison, P. E. Bayesian analysis for partially observed network data, missing ties, attributes and actors. *Social Networks 35*, 4 (2013), 514–527.

[77] Kossinets, G. Effects of missing data in social networks. *Social networks 28*, 3 (2006), 247–268.

[78] Lang, J. C., De Sterck, H., Kaiser, J. L., and Miller, J. C. Analytic models for SIR disease spread on random spatial networks. *Journal of Complex Networks* (2018), cny004.

[79] LaRock, T., Nanumyan, V., Scholtes, I., Casiraghi, G., and Eliassi-Rad, T. Detecting path anomalies in time series data on networks. *arXiv. org*, 1905.10580 (2019).

[80] Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research 11* (2010), 985–1042.

[81] Leskovec, J., Kleinberg, J., and Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 2.

[82] Leskovec, J., and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[83] Leslie, P. H., Chitty, D., and Chitty, H. The estimation of population parameters from data obtained by means of the capture-recapture method: III. an example of the practical applications of the method. *Biometrika 40*, 1/2 (1953), 137–169.

[84] Linderman, S., and Adams, R. Discovering latent network structure in point process data. In *International Conference on Machine Learning* (2014), pp. 1413–1421.

[85] Liu, J. S., Liang, F., and Wong, W. H. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association 95*, 449 (2000), 121–134.

[86] Liu, L., Tang, J., Han, J., and Yang, S. Learning influence from heterogeneous social networks. In *Data Mining and Knowledge Discovery* (2012), vol. 25, pp. 511–544.

[87] Lusher, D., Koskinen, J., and Robins, G. *Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 2012.

[88] Malmgren, R. D., Stouffer, D. B., Motter, A. E., and Amaral, L. A. A Poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences 105*, 47 (2008), 18153–18158.

[89] Mao, Z. M., Govindan, R., Varghese, G., and Katz, R. H. Route Flap Damping exacerbates internet routing convergence. *SIGCOMM Computer Communication Review 32*, 4 (2002), 221–233.

[90] Maslov, S., and Sneppen, K. Specificity and stability in topology of protein networks. *Science 296*, 5569 (2002), 910–913.

[91] Mathews, P., Gray, C., Mitchell, L., T. Nguyen, G., and Bean, N. SMERC: Social media event response clustering using textual and temporal information. In *2018 IEEE International Conference on Big Data (Big Data)* (12 2018), pp. 3695–3700.

[92] Metropolis, N., and Ulam, S. The Monte Carlo method. *Journal of the American Statistical Association 44*, 247 (1949), 335–341.

[93] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and Austein, R. BGP Prefix Origin Validation. RFC 6811, IETF, January 2013.

[94] Moreno, Y., Pastor-Satorras, R., and Vespignani, A. Epidemic outbreaks in complex heterogeneous network. *The European Physical Journal B 26* (2002), 521–529.

[95] Morris, M., Handcock, M., and Hunter, D. Specification of exponential-family random graph models: Terms and computational aspects. *Journal of Statistical Software, Articles 24*, 4 (2008), 1–24.

[96] Morris, S. Contagion. *Review of Economic Studies 67* (2000), 57–78.

[97] Neal, R. M. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo* (2011), 113.

[98] Netrapalli, P., and Sanghavi, S. Learning the graph of epidemic cascades. In *ACM SIGMETRICS Performance Evaluation Review* (2012), vol. 40, pp. 211–222.

[99] Newman, M. Random graphs as models of networks. *Handbook of graphs and networks 1* (2003), 35–68.

[100] Newman, M. *Networks: An Introduction*. Oxford University Press, 2010.

[101] Newman, M. Estimating network structure from unreliable measurements. *Physical Review E 98* (2018), 062321.

[102] Nishimura, J. The connectivity of graphs of graphs with self-loops and a given degree sequence. *Journal of Complex Networks 6*, 6 (2018), 927–947.

[103] Padmanabhan, V. N., Qiu, L., and Wang, H. J. Passive network tomography using Bayesian inference. In *Proceedings of ACM Internet Measurement Workshop* (New York, NY, USA, 2002), ACM, pp. 93–94.

[104] Pastor-Satorras, R., Castellano, C., Van Mieghem, P., and Vespignani, A. Epidemic processes in complex networks. *Reviews of Modern Physics 87* (2015), 925–979.

[105] Payne, J. L., Dodds, P. S., and Eppstein, M. J. Information cascades on degree-correlated random networks. *Physical Review E 80* (2009), 026125.

[106] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[107] Peixoto, T. Reconstructing networks with unknown and heterogeneous errors. *Physical Review X 8* (2018), 041011.

[108] Peixoto, T. Network reconstruction and community detection from dynamics. *Physical Review Letters 123* (2019), 128301.

[109] Pelsser, C., Maennel, O., Mohapatra, P., Bush, R., and Patel, K. Route flap damping made usable. In *Passive and Active Measurement* (Berlin, Heidelberg, 2011), N. Spring and G. F. Riley, Eds., Springer Berlin Heidelberg, pp. 143–152.

[110] Piedrahita, P., Borge-Holthoefer, J., Moreno, Y., and González-Bailón, S. The contagion effects of repeated activation in social networks, 2017. arXiv preprint: 1703.04017.

[111] Pouget-Abadie, J., and Horel, T. Inferring graphs from cascades: A sparse recovery framework. In *Proceedings of the 24th International Conference on World Wide Web* (New York, NY, USA, 2015), WWW '15 Companion, ACM, pp. 625–626.

[112] Raftery, A. E., and Lewis, S. M. Practical Markov chain Monte Carlo: One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. *Statistical Science 7*, 4 (1992), 493–497.

[113] Ramezani, M., Rabiee, H., Tahani, M., and Rajabi, A. Dani: A fast diffusion aware network inference algorithm. *ACM Transactions on Knowledge Discovery from Data (TKDD)* (06 2017).

[114] Rapoport, A., and Horrath, W. A study of a large sociogram. *Behavioral Science 6*, 4 (1961), 279–291.

[115] Rechner, S., Strowick, L., and Müller-Hannemann, M. Uniform sampling of bipartite graphs with degrees in prescribed intervals. *Journal of Complex Networks 6*, 6 (2017), 833–858.

[116] Robert, C., and Casella, G. A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data. *Statistical Science* (2011), 102–115.

[117] Robert, C. P., and Casella, G. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[118] Rogers, E. M. *Diffusion of innovations*. Simon and Schuster, 2010.

[119] Roughan, M., Tuke, J., and Parsonage, E. Estimating the parameters of the Waxman random graph, 2015. arXiv preprint: 1506.07974.

[120] Roughan, M., Willinger, W., Maennel, O., Perouli, D., and Bush, R. 10 lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE Journal on Selected Areas in Communications 29*, 9 (2011), 1810–1821.

[121] Santos-Fernández, E., Mengersen, K., and Wu, P. *Bayesian Methods in Sport Statistics*. Wiley, 02 2019, pp. 1–8.

[122] Scellato, S., Noulas, A., Lambiotte, R., and Mascolo, C. Socio-spatial properties of online location-based social networks. In *Fifth international AAAI conference on weblogs and social media* (2011).

[123] Schelling, T. Models of segregation. *The American Economic Review 59*, 2 (1969), 488–493.

[124] Schelling, T. Dynamic models of segregation. *The Journal of Mathematical Sociology 1*, 2 (1971), 143–186.

[125] Shaghaghian, S., and Coates, M. Bayesian inference of diffusion networks with unknown infection times. *2016 IEEE Statistical Signal Processing Workshop (SSP)* (2016), 1–5.

[126] Shaghaghian, S., and Coates, M. J. Online Bayesian inference of diffusion networks. *IEEE Transactions on Signal and Information Processing over Networks 3* (2017), 500–512.

[127] Sisson, S., Fan, Y., and Beaumont, M. *Handbook of Approximate Bayesian Computation*. Chapman and Hall/CRC, New York, 2019.

[128] Sprague, D., and House, T. Evidence for complex contagion models of social contagion. *PLoS ONE 12*, 7 (2017).

[129] Stan Development Team. PyStan: the Python interface to Stan. http://mc-stan.org, Version 2.17.1.0 2018.

[130] Stewart III, J. W. *BGP4: inter-domain routing in the Internet*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[131] Tabourier, L., Roth, C., and Cointet, J.-P. Generating constrained random graphs using multiple edge switches. *Journal of Experimental Algorithmics 16* (2011), 1–15.

[132] Tan, Q., Liu, Y., and Liu, J. Motif-aware diffusion network inference. In *Advances in Knowledge Discovery and Data Mining* (Cham, 2018), D. Phung, V. S. Tseng, G. I. Webb, B. Ho, M. Ganji, and L. Rashidi, Eds., Springer International Publishing, pp. 638–650.

[133] Testart, C., Richter, P., King, A., Dainotti, A., and Clark, D. To filter or not to filter: Measuring the benefits of registering in the RPKI today. In *Proceedings of Passive and Active Measurement Conference* (Berlin Heidelberg, 2020), vol. 12048 of *LNCS*, Springer, pp. 71–87.

[134] Tutte, W. T. The dissection of equilateral triangles into equilateral triangles. *Mathematical Proceedings of the Cambridge Philosophical Society 44*, 4 (1948), 463–482.

[135] Vaisman, R., Roughan, M., and Kroese, D. P. The multilevel splitting algorithm for graph colouring with application to the Potts model. *Philosophical Magazine 97*, 19 (2017), 1646–1673.

[136] Vajdi, A., and Scoglio, C. Identification of missing links using susceptible-infected-susceptible spreading traces. *IEEE Transactions on Network Science and Engineering* (2018).

[137] Valente, T. W. Social network thresholds in the diffusion of innovations. *Social networks 18*, 1 (1996), 69–89.

[138] Vardi, Y. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association 91*, 433 (1996), 365–377.

[139] Viger, F., and Latapy, M. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In *Computing and Combinatorics* (Berlin, Heidelberg, 2005), L. Wang, Ed., Springer Berlin Heidelberg, pp. 440–449.

[140] Wang, S., Hu, X., Yu, P. S., and Li, Z. MMRate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge discovery and data mining* (2014), ACM, pp. 1246–1255.

[141] Watts, D. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences 99*, 9 (2002), 5766–5771.

[142] Watts, D., and Dodds, P. Influentials, networks and public opinion formation. *Journal of Consumer Research 34*, 4 (December 2007), 441–458.

[143] Waxman, B. M. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications 6*, 9 (Dec 1988), 1617–1622.

[144] Willinger, W., and Roughan, M. Internet topology research redux. *ACM SIGCOMM eBook: Recent Advances in Networking* (2013).

[145] Wright, G. R., and Stevens, W. R. *TCP/IP Illustrated, Volume 2.* Addison-Wesley Publishing Company, 1995, ch. 31.

[146] Ying, X., and Wu, X. Graph generation with prescribed feature constraints. In *Proceedings of the 2009 SIAM International Conference on Data Mining* (2009), pp. 966–977.

[147] Zhang, J., Litvinova, M., Wang, W., Wang, Y., Deng, X., Chen, X., Li, M., Zheng, W., Yi, L., Chen, X., et al. Evolving epidemiology and transmission dynamics of coronavirus disease 2019 outside Hubei province, China: a descriptive and modelling study. *The Lancet Infectious Diseases 20*, 7 (2020), 793–802.

[148] Zhao, Q., Erdogdu, M., He, Y., Rajaraman, A., and Leskovec, J. SEISEMIC: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining.* (2015), ACM, SIGKDD.

[149] Zhao, Z., Zhao, J., Sano, Y., Levy, O., Takayasu, H., Takayasu, M., Li, D., Wu, J., and Havlin, S. Fake news propagates differently from real news even at early stages of spreading. *EPJ Data Science 9*, 7 (2020).