The Dissertation Committee for Jessica Hélène Hoffmann
certifies that this is the approved version of the following dissertation:

Epidemics on Graphs under Uncertainty

Committee:

Constantine Caramanis, Supervisor

Alexandros Dimakis

Adam Klivans

Sanjay Shakkottai

Epidemics on Graphs under Uncertainty

by

Jessica Hélène Hoffmann

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2020

Dedicated to Işıl Dillig, and to the women in science who left due to lack of support.

Acknowledgments

First and foremost, I would like to thank my advisor Constantine Caramanis for teaching, mentoring, guiding, and showing me by example what it means to be a researcher. There is not one aspect of my research life which hasn't been influenced by his style: my general technical skills, a curiosity for the unexpected, a taste for challenge, and an unending quest for meaningful and impactful work. He demonstrated to me what it means to make brave choices, especially within the context of the Black Lives Matter movement. I will never forget the personal support he provided me at difficult times, be it after the 2016 election, the coronavirus crisis, or my job search. He has shown me how to have a successful work relationship without neglecting the human side of it. For this, and for everything, I am forever grateful.

I would also like to thank the other members of my committee, who shaped my experience in WNCG and in the CS department. Attending Alex's group meeting was invaluable at the beginning of my PhD, and helped me understand how to come up with a research problem. His energy and cheerfulness were always inspiring. I spent countless hours working in Adam's lab, where the entire group was vibrant with ideas and research energy. It was always a pleasure when Adam would come by, and maybe one day I will win one of his challenges. Finally, Sanjay's unending energy and passion has always been a

source of motivation for me and my coworkers.

Research would not be half as interesting without the collaborations I've made along the way. I would like to thank all my co-authors: Constantine, Ashish, Surbhi, Soumya, Moein, Ahmad, Reza, Mahdi, Adrian, and (hopefully) Matt, it has been exciting and a beautiful learning experience to work with all of you. I also want to thank my lab, past and present, for their ideas and support: Tianyang, for all the up-to-date articles; Kiyeon, for in-depth discussions about American slang; Eirini, for hours of work and fun; Liu, for showing me what persistence and determination is; Ashish, for unending support and good collaborations; Jiacheng, for his motorbike knowledge and incredible research productivity; Jeong Yeol, for his quiet but brilliant and incisive mind; Orestis and Isidoros for sharing my love of combinatorics; and finally Matthew and Liam for giving me an occasion to act as a senior student. I am particularly grateful to Manuel Gomez-Rodriguez, for the incredibly exciting research discussions when I was invited to visit his lab. I would also like to thank everyone who worked with me outside of my lab, in WNCG and beyond: Taylor, Eftychia, Brooke, Sushi, Aravind, Surbhi, Matt, Sepideh and Josh. A big shoutout to Katie, my administrative savior, as well as Karen and Jaymie, and of course Apipol, my favorite bocce partner. Finally, I just want to add one more thank-you to Constantine for fostering such a friendly atmosphere in our group, which led to so many fruitful collaborations.

None of this would have been possible without my friends, my husband, my sister, and the rest of my family, who were by my side during the ups and

downs of this thrilling adventure. This is not the medium to tell you how much you mean to me, but you should hear from me individually soon.

Epidemics on Graphs under Uncertainty

Publication No. _____

Jessica Hélène Hoffmann, Ph.D.
The University of Texas at Austin, 2020

Supervisor: Constantine Caramanis

Epidemic processes can model anything that spreads. As such, they are a useful tool for studying not only human diseases, but also network attacks, spikes in the brain, the propagation of real or fake news, the spread of viral tweets, and other processes. This proposed thesis focuses on epidemics spreading on an underlying graph.

Currently, most state-of-the-art research in this field assumes some form of perfect observation of the epidemic process. This is an unrealistic assumption for many real-life applications, as the recent COVID-19 pandemic tragically demonstrated: data is scarce, delayed, and/or imprecise for human epidemics, and symptoms may appear in a non-deterministic fashion - if they appear at all. We show in this work not only that the algorithms developed previously are not robust to adding noise into the observation, but that some theoretical results cannot be adapted to this setting. In other words, uncertainty fundamentally changes how we must approach epidemics on graphs.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Epidemic models are used widely across biological and social sciences, engineering, and computer science. They have made an important impact on the study of the dynamics of human disease and computer viruses as well as trends, rumors, viral videos, and (most recently) the spread of fake news on social networks. In this work, we focus on epidemics propagating on a graph, as introduced by the seminal paper [65]. In this setting, an infected node can only propagate the infection to its susceptible neighbors, as opposed to the fully mixed models considered in the early literature. This graph-based approach provides a more realistic model in which the spread of the epidemic is determined by the connectivity of the graph, and, accordingly, some nodes may play a larger role than others in the spread of the infection. This work therefore leverages concepts and results from graph theory.

Most state-of-the-art research on epidemic processes on graphs assumes we have perfect information. However, this is often not the case. In the midst of the COVID-19 crisis, it is painfully evident that it is unrealistic to assume we know exactly who is infected at any given time, as people do not constantly report their infection state. Similarly, it is unrealistic to assume we know exactly

when someone became infected: official diagnosis (and hence recording by any tracking entity such as the Centers for Disease Control and Prevention) may come days, weeks, or – in important examples such as HIV – *years* after the actual moment of infection. Moreover, this delay can vary widely from person to person, hence the infector is often diagnosed after the infectee. Similar issues arise with biological networks: we only know the expression of a gene when we perform an experiment, which can happen after a typically arbitrary delay. Finally, even in the realm of online epidemics (e.g. viral tweets), for which perfect information is often taken for granted, there can be uncertainty about what really infected the nodes. Indeed, posts are shared differently based on their topic, which may be unclear (e.g. someone sharing "We won!!!" could indistinguishably be talking about sports or politics). This implies epidemics might not spread on graphs, but on hidden mixtures of graphs.

Accordingly, this thesis aims to answer the following question: *How can we tackle the problems of curing and network inference for epidemics spreading on graphs under various forms of uncertainty in the observation model?* To do so, we draw on results from graph theory, probability, random walks, combinatorics, convex optimization, and information theory to prove our results. In particular, we make use of Wald's inequality, Sanov's theorem, and Fano's lemma.

1.1 Overview of results

1.1.1 Uncertainty about who is infected

In Chapter 3, we consider the problem of curing a graph under a restricted budget when there is uncertainty about which nodes are infected. We show that this uncertainty dramatically changes the landscape: we exhibit graphs which cannot be cured in polynomial time for a given curing budget in the uncertain setting, despite being curable in sublinear time for a fraction of the same budget in the perfect information setting.

1.1.2 Uncertainty about when nodes are infected

In Chapter 4, we consider the inverse problem of graph inference from epidemic cascades, specifically from noisy times of infection. We prove we can reliably learn the edges of graphs from a very weak observation model, and learn the weights for *any* known noise distribution. For trees and bounded-degree graphs, our methods are sample-optimal (up to log factors).

1.1.3 Uncertainty about what is infecting nodes

In Chapter 5, we still consider graph inference from epidemic cascades, but this time we consider the case in which the cascades spread on an (unknown) mixture of two graphs.We show this problem is not always identifiable, and we give necessary and sufficient conditions for the problem to be identifiable with polynomial samples. In the identifiable case, we provide a sample-optimal (up to log factors) algorithm which reconstructs both graphs.

1.2 Published work

The work presented in this thesis is based on joint work with collaborators, as described below.

Chapter 3 and Chapter 4 are based on joint research with Constantine Caramanis and appeared in SIGMETRICS 2018 [40] and 2019 [41] respectively. The latter paper won 2^{nd} *place* in the 2019 INFORMS George E. Nicholson Student Paper Competition.

Chapter 5 features joint work with Soumya Basu, Surbhi Goel, and Constantine Caramanis and appeared at ICML 2020 [39].

Chapter 2

Preliminaries

2.1 Notations

$G = (V, E)$	Graph G, V set of nodes, E set of edges.
N	Number of nodes in the graph
p_{ij}	Weight of edge (i, j), corresponding to the probability that i infects j.
I_t	Number of infected nodes at time t
S_t	Number of susceptible nodes at time t

Table 2.1: Global notations

2.2 Relevant background

2.2.1 Epidemic models

The study of epidemics dates to 1760, when Bernoulli [8] introduced what would later be called compartment models. In this model, we assume that every individual belongs to a state, which could be:

Infected: Infected individuals have the spreading agent, and spread it.

Susceptible: Susceptible individuals can become infected.

Removed: Removed individuals no longer interact with the spreading agent, and in particular cannot become infected.

The crucial assumption of compartment models is that *everyone can infect everyone*, which is why this type of model is sometimes called a fully mixed model. There exist other extensively studied states: for instance, the exposed state, in which individuals already have the spreading agent in them, but are not infectious yet. While this thesis does not consider fully mixed models, we still make use of the same terminology, and we will focus on the susceptible, infected, and removed states.

2.2.1.1 SIR model

Figure 2.1: SIR model.

In the SIR (Susceptible $\rightarrow$ Infected $\rightarrow$ Removed) model, susceptible individuals can become infected. Infected individuals infect other susceptible nodes until they become removed, and stop interacting with the spreading process. The SIR model is the model of choice for many human epidemics, as humans usually develop immunity to a disease (or die, in which case they also are removed). Chickenpox is one of the most well-known examples of SIR epidemics.

The SIR model is also useful for representing online epidemics, and in particular the spread of viral content on social networks. Indeed, people usually interact (e.g. share, like, tag other people) with new content (video,

article, meme, etc.) only once: the first time they see it.

It is possible for a SIR epidemic to never reach the status of an outbreak. Indeed, if it is hard to infect new individuals, but easy to transition to the removed state once infected, a SIR epidemic can die out quickly. If it does not, however, then the outbreak will develop in two phases: an exponential explosion of the number of infected individuals at the start, when every infected node can easily spread the agent to many susceptible nodes, and an exponential decrease, when susceptible individuals become rarer and most infected individuals transition to the removed state without infecting new people. At the end of an outbreak, not everyone will have been infected, and the expected number of people in the removed state can be calculated from the parameters of the epidemic.

2.2.1.2 SI model

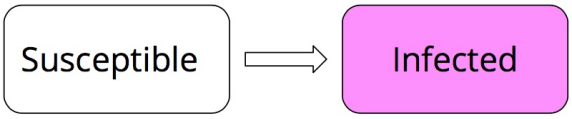


Figure 2.2: SI model.

In the SI (Susceptible → Infected) model, susceptible individuals become infected, and stay infected. As a result, without outside intervention, everyone will be infected at the end of the process.

Noteworthy examples of SI epidemics include the spread of news (e.g. the election of a new president) or a zombie apocalypse.

2.2.1.3 SIS model

Figure 2.3: SIS model.

In the SIS (Susceptible → Infected → Susceptible) model, susceptible individuals become infected, and can then become susceptible again. Contrary to the two examples above, in which the process dies out quickly (when people are all susceptible or removed in the SIR model, or when they are all infected in the SI model), an SIS epidemic can live indefinitely, as new individuals become infected continuously, and previously infected individuals become future potential targets when they revert to the susceptible state. In this case, a constant fraction of the population is continuously infected, and we say the epidemic is *endemic*.

SIS epidemics are particularly useful for modeling the spread of computer viruses. They are also useful models for diseases caused by viruses that mutate (e.g. the cold, the flu, or AIDS).

2.2.2 Epidemics on graphs

2.2.2.1 Modeling choices

In this thesis, we focus on epidemics propagating on a graph. In contrast to the fully mixed models, in which every infected node can infect every

susceptible node, here the spread is restricted by the topology of the graph: infected nodes can only infect their (susceptible) neighbors in the graph. Likelihood of transmission can then be represented by the weights of the edges in the graph.

Epidemics on graphs provide the highest level of granularity for the study of epidemics. As such, a large body of work has focused on this setting, spanning numerous applications: modeling epidemics [13, 17, 36, 55, 81, 88], detecting them [4, 6, 46, 53, 58, 60, 62], detecting communities [69, 82], finding their source [20, 71, 72, 74, 76, 77, 80], obfuscating the source, [26–28], or controlling their spread [22, 23, 30, 49, 67, 79, 84]. We choose to focus on this model for our theoretical results as it guarantees the highest level of generalization for diseases. We briefly mention other commonly adopted models for studying epidemics:

- Epidemiologists need computationally efficient models for their simulation-intensive predictive tasks. This is why they sometimes prefer meta-population [38, 42, 54, 66, 78] or agent-based [3, 68] models when studying real diseases.

- In the case of social network epidemics, the fresher and more popular the content already is, the more likely it will be to become even more popular. This community might therefore prefer to employ self-exciting processes (e.g. Hawkes processes [14, 25, 31, 48, 57, 89]).

2.2.2.2 The independent cascade model

One central definition in the field of epidemics on graphs is the following:

Definition 2.2.1 (Cascade)**.** One instance of a spreading process, from the moment one or multiple nodes are chosen as sources (i.e. first infected nodes) to the moment when there exist no infected nodes, is called a cascade.

When we observe multiple cascades, for instance in Chapters 4 and 5, we assume *cascade are independent*. This is a variant of the model introduced by [34] and further studied by [45].

A cascade can represent one viral article spreading on social networks or one disease spreading in a city. In the independent cascade model, we assume that how previous cascade spread does not affect how future cascades will spread.

2.2.3 Cutwidth and curing budget

We now present a result which will be used in Chapter 3. The problem of curing an epidemic with a limited budget but with perfect observation (i.e., perfect knowledge of the state of nodes at each point in time) has been recently considered in [22, 24]. Their budget is a bound on the *curing effort* they can expend at a given time (as opposed to the total curing effort over time). In this setting, the problem is to optimize the allocation of the curing budget across nodes at every point in time. They characterize the budget required

for fast curing as a function of a combinatorial property of the graph – its CutWidth:

Definition 2.2.2 (CutWidth)**.** Given a graph $G = G(V, E)$, and any subset of the nodes, $S \subseteq V$, the Cut of S is the number of edges crossing from S to S^c. Given any sequence of $|V| + 1$ subsets $S_0, \ldots, S_{|V|}$ such that $S_0 = \emptyset$, $S_{|V|} = V$, and S_k and S_{k+1} differ by the addition of a single node (called a *crusade* in [22, 23]), the cut of the sequence is the largest cut of any of the sets S_k. The CutWidth of a graph is the minimum cut of any sequence satisfying the above properties.

We now present the main results of [22, 24]:

Theorem 2.2.1. *Consider an epidemic spreading on a graph with bounded degree. Suppose the epidemic spreads from infected nodes to their neighbors following an exponential clock of parameter 1, and we can probabilistically cure nodes one by one at speed following an exponential clock of parameter r. Let W be the* CutWidth *of the graph, N be the number of nodes, and $\epsilon > 0$ be a constant. Then:*

- *If $r \geq (1+\epsilon) \cdot \max(W, \log(N))$, there exists a curing strategy which cures the entire graph in sublinear (in N) expected time.*
- *If $r \leq (1-\epsilon) \cdot \max(W, \log(N))$, no curing strategy can guarantee we can cure the graph in less than exponential (in N) expected time.*

Chapter 3

Uncertainty About Who Is Infected

3.1 Introduction

In this chapter[1], we challenge a commonly accepted assumption in epidemic studies, which is that we know whether or not someone is infected. As the COVID-19 pandemic (ongoing as of this writing) has vividly demonstrated, this assumption is far from true in real-life scenarios. Individuals can be infectious before showing symptoms, or even without ever showing symptoms (asymptomatic individuals)[2]. To emphasize the impact of uncertainty, we consider an optimistic observation model: every node reports its status at every time step, but this report is corrupted with probability $(1 - p)$ if the node is infected, and with probability q if the node is susceptible. Note that we obtain information from *every* individual at *every* time step, which is already a strong model of observation. However, we show that introducing even this level of uncertainty radically changes the results proven with perfect information: there exist graphs which cannot be cured in polynomial time for a given curing budget in the uncertain setting, despite being curable in sublinear time

[1]This chapter covers the material prebiously published in *The Cost of Uncertainty in Curing Epidemics*. My main contribution was solving the theoretical challenges.

[2]Worse, they can be showing symptoms but choosing to not report them.

for a fraction of the same budget in the perfect information setting.

3.1.1 Setting

We focus on curing SI models, where infected nodes probabilistically propagate the infection to their non-infected neighbor, while we can choose nodes to attempt to cure, which corresponds to probabilistically transitioning them back to the susceptible state.

At any point in time, the *state* of an SI-type epidemic on a graph is given by the list of nodes on the graph that are infected, and their relative topology (position) in the graph. Having a good estimate of the state is critical, as it determines the dynamics of the spread of the epidemic into the future. As a simple example, we can ask what the spreading rate is on an N-node line graph of an infection with $N/2$ infected nodes. If those nodes are contiguous, then it will take $O(N)$ time for the epidemic to spread to the entire graph. If every other node is infected, it will take $O(1)$ time.

If we have access to the status of each node (infected or not), then we know the state exactly. Much work has focused on the state estimation problem, in the setting where only noisy information is available. Indeed, work in [59, 61, 62], [4–6], and elsewhere, considers a setting where only noisy observations of the status of each node are possible, and even answering whether there is an epidemic or not is a challenge. Those and related works, as we discuss in more detail below, focus on the problems related to epidemic state estimation, and do not consider the control problem of curing the epidemic.

On the other side, the problem of curing an epidemic with a limited budget, *but with perfect observation* (i.e., perfect knowledge of the state at each point in time), has been recently considered in [22, 24]. Their budget, as we explain more precisely further below, is essentially a bound on the *curing effort* they can expend at a given time (as opposed to total curing effort over time). In this setting, the problem is to optimize the allocation of the curing budget across nodes at every point in time. They characterize the budget required for fast curing, as a function of a combinatorial property of the graph – its CutWidth (we define this below).

However, as metioned before, knowing the state of every node in the graph is often impraticable, or even impossible for some disease (e.g. asymptomatic individuals for COVID-19). The problem of curing an epidemic with a limited budget and partial observation of the state of the epidemic (i.e., which nodes are infected and which are not) introduces a fundamentally new element to the problem. Indeed, this interaction represents a fundamental tension: our estimate of the state of a node improves the longer we observe it, and so the longer we wait to cure a node, the less likely we are to waste precious curing resources on non-infected nodes. On the other hand, the longer an infected node remains untreated, the more the epidemic spreads. To the best of our knowledge, no work has successfully attacked the problem of curing an epidemic with a limited budget and partial observation of the state of the epidemic (i.e., which nodes are infected and which are not). Our work considers precisely this problem, and therefore, broadly speaking, is about the

interaction of – specifically, simultaneous – learning and control.

By considering learning the state and controlling the epidemic simultaneously, we prove a lower bound that shows (see Section 3.5 for precise result) that partial information can have a dramatic impact on the resources (either time or budget) required to cure an infection: even with slightly imperfect/incomplete information, the time to cure a particular graph may increase exponentially, unless the budget is also significantly increased. Concretely, we show that if instead of receiving the state of each node at each point in time, we receive a slightly noisy (e.g., only 99% accurate) guess of the state, then there is no constant factor of the CutWidth which is sufficient for *any algorithm* to cure the epidemic in linear (expected) time.

3.1.2 Related Work and Background

Detecting an epidemic, as well as its location, under noisy data, has been well-studied in [4], in the context of detecting a multidimensional anomalous cluster, with time playing the same role as any other dimension. Graph-specific epidemic detection has been further studied by [75], with constraints based on the cut of this anomalous cluster. [62] study the detection of epidemic-specific clusters by detecting the shapes which arise specifically when there is an epidemic. The focus in those works has been to understand the limit of information required in order to detect the epidemic. More generally, inverse problems have also been of interest, especially source detection [71, 73, 74, 76, 80] or obfuscation [26, 29].

In our work, we adopt a much stronger observation model than in the papers listed above; our negative result establishes, however, that controlling the epidemic is impossible with weaker information than the threshold we characterize.

In [22, 23], the authors tackle the problem of curing graphs with perfect knowledge of the state of each node, constrained by a budget which corresponds to the speed at which the nodes are cured. Their results show that there exists a threshold phenomenon: for any given graph, if the curing budget is lower than a combinatorial quantity of the graph called the CutWidth, the curing time is exponential; if it is higher, they exhibit a strategy to cure any graph in sublinear time. The CutWidth captures a key bottleneck in curing, and is important in our work as well. Therefore it is useful to define this precisely now.

Definition 3.1.1. Given a graph $G = G(V, E)$, and any subset of the nodes, $S \subseteq V$, the Cut of S is the number of edges crossing from S to S^c. Given any sequence of $|V|+1$ subsets $S_0, \ldots, S_{|V|}$ such that $S_0 = \emptyset$, $S_{|V|} = V$, and S_k and S_{k+1} differ by the addition of a single node (called a *crusade* in [22, 23]), the cut of the sequence is the largest cut of any of the sets S_k. The CutWidth of a graph is the minimum cut of any sequence satisfying the above properties.

Intuitively, the CutWidth of a graph is the largest cut one would be *forced* to encounter when curing a graph. The cut of a subset is critical, because for an infected set of nodes S, its cut is the number of non-infected nodes

adjacent to infected nodes, and hence is the instantaneous rate of infection of the epidemic at that moment (in that configuration). For an illustration, consider again an N-node line graph. Its CutWidth is equal to one, since when curing the graph from one end to the other, we have only one single non-infected node adjacent to an infected node at any time. Note that this is the best case, because if we were to start curing nodes in the middle of the infection, the cut between the infected nodes and the non-infected nodes could be made as large as $O(N)$.

Their strategy is based on two main ideas. The nodes are cured following an ordering which keeps the cut between the infected set and the non-infected set as low as possible. Then, as soon as there is a new infection, the strategy switches to damage control, and focuses on returning to the ordering previously mentioned.

Our result hinges on the fact that the damage-control part of the strategy is exactly the part which is hard to accomplish with partial information. If the number of k-hop neighbors of a node grows exponentially, as is the case for the binary tree, detecting where the infection can have spread becomes a difficult task. Moreover, if we can detect such an escape path, but the infection has spread to a high number of nodes by the time we have enough information to try to prevent it, detection was useless. It is the tension between waiting less time and wasting budget on false alerts, or waiting too long and being unable to prevent the spread, which makes the problem of curing with partial information challenging.

3.2 Model and main contributions

The key elements that define our model are the dynamics of the spreading process and the controlled curing process, and then the stochastic process that defines the degradation from perfect information. We describe these in detail, in this order. We then provide a few basic definitions that appear repeatedly throughout the chapter, and then finally outline the main contributions of this work.

3.2.1 The SI + curing model

In a standard SI (susceptible $\rightarrow$ infected) model, an epidemic spreads along edges from infected nodes to their neighbors according to an exponential spreading model: when a node becomes infected, it infects each uninfected neighbor according to an exponential random variable. SIS models are SI models where infected nodes also transition to susceptible, again at an exponential rate. Here, we consider the setting where the rate at which nodes transition from infected to susceptible is under our control, subject to a budget. How to optimally use this budget is the main question at hand. We prefer to call this a *controlled SI* process rather than a SIS process, because we are interested in the regime where our total curing budget is $o(N)$, where N is the number of nodes. A SIS process typically has transitions from susceptible to infected of the same order as the infection rates; in our setting, this would correspond to a budget of at least $O(N)$. We note that much work has considered this setting, and has characterized the absorption time (into the "all cured"' state)

N	Number of nodes in the graph
I_t	Number of infected nodes at time t
τ	Size of a time step
r_i^t	Budget spent on node i at time t
$r = \sum_{i=1}^{N} r_i^t$	Total budget for each time step
$\mu = 1 - e^{-\tau}$	Probability of an infection along an edge between a susceptible and an infected node
$\delta_i^t = 1 - e^{-r_i^t \cdot \tau}$	Probability that infected node i gets cured at time t
$\delta = 1 - e^{-r \cdot \tau}$	Maximum probability of being cured for a node
p	$\mathbb{P}$(node i raises a flag at time t | node i is infected)
q	$\mathbb{P}$(node i raises a flag at time t | node i is susceptible)

Table 3.1: Summary of noations for this chapter

as a function of the topology of the network [33].

In the sequel, we consider a discrete, Bernoulli approximation to these exponential rate models, by considering the dynamics evolving with discrete time steps τ; *we then take the time step τ to zero*, hence recovering the continuous time dynamics. In particular, this model is a discretization of the exponential model of [24]. As $\tau \to 0$, the models become equivalent. This discretization and the subsequent limit as $\tau \to 0$ facilitate our quantification of uncertainty, i.e., how much information we receive about the state of each node, in a given time interval. This is defined precisely below.

The dynamics of this controlled stochastic process evolve as follows. At each time t, for all N nodes of the graph, the decision-maker assigns a budget r_i^t, subject to the constraints $\sum_{i=1}^{N} r_i^t = r$. During a time step of length τ, each node i is cured with probability $\delta_i^t = 1 - e^{-r_i^t \tau}$ if it was infected, and nothing happens otherwise – the budget is wasted. Then, for every edge between

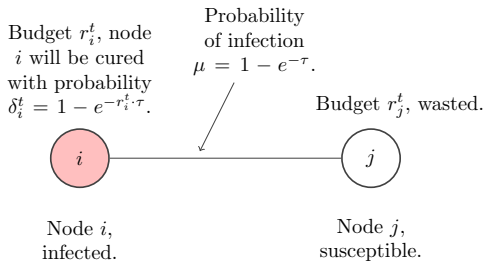


Figure 3.1: Visual representation of the different parameters.

an infected and a susceptible node, an infection occurs with probability $\mu = 1 - e^{-\tau}$. The number of infected nodes at time t is given by I_t. In particular, since the graph is completely infected at the beginning, we have $I_0 = N$.

We now give a few definitions related to the above quantities, that we use throughout this chapter.

Definition 3.2.1. We call **curing process** the stochastic process of cures and infections according to the model described in section 3.2.1. This process has a deterministic part (how much of the budget is assigned to which nodes at each time step), and a stochastic part (curing and infection follow geometric laws).

Definition 3.2.2. We call a **strategy** the set of budgets assigned for each node at each time: $\{r_i^t,\, i \in [N],\, t = k \cdot \tau,\, k \in \mathbb{N}\}$. We note that in the Partial Information setting that we introduce below, the actions taken at time t_1 may depend on the information accumulated until time $t_1 - \tau$.

In the rest of the chapter, we refer to the set of infected nodes (resp. susceptible nodes) as the **infected set** (resp. **susceptible set**). We may also

refer to the cut between the infected set and the susceptible set as the **cut**. When we use the word **distance** between two nodes in a graph, we refer to the number of nodes in the shortest path between these two nodes. The distance between a node and a set is the shortest distance between this node and any node of the set.

3.2.2 Partial Information/Blind Curing

In the Complete Information setting, we assume that the status (infected or susceptible) of each node is known at each point in time. In what we call the Blind Curing model, we never have any information about the status of each node. The Blind Curing model is a technical tool we use en route to the final result. We introduce a Partial Information model that interpolates between these two extremes, and indeed is our main object of interest. Our model of partial information provides a stark tradeoff for the decision-maker: allocate resources to nodes whose status is very uncertain, and thus significantly raise the probability of wasting curing resources, or wait to collect more information and hence more certainty about the status of a node, running the risk that an infected node was allowed to infect neighbors unfettered.

Our motivation for our partial information model comes from zero-day *behavioral* malware detectors, often called *Local Detectors* [9, 44], where anti-malware software raises alerts of "suspicious behavior" that are then related to a central authority. We refer to these alerts as "flags." Thus, in the Complete Information model, an infected node would raise a flag at each instant

with probability 1, and an uninfected node would never raise a flag. In the **Partial Information** model, at each time step, each node, independently of all others, raises a flag with some probability. The probability of getting a flag is p if the node is infected, q if the node is susceptible, with $p > q$. By aggregating the information about a node over multiple time steps, we can use basic concentration inequalities to deduce its state, and thus more observation time corresponds to higher certainty about a node's state.

As noted above, $p = 1, q = 0$ recovers the **Complete Information** setting, and $p = q$ the **Blind Curing** setting.

In order to recover the continuous time dynamics, we let $\tau \to 0$. The key quantity that measures the amount of information per fixed unit time, is given by the rate function from Sanov's theorem, normalized by the time step: $\frac{\mathcal{D}(p||q)}{\tau}$, where $\mathcal{D}(p||q)$ is the Kullback-Leibler distance between p and q [15]. To understand this intuitively, this says that when $\frac{\mathcal{D}(p||q)}{\tau}$ is a constant, observing a node for a fixed period of time corresponds to administering a test with a nonzero false positive and false negative probability. That is, we can know the state of a node with constant probability of error by observing this node over a constant amount of time, which is what one expects from a real-world source of information. Note that as $\tau \to 0$, if $p - q$ is constant (or, more generally, if $D(p||q)$ goes to zero sublinearly) then we recover the Complete Information setting. Hence, the setting of interest is where $(p - q) \to 0$ as $\tau \to 0$, and the critical scaling is controlled by $\mathcal{D}(p||q)/\tau$.

3.2.3 Main contributions

Our main result consists of two parts. First, we show that there exist graphs that cannot be cured in polynomial time in the Blind Curing model. We then use this result to get a lower bound for the cost of lack of information in the Partial Information model. We obtain an expression for the lower bound that shows the required tradeoff between $\frac{\mathcal{D}(p||q)}{\tau}$ (the information available per unit of time), and the budget, r.

Theorem 3.2.1. *A Partial Information impossibility result.*
We consider the task of curing a fully infected complete balanced binary tree with N nodes. Let $\frac{\mathcal{D}(p||q)}{\tau}$ be a measure of the amount of information we get per time step, and r be the budget (curing rate) of our curing process. If

$$\frac{\mathcal{D}(p||q)}{\tau} \leq \mathcal{O}\left(\frac{\log(N)\sqrt{\log(r)}}{r}\right), \tag{3.1}$$

as $\tau \to 0$, then it is fundamentally impossible for any algorithm (of any computational complexity) to cure the complete binary tree in polynomial expected time with budget $r = \mathcal{O}(W^{\alpha})$, where W is the CutWidth *of the graph and α is any constant.*

For the Blind Curing case, we also have the following *upper bound*.

Theorem 3.2.2. *For all $c > 0$, we can always cure the binary tree in expected linear time with budget $\mathcal{O}(e^{4/c}N^{c})$. In particular, our strategy does not require any information about the state of the nodes.*

Interpreting the result. Suppose that if a node is observed for a fixed period of time, we can estimate its state (infected or not) with probability $1-\delta$ for δ some constant. Our results say that regardless of what this constant is, e.g., even if we have a test that takes 1 minute (or other time unit) to implement and returns a result that is 99% (or any other constant quantity) accurate, then polynomial time curing is impossible, for budget any multiple of the CutWidth. Indeed, as explained above, a constant-error estimate in a fixed unit of time corresponds to $D(p||q)/\tau$, the left-hand side of (3.1), being a constant. On the other hand, if the budget is any multiple of the CutWidth, the right-hand side of (3.1) grows like $\sqrt{\log\log(N)}$, and in particular is larger than any constant. In contrast, with complete (and instantaneous) certainty of the state of each node (here the left-hand side of (3.1) can be infinite), [22] proves that every graph can be cured in linear expected time with budget higher than the CutWidth.

For the blind setting, Theorem 3.5.7 says that for budget of any polynomial of $\log(N)$, curing takes superpolynomial time. Theorem 3.2.2 gives an upper bound that shows that this lower bound is not too far off; it says that a budget of N^c is sufficient, for any $c > 0$. This can be proved by adding a buffer of size $\mathcal{O}(\log(N))$ between the supposedly cured nodes, and the known infected nodes.

Our result focuses on the binary tree. Since our main result is a *lower bound*, this specific example is sufficient to resolve the question of whether the CutWidth (or something proportional to it) is the right quantity to focus on

to build a curing strategy robust to noise in our node estimates. In addition to this, we note that many graphs contain trees as subgraphs. Since adding nodes and edges only makes curing more difficult, our results can be seen to apply to any graph structure with a binary tree as a subgraph (as long as adding edges does not dramatically change the CutWidth of the graph).

Proof Idea. Our proof focuses on bottlenecks of the curing process: events that *must happen with high probability, regardless of the policy used, en route to curing an infection.* Specifically, our proof hinges on showing two such bottlenecks. First, we show that regardless of the policy, regardless of the stochastics of the curing and infection process, with high probability the last nodes to be cured cannot all be far from the root node. As we discuss below, the intuitive reason for this relies on our graph topology, and the fact that the cut between the set of infected nodes and the set of uninfected nodes must remain low if we hope to control the infection. On a binary tree, a simple calculation (Proposition 3.4.9) shows that any $\frac{N}{r^4}$-node set with low cut must contain nodes close to the root. The significance of this result is that at all times that matter (namely, at all points where the curing policy might be close to succeeding), there will be infected nodes that are not far from (exponentially) many uninfected nodes. Next, we show that in any interval of time, there must be many uninfected nodes *that are also unprotected* by the curing policy, regardless of what the curing policy is doing (Lemma 3.4.12). In Theorem 3.4.17, we combine these results to show that the probability that an infection begins, travels through the root to the unprotected subset of nodes

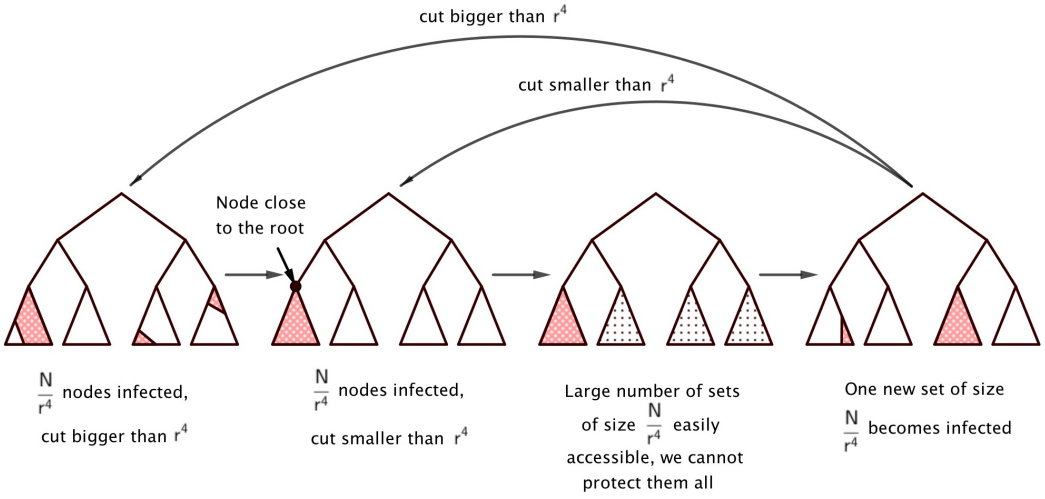


Figure 3.2: Visual representation of the main steps of the proof: when only $\frac{N}{r^4}$ nodes remain infected, no strategy can prevent the reinfection of $\frac{N}{r^4}$ new nodes in some other part of the graph. The graph can only be cured if the cycle is broken, a rare event which takes superpolynomial time in expectation.

and infects them before the remaining nodes are cured, is very close to 1.

3.3 Proof sketch

We first prove that polynomial curing is impossible in the Blind Curing setting if the budget is polynomial in the CutWidth. We then show that in the Partial Information setting, we do not obtain enough information to detect threats of reinfection, and thus cannot prevent them: we are "blind" to the threats until it is too late.

Our proof in the Blind Curing setting focuses on a subprocess which is bound to happen for any curing strategy. We consider the last $\frac{N}{r^4}$ infected nodes. We show that by the time we cure these last remaining infected nodes,

a new set of $\frac{N}{r^4}$ nodes becomes infected with high probability. Trying to cure the whole graph is then similar to playing a very long game of whack-a-mole with superpolynomial expected end time.

3.3.1 Blind Curing setting

Step 1 (Section 3.4.1): We first show that if a strategy allows the cut between the infected and susceptible set to be much higher than the available budget r, the infection becomes uncontrollable. In this case, the infection rate exceeds the curing rate, and the reinfection would be inevitable even if we had complete knowledge about the infection state of each node at each time (*i.e.* this happens even in the Complete Information setting). In particular, if $\frac{N}{r^4}$ nodes are infected and the cut is above r^4, the drift of the curing process is dominated by the infections. We can then use random walks results, such as Wald's Inequality, to prove that after a few time steps, we end up with at least as many infected nodes, but a cut below r^3 (we actually end up with many more infected nodes, but as many is enough for the proof). We can therefore focus on analyzing the situation in which $\frac{N}{r^4}$ nodes remain infected with a cut lower than r^4.

Step 2 (Section 3.4.2): Due to the topology of the binary tree, a cut below r^4 implies that there exists an infected node which is close to the root. This makes it easy for the infection to escape through the root, and reach a large number of susceptible nodes. One key point of the proof is that

this node will remain infected (and therefore potentially infecting) for a very long time, and an infection can start at any time step during this period.

Step 3 (Section 3.4.3): Since the infection escapes through the root, the number of uninfected nodes easily accessible is very large, and specifically, larger than the budget. This makes it impossible to cover all the potential escape routes. Notice that this is very specific to the Blind Curing setting: if we knew in which direction the infection was escaping, we could prevent it as in [24]. It is because the number of potential infected nodes is exponentially higher than the number of actual nodes infected, and because we do not know where the infection actually is, that we end up wasting considerable curing budget on uninfected nodes. Therefore, the infection is very likely to escape, and a new set of $\frac{N}{r^4}$ nodes becomes infected again.

3.3.2 Partial Information setting

To extend this result to the Partial Information setting, we notice that as soon as the cut of the new infection reaches $3r$, we can use Gambler's Ruin results to show that at least $\frac{N}{r^4}$ nodes will become infected with constant probability. If we cannot detect the infection escaping until a cut of $3r$ is reached, we therefore cannot prevent the reinfection with constant probability. Using Sanov's Theorem, we show that the uncertainty in our state estimation for any node does not resolve itself quickly enough (in particular, with respect

to how fast the neighborhoods of the binary tree grow). Specifically, the infection remains undetectable with constant probability until a cut of $3r$ is attained. This allows us to extend the result from the Blind Curing Setting to the Partial Information setting.

3.4 Proof for Blind Curing

In this section, we prove that in the Blind Curing setting, we cannot cure a complete binary tree in polynomial time with budget $\mathcal{O}(W^\alpha)$, where W is the CutWidth, and α is any constant. A complete binary tree has CutWidth smaller than $\log(N)$. Therefore, in the rest of the chapter, we set $r = \log^\alpha(N)$.

We focus on the last moments of the curing, when only $\frac{N}{r^4}$ nodes remain infected. The proof relies on the fact that by the time we cure these last $\frac{N}{r^4}$ nodes, a new set of $\frac{N}{r^4}$ nodes will have become infected in another part of the graph with probability superpolynomially close to 1.

3.4.1 The infection cannot be controlled when the cut is too high

We start by proving that without loss of generality, we can suppose the cut between the infected set and the susceptible set is less than r^4 when $\frac{N}{r^4}$ nodes remain infected. If the cut is above r^4, the infection becomes uncontrollable with high probability, and we end up with at least as many nodes

infected [3], but a cut below r^3 after some time steps. Therefore, supposing the cut is below r^4 only reduces the expected time of curing.

Intuitively, if the budget is much smaller than the cut, the leading term in the drift of the infection process will be driven by the new infections taking place, regardless of the policy in use. Trying to eradicate, or even contain, an epidemic in these conditions would be like fighting an avalanche with a flamethrower: some snow will melt, but it will not stop the avalanche - which will only stop by itself. Similarly, we can only hope to regain some control over the infection process when the budget is at least of the same order of magnitude as the cut.

To prove this result, we introduce a random walk G_t which stochastically dominates the curing process (Lemma 3.4.1). We define a stopping time, T_{SmallCut}, which corresponds to the first time the cut reaches r^3. We prove that by the time we reach this stopping point, many infections must have taken place (Lemma 3.4.4 and 3.4.6), which implies that many time steps must have gone by. We can then use concentration inequalities to prove there are at least as many infected nodes at T_{SmallCut} as there were at the beginning of the random walk (Lemma 3.4.8).

Definition 3.4.1. Let $A_t \sim \mathcal{B}(r, \delta)$, a binomial random variable with r trials and probability δ, and $B_t \sim \mathcal{B}(\frac{r^3}{3}, \mu)$, a binomial with $\frac{r^3}{3}$ trials and probability

[3] It is actually more likely that a large number of new nodes will become infected. However, our proof only requires the total number of infected nodes to not decrease, so this is what we prove.

μ.

We define the random walk G_t:

$$G_t = \sum_{t'=t_0}^{t} A_{t'} - B_{t'}.$$

We are especially interested in the sign of the random variable $G_{T_{\text{SmallCut}}} = \sum_{t'=t_0}^{T_{\text{SmallCut}}} A_{t'} - B_{t'}$.

Definition 3.4.2. We call the **increase in susceptible** (uninfected) **nodes** since t_0 the random variable $I_{t_0} - I_t$, for $t > t_0$. This is the difference between the total number of infected nodes at time t_0, and the total number of infected nodes at time $t > t_0$. In other words, it corresponds to the difference between the number of nodes we successfully cured and the number of newly infected nodes between the times t_0 and t. Note that if more infections than curings have happened since t_0, the increase in susceptible nodes is negative.

Definition 3.4.3. A random variable X_1 is **stochastically dominated** by a random variable X_2, if $\mathbb{P}[X_1 \geq x] \leq \mathbb{P}[X_2 \geq x]$ for all x.

Lemma 3.4.1. *Let t_0 be the first time such that $I_{t_0} = \frac{N}{r^4}$ and the cut is above r^4. The random walk G_t, defined above, stochastically dominates the quantity $I_{t_0} - I_t$ (the increase in susceptible nodes since t_0) for any $t \leq T_{\text{SmallCut}}$, for every strategy.*

Proof. At each time step t, each node i is assigned a budget r_i^t, with $r_i^t \leq r$, and gets cured with probability $\delta_i = 1 - e^{-r_i^t \cdot \tau} \leq \delta = 1 - e^{-r \cdot \tau}$. By assumption

of our model, there are at most r nodes being cured, among which at most r are infected (we do not know for sure if the nodes we are curing are infected or not since we are in the Blind Curing setting). Each of these infected nodes can therefore return to the susceptible state with probability at best δ. In other words, the number of cured nodes is stochastically dominated by a binomial variable with r trials and probability δ, *i.e.,* it is stochastically dominated by A_t.

Before the stopping time, the cut is at least as big as r^3. The maximum degree in a tree is 3, so 3 of these edges could lead to the same node. Therefore, there are at least $\frac{r^3}{3}$ potential infections happening with probability μ. B_t is therefore stochastically dominated by the number of new infections in the curing process, for any strategy.

Thus, G_t stochastically dominates $I_{t_0} - I_t$, for any $t \leq T_{\text{SmallCut}}$, for every strategy.

We use random walks properties to exponentially bound the probability that $G_{T_{\text{SmallCut}}}$ is positive, which correponds to more cures than infections. We recall Wald's Inequality for random walks, whose proof appears in Section 9.4 of [32].

Theorem 3.4.2. *Wald's identity for 2 thresholds*
Let X_i, $i \geq 1$ be i.i.d. and let $\gamma(r) = \log(\mathbb{E}[e^{rX}])$ be the Moment Generating Function (MGF) of X_1. Let $\text{Int}(X)$ be the interval of r over which $\gamma(r)$ exists. For each $n \geq 1$, let $S_n = X_1 + \cdots + X_n$. Let $\epsilon > 0$ and $\beta < 0$ be arbitrary,

and let J be the smallest n for which either $S_n \geq \epsilon$ or $S_n \leq \beta$. Then for each $r \in \text{Int}(X)$:

$$E[\exp(rS_J - J\gamma(r))] = 1.$$

Corollary 3.4.3. *Under the conditions of Theorem 3.4.2, assume that $\mathbb{E}[X] < 0$ and that $r^* > 0$ exists such that $\gamma(r^*) = 0$. Then:*

$$\mathbb{P}[S_J \geq \epsilon] \leq \exp(-r^*\epsilon).$$

We now use Wald's Inequality to prove $I_{t_0} - I_t$ cannot be very large.

Lemma 3.4.4. *If the cut is above r^3, the probability that the increase in susceptible nodes $I_{t_0} - I_t$ is higher than K is exponentially small in K.*

Proof. The curing process is stochastically dominated by the random walk described above. Let P_{curingK} be the probability that G_t reaches the value K before stopping. Using Wald's Inequality (Corollary 3.4.3):

$$P_{\text{curingK}} \leq e^{-x^* \cdot K}.$$

where x^* is a value for which the MGF of G_t is 1. Through careful but standard analysis, we can prove that there exists such a $x^* > 0$, and that x^* converges to $\log(\frac{r}{3})$ when $\tau \to 0$.

Corollary 3.4.5. *The increase in susceptible nodes $I_{t_0} - I_t$ is bounded above by $\frac{\log^2(N)}{x^*}$ with probability at least $1 - e^{-\log^2(N)}$.*

Proof. Using Lemma 3.4.4, we have: $e^{-x^* \cdot K} \geq e^{-\log^2(N)} \implies K \leq \frac{\log^2(N)}{x^*}$. We conclude with setting $K = I_{t_0} - I_t$.

We deduce from the previous result that many infections must have taken place.

Proposition 3.4.6. *At T_{SmallCut} (when the cut reaches r^3), at least $\frac{r^4}{7}$ infections will have taken place.*

Proof. Let C be the number of nodes cured between t_0 and T_{SmallCut} , and I be the number of new infections in the same time period. Any curing or infection reduces the cut by at most 3, since the graph is a binary tree. Therefore:

$$3C + 3I \geq r^4 - r^3.$$

On the other hand, using Corollary 3.4.5, we can bound the increase in susceptible nodes:

$$C - I \leq \frac{\log^2(N)}{x^*}.$$

Combining the two inequalities:

$$I \geq \frac{r^4 - r^3}{6} - \frac{\log^2(N)}{x^*} \geq_{N \gg 1} \frac{r^4}{7}.$$

The previous Proposition proved that many infections happened. We now show this implies that many time steps must have passed by, which allows us to use concentration inequalities. To prove the next Lemma, we recall Hoeffding's Inequality:

Theorem 3.4.7. *(Hoeffding's Inequality for general bounded random variables).*

Let $X_1, \ldots, X_k$ be independent random variables. Assume that $X_t \in [m_t, M_t]$ almost surely for every i. Then, for any $\epsilon > 0$, we have

$$\mathbb{P}\left(\sum_{t=1}^{k}(X_t - \mathbb{E}[X_t]) \geq \epsilon\right) \leq e^{-\frac{2\epsilon^2}{\sum_{t=1}^{k}(M_t - m_t)^2}}.$$

Lemma 3.4.8. *The probability that the random walk reaches the stopping time with $I_{t_0} - I_t < 0$ tends to 0 as $\tau \to 0$.*

Proof. Using Hoeffding's Inequality:

$$\begin{aligned}\mathbb{P}\left(\sum_{t=1}^{k} A_t - B_t \geq 0\right) &= \mathbb{P}\left(\sum_{t=1}^{k} A_t - B_t - \mathbb{E}[A_t - B_t] \geq -k\mathbb{E}[A_t - B_t]\right) \\ &\leq \exp\left(\frac{2 \cdot (k\mathbb{E}[A_t - B_t])^2}{\sum_{t=1}^{k}(r - (-\frac{r^3}{3}))^2}\right) \leq e^{-k\frac{2(r\delta - \frac{r^3}{3}\mu)^2}{(r+\frac{r^3}{3})^2}}.\end{aligned}$$

Let $MoreCuring$ be the event that the increase in susceptible nodes at time T_{SmallCut} $(I_{t_0} - I_{T_{\text{SmallCut}}})$ is non-negative. We use Hoeffding's Inequality to bound $\mathbb{P}\left(\sum_{t=1}^{k} A_t - B_t \geq 0\right)$. Then, b Proposition 3.4.6, we know that at least $I = \frac{r^4}{7}$ infections must have taken place. To simplify the notations for this proof, we introduce two new stopping times, $T_{\text{ManyInfections}}$ and $T_{\text{NegBinomialRW}}$. T_{SmallCut} stochastically dominates $T_{\text{ManyInfections}}$, the number of time steps it takes for the random walk to infect I new nodes. Since the infection rate is at least $\frac{r^3}{3}$, $T_{\text{ManyInfections}}$ in turn stochastically dominates $T_{\text{NegBinomialRW}}$, a negative binomial distribution of parameter $\frac{3I}{r^3}$ and probability of failure μ. We can therefore replace T_{SmallCut} by the simpler quantity $T_{\text{NegBinomialRW}}$ in

the following calculations:

$$\begin{aligned}
\mathbb{P}(MoreCuring) &= \mathbb{P}\left(\sum_{t=1}^{T_{\text{SmallCut}}} A_t - B_t \geq 0\right) \\
&= \sum_{k=0}^{\infty} \mathbb{P}\left(\sum_{t=1}^{k} A_t - B_t \geq 0\right) \cdot \mathbb{P}\left(T_{\text{SmallCut}} = k\right) \\
&\leq \sum_{k=0}^{\infty} e^{-k\frac{2(r^3\mu - r\delta)^2}{(r^3+r)^2}} \cdot \mathbb{P}\left(T_{\text{NegBinomialRW}} = k\right) \\
&\leq e^{-\frac{I}{r^3}\frac{6(r^3\mu - r\delta)^2}{(r^3+r)^2}} \sum_{k=0}^{\infty} e^{-k\frac{2(r^3\mu - r\delta)^2}{(r^3+r)^2}} \cdot \mathbb{P}\left(T_{\text{NegBinomialRW}} = \frac{3I}{r^3} + k\right) \\
&\leq e^{-\frac{I}{r^3}\frac{6(r^3\mu - r\delta)^2}{(r^3+r)^2}} \left(\frac{\mu}{1 - \mu e^{-\frac{2(r^3\mu - r\delta)^2}{(r^3+r)^2}}}\right)^{\frac{3I}{r^3}} \to_{\tau \to 0} 0,
\end{aligned}$$

where we have used that the MGF of a negative binomial of parameter M, probability of success p, evaluated at u, is $\left(\frac{1-p}{1-e^u p}\right)^M$.

3.4.2 There exists an infected node close to the root

From the moment we start curing the last $\frac{N}{r^4}$ nodes, to the moment we have cured half of them and only $\frac{N}{2r^4}$ of these nodes remain infected, we show in this section that there exists an infected node at distance $\mathcal{O}(\log\log(N))$ from the root (Proposition 3.4.9). This node stays infected for a high number of steps (Proposition 3.4.10).

Proposition 3.4.9. *If we select a set of $\frac{N}{2r^4}$ nodes in a tree such that the cut of this set is lower than r^4, then there is at least one node from this set at distance $9\log(r) = 9\alpha\log\log(N) = \mathcal{O}(\log\log(N))$ from the root.*

Proof. We prove the contrapositive: if all the nodes of this set are at distance greater than $9\alpha \log\log(N)$ from the root, then the cut is higher than r^4.

Any subtree rooted at distance $9\alpha \log\log(N)$ from the root contains $\frac{N}{r^9}$ nodes, and has a cut of at least 1. Suppose all the $\frac{N}{2r^4}$ nodes of the selected set are at distance $9\alpha \log\log(N)$ or more from the root. We therefore need at least $\frac{N}{2r^4}/\frac{N}{r^9} = \frac{r^5}{2}$ such subtrees, for a total cut of at least $\frac{r^5}{2} > r^4$. Hence, the closest node is at distance at most $9\alpha \log\log(N) = \mathcal{O}(\log\log(N))$ from the root.

We now show it takes many time steps to cure $\frac{N}{2r^4}$ nodes, regardless of the policy.

Proposition 3.4.10. *Curing half of the last $\frac{N}{r^4}$ nodes requires more than $\frac{N}{2r^4} \cdot \frac{1}{\delta}$ time steps in expectation.*

Proof. If we ignore any potential infections, the time needed to cure $\frac{N}{2r^4}$ nodes is at least the sum of $\frac{N}{2r^4}$ geometric random variables of parameter δ. The result follows by linearity of expectation.

Proposition 3.4.11. *Let $T_{\frac{N}{2r^4}}$ be the random variable representing the time to cure half of the $\frac{N}{r^4}$ last nodes. Then:*

$$\mathbb{P}\left(T_{\frac{N}{2r^4}} \leq \frac{N}{4r^5\delta}\right) \leq e^{-\frac{N}{8r^5}}.$$

Proof. This follows from representing geometric random variables as the sum of Bernoulli random variable, which allows us to use Chernoff bounds.

Therefore, there exists an infected node close to exponentially many uninfected nodes, during at least $\frac{N}{4r^5\delta}$ time steps. We now establish a lower bound on the probability of reinfecting $\frac{N}{r^4}$ new nodes in some other part of the graph, starting from this node.

3.4.3 Low-cut case

We prove in this section that the probability of infecting $\frac{N}{r^4}$ new nodes in some other part of the graph, by the time it takes to cure half of the $\frac{N}{r^4}$ last infected nodes, is superpolynomially close to 1 for every strategy (Lemma 3.4.15). The graph can only be cured if this does not happen.

The following Lemma is key to understanding why no strategy can prevent the reinfection. In the Blind Curing setting, we do not know which nodes are infected. Since there are exponentially many infection routes from the root of the tree, spreading the budget means there will always be a subtree on which very small budget is allocated. If the infection reaches this tree, reinfecting a lot of nodes becomes very likely.

Lemma 3.4.12. *For every time t_0, there exist r subtrees containing $\frac{N}{r^3}$ nodes for which less than $\frac{t_3}{r}$ budget is used in the interval $[t_0, t_0 + t_3]$. We call any of these trees a* ***minimal tree*** *for $[t_0, t_0 + t_3]$.*

Proof. By the pigeonhole principle, since the total budget during this interval is $t_3 \cdot r$, and there are at least r^3 disjoint subtrees containing $\frac{N}{r^3}$ nodes, there are at least r subtrees that contain less than $r \cdot \frac{t_3 \cdot r}{r^3} = \frac{t_3}{r}$ budget on this interval.

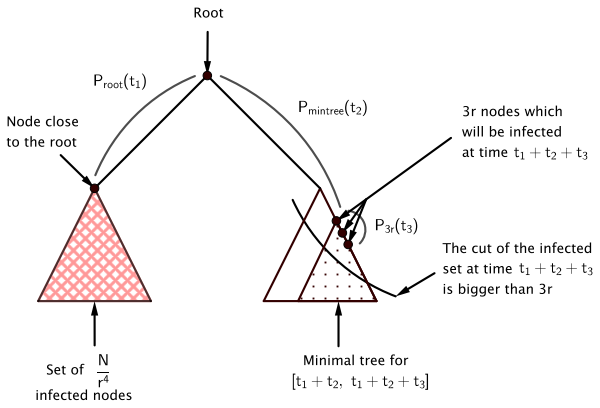


Figure 3.3: Visual representation of $P_{\text{root}}(t_1)$, $P_{\text{mintree}}(t_2)$, and $P_{3\text{r}}(t_3)$.

Definition 3.4.4. From Proposition 3.4.9, we know there exists an infected node close to the root. We call an **Escape**(t_0, t_1, t_2, t_3) the conjunction of the following events:

1. At time t_0, this node infects its parent.

2. The infection propagates from the parent node to the root in time t_1, without any node being cured.

3. The infection propagates from the root of the tree to the root of a minimal tree for $[t_0 + t_1 + t_2, t_0 + t_1 + t_2 + t_3]$ in time t_2, without any node being cured.

4. $3r$ new nodes are infected in a minimal tree for $[t_0+t_1+t_2, t_0+t_1+t_2+t_3]$ in time t_3, without any node in a minimal tree being cured.

5. The number of newly infected nodes reaches $\frac{N}{r^4}$ before it reaches r.

We notice that if an $Escape$ happens, then $\frac{N}{r^4}$ new nodes in some other part of the graph were reinfected. However, it is possible to reinfect $\frac{N}{r^4}$ new nodes without any $Escape$ happening.

If $\{t_0, t_1, t_2, t_3\} \neq \{t'_0, t'_1, t'_2, t'_3\}$, then $Escape(t_0, t_1, t_2, t_3)$ and $Escape(t'_0, t'_1, t'_2, t'_3)$ are disjoint events.

We notice that the probability of all the events defined above is independent of t_0. To simplify notations, we set $t_0 = 0$ for the following definitions.

- $P_{\text{root}}(t_1)$, the probability that the infection reaches the root in time exactly t_1.
- $P_{\text{mintree}}(t_2)$, the probability that the infection reaches a minimal tree for $[t_1 + t_2,\, t_1 + t_2 + t_3]$ in time t_2, conditioned on the fact that the root of the tree is infected. Interestingly, by symmetry of the binary tree (all potential minimal trees are at the same distance from the root of the tree), this quantity does not depend on t_1 or t_3.
- $P_{3\text{r}}(t_3)$, the probability that $3 \cdot r$ nodes are reinfected in a minimal tree in time t_3, conditioned on the fact that the root of a minimal tree is infected.
- P_{spread}, the probability that the increase in susceptible nodes since time $t_1 + t_2 + t_3$ reaches $-\frac{N}{r^4} + 3 \cdot r$ before it reaches $2r$, conditioned on the fact that $3r$ new nodes are infected at time $t_1 + t_2 + t_3$.

The proof relies on the fact that *no strategy can adapt to the infection moving towards a minimal tree.* In the Blind Curing setting, most of the budget is wasted covering nodes which are not infected or about to be infected, while most of the graph is left unprotected.

We now bound the probabilities defined above.

Proposition 3.4.13.

- $P_{\text{root}}(t_1) \geq \binom{t_1}{9\alpha \log\log(N)} \cdot \mu^{9\alpha \log\log(N)+1}(1-\mu)^{t_1-9\alpha \log\log(N)}(1-\delta)^{t_1}$,
- $P_{\text{mintree}}(t_2) \geq \binom{t_2}{3\alpha \log\log(N)} \cdot \mu^{3\alpha \log\log(N)+1}(1-\mu)^{t_2-3\alpha \log\log(N)}(1-\delta)^{t_2}$,
- $P_{3\text{r}}(t_3) \geq \binom{t_3}{3r} e^{-\frac{t_3}{r}\cdot\tau} \cdot \mu^{3r+1}(1-\mu)^{t_3-3r} e^{-\frac{t_3}{r}\cdot\tau}$.

Proof. These are straightforward combinatorics calculations.

Proposition 3.4.14. *Conditioned on the cut of the infected set being at least $3r$, the probability that the increase in susceptible nodes since time $t_1+t_2+t_3$ reaches $-\frac{N}{r^4}+3\cdot r$ before it reaches $2r$, is at least $\frac{1-\frac{1}{2}^{3\cdot r}}{1-\frac{1}{2}^{\frac{N}{r^4}}} \geq \frac{1}{2}$.*

Proof. This is a classic Gambler's Ruin problem, with low boundary $2r$ and high boundary $\frac{N}{r^4}$. During the infection process, which starts with $3r$ infected nodes, the cut is always higher than $2r$, so the infection rate is always higher than $2r$, while the curing rate is r. The proof can be found in [37].

We now combine the previous results to bound the probability of escaping in one time step.

Lemma 3.4.15. *Let $P_{\text{escapeOneStep}}$ be the probability that an Escape starts at a given time step. Then:*

$$P_{\text{escapeOneStep}} \geq \mu \left(\frac{\mu(1-\delta)}{\delta + \mu(1-\delta)} \right)^{9\alpha \log\log(N)} \cdot \left(\frac{\mu e^{-\frac{1}{r}}}{(1-e^{-\frac{1}{r}}) + \mu(e^{-\frac{1}{r}})} \right)^{3r} \cdot \frac{1}{2}.$$

Therefore, for τ sufficiently small (and in particular, as $\tau \to 0$),

$$P_{\text{escapeOneStep}} \geq \frac{\tau}{2e^3 e^{12\alpha^2 \log^2 \log(N)}} + o(\tau).$$

Proof. We notice $P_{\text{root}}(t_1)$ only depends on t_1, $P_{\text{mintree}}(t_2)$ only depends on t_2, $P_{3\text{r}}(t_3)$ only depends on t_3, while P_{spread} is independent of t_1, t_2, and t_3. We obtain the values of $P^{\text{startPath}}$ and P^{path} through standard combinatorics:

$$\begin{aligned}
P_{\text{escapeOneStep}} &= \sum_{t_1,t_2,t_3=0}^{\infty} P_{\text{root}}(t_1) \cdot P_{\text{mintree}}(t_2) \cdot P_{3\text{r}}(t_3) \cdot P_{\text{spread}} \\
&= \sum_{t_1=0}^{\infty} P_{\text{root}}(t_1) \cdot \sum_{t_2=0}^{\infty} P_{\text{mintree}}(t_2) \cdot \sum_{t_3=0}^{\infty} P_{3\text{r}}(t_3) \cdot (P_{\text{spread}}) \\
&\geq P^{\text{startPath}}_{9\alpha \log\log(N)} \cdot P^{\text{path}}_{3\alpha \log\log(N)} \cdot \left(\frac{\mu e^{-\frac{\tau}{r}}}{(1-e^{-\frac{\tau}{r}}) + \mu(e^{-\frac{\tau}{r}})} \right)^{3r} \cdot P_{\text{spread}} \\
&\geq \mu \left(\frac{\mu(1-\delta)}{\delta + \mu(1-\delta)} \right)^{12\alpha \log\log(N)} \cdot \left(\frac{\mu e^{-\frac{\tau}{r}}}{(1-e^{-\frac{\tau}{r}}) + \mu(e^{-\frac{\tau}{r}})} \right)^{3r} \cdot \frac{1}{2}.
\end{aligned}$$

As $\tau \to 0$:

$$\begin{aligned}
P_{\text{escapeOneStep}} &\sim_{\tau \to 0} \tau \left(\frac{\tau}{(r+1)\tau} \right)^{12\alpha \log\log(N)} \cdot \left(\frac{\tau}{(\frac{1}{r}+1)\tau} \right)^{3r} \cdot \frac{1}{2} \\
&\geq_{\tau \to 0} \tau \left(e^{-12\alpha^2 \log^2 \log(N)} \right) \cdot \frac{e^{-\log(1+\frac{1}{r}) \cdot 3r}}{2} + o(\tau) \\
&\geq_{\tau \to 0} \tau \left(e^{-12\alpha^2 \log^2 \log(N)} \right) \cdot \frac{e^{-3}}{2} + o(\tau) \\
&\geq_{\tau \to 0} \frac{\tau}{2e^3 e^{12\alpha^2 \log^2 \log(N)}} + o(\tau).
\end{aligned}$$

We therefore deduce the probability that no *Escape* happens by the time we cure half of the $\frac{N}{r^4}$ infected nodes.

Lemma 3.4.16. *Let NoEscape be the event that no Escape happens by the time we cure half of the $\frac{N}{r^4}$ infected nodes.*

$$\mathbb{P}\left(NoEscape\right) \leq_{N\gg 1} e^{-\frac{N}{e^{24\alpha^2\log^2\log(N)}}}.$$

Proof. Since there are always more than $\frac{N}{2r^4}$ nodes infected, there is at least one infected node at distance $9\alpha\log\log(N)$ from the root (Lemma 3.4.9), which means the bound for $P_{\text{escapeOneStep}}$ established in Lemma 3.4.15 holds. We split the analysis into two cases: whether we can cure $\frac{N}{2r^4}$ in less than $\frac{N}{4r^5\delta}$ time steps or not. The probability of one *Escape* starting at time t being independent from the probability of an *Escape* starting at any other time step t':

$$\begin{aligned}
\mathbb{P}\left(NoEscape\right) &\leq \left(1-P_{\text{escapeOneStep}}\right)^{T_{\frac{N}{2r^4}}} \\
&\leq \mathbb{P}(T_{\frac{N}{2r^4}} \leq \frac{N}{4r^5\delta})\cdot 1 + \mathbb{P}(T_{\frac{N}{2r^4}} \\
&\geq \frac{N}{4r^5\delta})\cdot\left(1-P_{\text{escapeOneStep}}\right)^{\frac{N}{4r^5\delta}} \\
&\leq e^{-\frac{N}{8r^5}} + \left(1-P_{\text{escapeOneStep}}\right)^{\frac{N}{4r^5\delta}} \\
&\leq e^{-\frac{N}{8r^5}} + \left(1-\left(\mu\frac{\mu(1-\delta)}{\delta+\mu(1-\delta)}\right)^{12\alpha\log\log(N)}\right)^{\frac{N}{4r^5\delta}} \\
&\leq_{\tau\to 0, N\gg 1} e^{-\frac{N}{e^{24\alpha^2\log^2\log(N)}}}.
\end{aligned}$$

3.4.4 A Blind Curing result

From Sections 3.4.1 and 3.4.3, we know the graph can only be cured if we are in one of these two cases:

1. The cut was above r^4, but we cured the whole graph anyway, which happens with probability less than $e^{-\log^2(N)}$ (Proposition 3.4.5)

2. The cut was below r^4, but no *Escape* happens by the time it takes to cure half of $\frac{N}{r^4}$ infected nodes, which happens with probability less than $e^{-\frac{N}{e^{24\alpha^2}\log^2\log(N)}}$ (Lemma 3.4.16).

We can therefore obtain a bound on the expected time it takes to cure the whole graph.

Theorem 3.4.17. *In the Blind Curing setting, curing a complete binary tree takes* $\Omega\left(e^{\log^2(N)}\right)$ *time in expectation with any budget polynomial in the* CutWidth. *Therefore, no polynomial expected time curing strategy exists for budget* $r = \mathcal{O}(W^\alpha) = \mathcal{O}(\log^\alpha(N))$*, for all* α *constant.*

Proof. Let $CureLastNodes$ be the event that we are in case (1) or (2) described above. By union bound:

$$\begin{aligned}\mathbb{P}\left(CureLastNodes\right) &\leq e^{-\log^2(N)} + e^{-\frac{N}{e^{24\alpha^2}\log^2\log(N)}} \\ &\leq 2e^{-\log^2(N)}.\end{aligned}$$

We have $\frac{1}{r} \leq \frac{\tau}{\delta}$. The number of times we try to cure the last $\frac{N}{r^4}$ is stochastically bounded below by a geometric variable of parameter $\mathbb{P}\left(CureLastNodes\right)$. Following Proposition 3.4.10, curing $\frac{N}{2r^4}$ lasts at least $\frac{N}{2r^4} \cdot \frac{1}{\delta}$ time steps, so $\frac{N}{2r^4} \cdot \frac{\tau}{\delta}$ time. Therefore, the expectation of the length of the curing process is the number of times we try to cure the last infected nodes, multiplied by the time it takes to cure them.

$$\mathbb{E}(\text{Length}) \geq \underbrace{\frac{1}{\mathbb{P}\left(CureLastNodes\right)}}_{\substack{\text{expected number}\\ \text{of times we try}\\ \text{to cure } \frac{N}{r^4} \text{ nodes}}} \cdot \underbrace{\frac{N}{2r^4\delta}}_{\substack{\text{minimal number of}\\ \text{time steps to}\\ \text{cure } \frac{N}{r^4} \text{ nodes}}} \cdot \underbrace{\tau}_{\substack{\text{size of a}\\ \text{time step}}}$$

$$\geq \frac{e^{\log^2(N)}}{2} \cdot \frac{N}{2r^4} \cdot \frac{1}{r} = \Omega\left(e^{\log^2(N)}\right).$$

Hence, it is not curable in polynomial time for budget $r = \mathcal{O}(W^\alpha) = \mathcal{O}(\log^\alpha(N))$, for all α constant.

3.5 Proof for Partial Information

Definition 3.5.1. We call **sample** the information given by one node at a given time step (i.e., whether a flag was raised or not). We call an **infected-sample** a sample from an infected node.

When an *Escape* happens, we show that with constant probability, not too many infected-samples are produced. In particular, by the time reinfecting $\frac{N}{r^4}$ new nodes becomes inevitable with constant probability, not enough infected-samples are produced to determine if one of the minimal trees' nodes is infected with better than constant error probability. In other words, no strategy can utilize the information available without making mistakes a constant fraction $P_{confuse}$ of the time.

If we cannot recognize that an infection has happened before it is too late to prevent it, everything is as if we were in the Blind Curing model. We can therefore extend the results from the previous section.

3.5.1 Quantity of information available before the cut reaches $3r$

If we reuse the terms introduced in Section 3.4.3, the *Escapes* we consider are composed of four phases:

1. reaching the root
2. reaching the root of a minimal tree. There are r possible such minimal trees.
3. infecting $3r$ nodes in this minimal tree
4. spreading the infection from $3r$ to $\frac{N}{r^4}$ nodes

Since the spreading phase (4) happens with constant probability even in the Complete Information setting, we focus on the number of samples created by the first three phases. We focus in particular on the number of samples created by phase (3) in Lemma 3.5.1. We show in the proof of Lemma 3.5.2 that the number of samples produced by phases (1) and (2) is negligible compared to the number of samples produced by phase (3). To make sure that no other infected-samples can be gathered, we forbid infections from happening outside of an *Escape*.

Let us notice that every time a new infection takes place, the cut increases by 1 (every new node infected gives access to 2 new nodes, but the edge leading to it is not part of the cut any longer).

The next lemma says that in the event of an infection in a minimal tree, it is likely that the number of infected-samples we obtain is small.

Lemma 3.5.1. *In the event that the root of the minimal tree becomes infected, then conditioned on the event that $3r$ new nodes of this minimal tree become infected, we gather at most $\frac{6r}{\tau}$ samples from the newly infected nodes, with probability at least $\frac{1}{2}$.*

Proof. Let N_{samples} be the number of samples produced by infected nodes from the time one node was infected to the moment the $3r^{th}$ node was infected. The time to infect one more node when i nodes are infected is given by a geometric variable $Geo(i, \mu)$ of parameter $1 - (1 - \mu)^i$. Therefore, the jth node to be infected produces $\sum_{i=j}^{3r-1} Geo(i, \mu)$ samples. Thus, conditioned on the event that $3r$ nodes become infected:

$$N_{\text{samples}} = \sum_{j=1}^{3r-1} \sum_{i=j}^{3r-1} Geo(i, \mu) = \sum_{j=1}^{3r-1} j \cdot Geo(j, \mu).$$

Therefore, again conditioned on $3r$ nodes becoming infected, the expected number of samples is:

$$\begin{aligned} \mathbb{E}(N_{\text{samples}} \,|\, 3r \text{ infected}) &= \sum_{j=1}^{3r-1} j \cdot \mathbb{E}(Geo(j, \mu)) = \sum_{j=1}^{3r-1} j \cdot \frac{1}{1 - (1 - \mu)^j} \\ &=_{\tau \to 0} \sum_{j=1}^{3r-1} j \cdot \frac{1}{j\tau} \qquad \leq_{\tau \to 0} \frac{3r}{\tau}. \end{aligned}$$

We conclude by using Markov's Inequality.

We now count the number of samples available from phases (1), (2) and (3) of an *Escape*.

Lemma 3.5.2. *Conditioned on the event that an Escape happened, we gather at most* $\frac{6r}{\tau}$ *infected-samples from phase (3), and* $\frac{360\log^2(r)}{\tau}$ *infected-samples from phases (1) and (2), with probability at least* $\frac{1}{4}$.

Proof. Using Proposition 3.4.9, there are at most $9\log(r)$ nodes from one infected node to the root. Using Proposition 3.4.12, the minimal tree is at distance $3\log(r)$ from the root. We then need $3r$ additional infections to get to a point where the infection is unstoppable (Proposition 3.4.14). Using Markov's Inequality, we can infect these $9\log(r) + 3\log(r) = 12\log(r)$ nodes in $\frac{24\log(r)}{\tau}$ time steps with probability $\frac{1}{2}$. Using standard concentration inequalities, we can infect the $3r$ nodes in $\frac{2\log(3r)}{\tau} \leq \frac{6\log(r)}{\tau}$ time steps with probability $\frac{1}{2}$. Therefore, we can infect all these nodes in $\frac{30\log(r)}{\tau}$ time steps with probability $\frac{1}{4}$, which gives at most $\frac{30\log(r)}{\tau} \cdot 12\log(r)$ samples for the first $12\log(r)$ nodes, and $2 \cdot \frac{3r}{\tau} = \frac{6r}{\tau}$ samples for the last $3r$ nodes, which concludes the proof.

Conditioned on reaching a cut of $3r$ in a minimal tree in less than $\frac{30\log(r)}{\tau}$ time steps, we now bound the probability of not infecting any nodes which are not part of the *Escape*. This ensures that the only infected samples we could get come from the nodes in the *Escape*.

Proposition 3.5.3. *Conditioned on reaching a cut of* $3r$ *in a minimal tree in less than* $\frac{30\log(r)}{\tau}$ *time steps, the probability* $P_{\text{NoOtherInfections}}$ *of not infecting any nodes outside of the Escape is bounded by:*

$$P_{\text{NoOtherInfections}} \leq e^{-\frac{360\log^2(r)\mu}{\tau}} \leq_{\tau\to 0} e^{-360\log^2(r)}.$$

Proof. The proof is similar to the other calculations so far.

3.5.2 Distinguishing between infected and not infected

The proof relies on this idea: any strategy attempting to prevent an *Escape* needs to shift its budget towards the minimal tree in which the infection is progressing. For this to happen, it is necessary to realize that one of the minimal trees is threatened. Determining which one amounts to distinguishing between the following hypotheses:

- H_0: In the null hypothesis, none of the r minimal trees have any infected nodes.
- H_1: One of the r minimal trees has at least one infected node, while the others do not.

We use the results in Lemma 3.5.4 to show that there are not enough infected-samples created by nodes on the path from the node close to the root to the root of a minimal tree to realize that even one node is infected. This implies that the nodes from phase (1) and (2) cannot help detect a threat to a minimal tree. Lemma 3.5.4 is also used to show that we do not gather enough infected-samples from phase (3) to distinguish between H_0 and H_1 defined above, which means we cannot know if there is at least one infected node in one minimal tree.

Thus, we combine all these results to calculate $\mathbb{P}(NoEscapePI)$, the probability that an *Escape* happens, that not too many samples are produced

during this *Escape*, that no other nodes are infected outside of the *Escape*, that the samples from phase (1) and (2) do not allow the identification of the infected minimal tree, and that the samples from phase (3) are not enough to reveal whether or not one minimal tree is indeed infected.

We finally use these results to extend the Blind Curing theorem to the Partial Information setting (Theorem 3.5.7).

Lemma 3.5.4. *1. We need* $\Omega\left(\frac{\log\left(\frac{1}{\epsilon}\right)}{\mathcal{D}(p||q)}\right)$ *samples to decide if a node is infected or not with probability at least* $\Omega\left(1-\epsilon\right)$.

2. We need $\Omega\left(\frac{\log\left(\frac{1}{\epsilon}\right)\sqrt{\log(r)}}{\mathcal{D}(p||q)}\right)$ *samples to distinguish between hypothesis* H_0 *and* H_1*, and detect if one minimal tree has at least one node infected among* r *minimal trees.*

Proof. 1. Using Sanov's Theorem [70], following the proof in Proposition 5.6 of [56], we know that we need $\Omega\left(\frac{1}{\mathcal{D}(p||q)}\right)$ samples to distinguish between two coins of parameters p and q with probability $\Omega(1)$. We can boost this probability to show that we need $\Omega\left(\frac{\log\left(\frac{1}{\epsilon}\right)}{\mathcal{D}(p||q)}\right)$ to distinguish between p and q with probability $\Omega\left(1-\epsilon\right)$.

2. This follows by considering the maximum of r $\mathcal{B}(n,q)$ binomial random variables. Using a Gaussian approximation to the binomial, and the fact that the maximum of r standard Gaussian random variables is $\sqrt{2\log(r)}$, we obtain the desired result.

Corollary 3.5.5. *The probability P_{confuse} of not being able to detect which minimal tree is infected during phase (3) is bounded away from 0.*

In particular, we have:

$$P_{\text{confuse}} =_{N \gg 1} \Omega\left(e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot \frac{r}{\sqrt{\log(r)}}}\right).$$

Proof. We separate the samples in two groups:

- The sample from phase (1) and (2) are used to detect if one node was infected on the path from the node close to the root to the root of a minimal tree. We get $\frac{360 \log^2(r)}{\tau} = \frac{\log\left(e^{\frac{\mathcal{D}(p||q)}{\tau} \cdot 360 \log^2(r)}\right)}{\mathcal{D}(p||q)}$ samples from phase (1) and (2) (Lemma 3.5.1), so the probability of confusing coins of parameters p and q is at least $\Omega\left(e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot 360 \log^2(r)}\right)$ according to Lemma 3.5.4.

- The samples from all phases are used to distinguish between H_0 and H_1. We get $\frac{6r}{\tau} = \frac{\log\left(e^{\frac{\mathcal{D}(p||q)}{\tau} \cdot \frac{6r}{\sqrt{\log(r)}}}\right)\sqrt{\log(r)}}{\mathcal{D}(p||q)}$ samples from phase (1) and (2) (Lemma 3.5.1), so the probability of confusing coins of parameters p and q is at least $\Omega\left(e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot \frac{6r}{\sqrt{\log(r)}}}\right)$ according to Lemma 3.5.4.

Combining the two, the probability of not detecting the threat is at least:

$$P_{\text{confuse}} = \Omega\left(e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot \left(\frac{6r}{\sqrt{\log(r)}} + 360 \log^2(r)\right)}\right) =_{N \gg 1} \Omega\left(e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot \frac{r}{\sqrt{\log(r)}}}\right).$$

We now consider the time needed to cure the graph for $P_{\text{confuse}} > 0$.

Lemma 3.5.6. *Let EscapePI be the event that by the time it takes to cure half of $\frac{N}{r^4}$ infected nodes, an Escape happens but remains undetectable (i.e., the samples produced by the newly infected nodes during phases (1), (2), and (3) are not enough to deduce that there exists an infected minimal tree), and no node outside of the Escape becomes infected. We provide a bound for NoEscapePI, the complementary of this event.*

$$\mathbb{P}(NoEscapePI) \leq e^{-\frac{P_{\text{NoOtherInfections}} \cdot P_{\text{confuse}} \cdot N}{e^{96\alpha^2 \log^2 \log(N)}}}.$$

Proof. Let *EscapeOneStepPI* be the conjunction of all the following events:

- An *Escape* happens at a given time step, which happens with probability at least $P_{\text{EscapeOneStep}}$ (Lemma 3.4.15).
- Conditioned on an *Escape* happening, less than $\frac{6r}{\tau}$ samples are produced by the newly infected nodes during phase (3), and less than $\frac{360 \log^2(r)}{\tau}$ infected-samples are produced during phase (1)-(2), which happens with probability at least $\frac{1}{4}$ (Lemma 3.5.1).
- Conditioned on an *Escape* happening in less than $\frac{30 \log(r)}{\tau}$ time steps, no node outside of the *Escape* becomes infected, which happens with probability $P_{\text{NoOtherInfections}}$.
- Conditioned on all the above, the samples from phase (3) are not enough to reveal whether or not one minimal tree is indeed infected, which happens with probability P_{confuse}.

We notice that if we cannot tell whether or not a minimal tree is infected by the time it takes to reach phase (4) of an *Escape*, the situation is almost equivalent to the Blind Curing model. We can therefore apply exactly the same reasoning as in Theorem 3.4.17 if we replace $\mathbb{P}(EscapeOneStep)$ by $\mathbb{P}(EscapeOneStepPI)$.

$$\mathbb{P}(EscapeOneStepPI) \geq P_{\text{EscapeOneStep}} \cdot \frac{1}{4} \cdot P_{\text{NoOtherInfections}} \cdot P_{\text{confuse}},$$

Following the exact same reasoning as in Lemma 3.4.15, we get:

$$\mathbb{P}(NoEscapePI) \leq e^{-\frac{P_{\text{NoOtherInfections}} \cdot P_{\text{confuse}} \cdot N}{e^{96\alpha^2 \log^2 \log(N)}}}.$$

Theorem 3.5.7. *A Partial Information impossibility result*

Let $\frac{\mathcal{D}(p||q)}{\tau}$ *be a measure of the amount of information we get by time step. If:*

$$\begin{aligned}\frac{\mathcal{D}(p||q)}{\tau} &\leq \mathcal{O}\left(\left(\log\left(\frac{N}{e^{456\alpha^2 \log^2 \log(N)}}\right) - 2\log\log(N)\right)\frac{\sqrt{\log(r)}}{r}\right) \\ &= \mathcal{O}\left(\frac{\log(N)\sqrt{\log(r)}}{r}\right),\end{aligned}$$

we cannot cure the complete binary tree in polynomial expected time with budget $r = W^{\alpha}$*, for any* α *constant.*

Proof. From Lemma 3.5.6, we know:

$$\mathbb{P}(NoEscapePI) \leq e^{-\frac{P_{\text{confuse}} \cdot P_{\text{NoOtherInfections}} \cdot N}{e^{96\alpha^2 \log^2 \log(N)}}}.$$

From Corollary 3.5.5, we know:

$$P_{\text{confuse}} \geq \Omega\left(e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot \frac{r}{\sqrt{\log(r)}}}\right).$$

$$
\begin{aligned}
\mathbb{P}(NoEscapePI) &\geq e^{-\frac{P_{\text{NoOtherInfections}} \cdot P_{\text{confuse}} \cdot N}{e^{96\alpha^2 \log^2 \log(N)}}} \\
&\geq e^{-\mathcal{O}\left(\frac{e^{-\frac{\mathcal{D}(p||q)}{\tau} \cdot \frac{r}{\log(r)} \cdot N}}{e^{96\alpha^2 \log^2 \log(N) + 12 \log^2(r)}}\right)} \\
&\geq e^{-\mathcal{O}\left(\frac{e^{-\cdot \left(\log\left(\frac{N}{e^{456\alpha^2 \log^2 \log(N)}}\right) - 2\log\log(N)\right) \cdot \frac{r\sqrt{\log(r)}}{\sqrt{\log(r)}r} \cdot N}}{e^{456\alpha^2 \log^2 \log(N)}}\right)} \\
&\geq e^{-\mathcal{O}\left(\log^2(N)\right)}.
\end{aligned}
$$

Following the same reasoning as in Theorem 3.4.17, we conclude it takes at least $\frac{e^{\Omega\left(\log^2(N)\right)} \cdot N}{2\log^{4\alpha}(N)} \geq e^{\Omega\left(\log^2(N)\right)}$ time to cure the graph, so more than any polynomial expected time.

In particular, this holds for $\alpha = 1$. If we remember that the CutWidth of a tree is smaller than $\log(N)$, we obtain:

Corollary 3.5.8. *If the quantity of information by time step measured by* $\frac{\mathcal{D}(p||q)}{\tau}$ *is constant, no strategy can achieve polynomial time curing for the complete binary tree in the Partial Information setting, for budget* $\mathcal{O}(\log(N))$.

3.6 Conclusion

We have shown that unless we know the state of each node with perfect accuracy, and instantaneously, then the CutWidth of the graph is no longer the sole quantity which determines the budget required to cure an infection in polynomial time. Practically, this means that quickly obtaining signature-based diagnostic tools, even if expensive, is critical. On the theoretical side,

our work shows that the interplay between stochastic processes and combinatorial properties of graphs needs to be better understood. Indeed, resolving the gap between our upper and lower bounds as a function of general topological graph quantities remains an important question. Similarly, extending our understanding of upper and lower bounds to other infection models is important. This work demonstrates the important connection between budget for control, and budget for estimation, as for many interesting problems, these two are inextricably intertwined.

Chapter 4

Uncertainty about When Nodes Are Infected

4.1 Introduction

After focusing on curing in the previous chapter, we turn to another important problem in the field of epidemics: the problem of graph inference from epidemic cascades[1]. If curing relies on the knowledge of who exactly is infected, in the case of graph inference, the crucial information is the time of infection, as this is what state-of-the-art algorithms use to reconstruct the graph. once again, as the COVID-19 pandemic has demonstrated, we rarely have access to the real time of infection, as individuals show symptoms after a non-deterministic time, and then opt to get tested after another non-deterministic time. Previous algorithms cannot handle such a noise in the time of infection. We therefore develop completely new algorithms to tackle this problem, in two settings: when we only know who was infected at the end of the cascade, and when we know the noisy times of infection and some parameter of the noise distribution. Our algorithms are sample-optimal up to log factors.

[1]This chapter covers the material prebiously published in *Learning graphs from noisy epidemic cascades*. My main contributions were defining the setting, and solving the theoretical challenges.

4.1.1 Relevant work

The early works on this subject proposed a few heuristics and experimentally proved their effectiveness [35, 43]. Netrapalli et al. [63] established the first theoretical guarantees for this problem for discrete-time infections. They proved one can recover the edges of any graph with correlation decay, with access to the times of infection for multiple cascades spreading on the graph. They introduced a likelihood, proved it decouples into convex subproblems, and demonstrated that the edges of the graph can therefore be obtained efficiently. They also proved a sample complexity lower bound and showed their method is within a log factor of it. Abrahao et al. [2] also introduced a method of solving this problem, this time for a more realistic, continuous-time infection model, through learning only the first edge of each cascade. Zarezade et al. [87] proposed a first experimental attempt to tackle the case of correlated cascades using Hawkes processes. Khim et al. [47] extended the theoretical results to the case where the the cascades spreading on the graph are not independent, which required completely new machinery involving martingales and weighted Pólya urns.

All the results above assume we have perfect knowledge of the properties of the spread we use to reconstruct the graph. For most of the literature, those are the times of infection for all nodes for each cascade. This assumption may hold for online epidemics, as information is usually dated (for instance, posts or retweets on social networks have time stamps). For human networks, however, this assumption is often unrealistic: official diagnosis (and hence

recording by any tracking entity such as the CDC) may come days, weeks, or in important examples such as HIV, *years* after the actual moment of infection. Moreover, this can be highly variable from person to person, hence the infector is often diagnosed after the infectee. Similar issues arise with biological networks: we only know the expression of a gene when a measure is taken, which can happen after a typically arbitrary delay.

We therefore develop a method for learning the graph of epidemics with noisy times of infection. We demonstrate that past approaches are unusable, due to the fact that even small levels of noise are typically enough to cause order-of-diagnosis to differ from order-of-infection. We develop new techniques to deal with this setting, and prove our algorithm can reliably learn the edges of a tree in the limited-noise setting, for *any* noise distribution. We also show we can learn the structure of any bounded degree graph from a very weak observation model, in an sample-optimal fashion (up to log-factors). We finally provide an algorithm which learns the weights of any bounded-degree graph in the limited-noise setting.

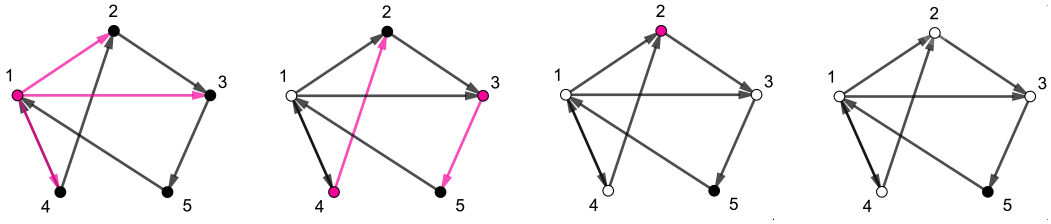
4.1.2 Model

We observe epidemics spreading on a graph, and aim to reconstruct the graph structure from noisy estimates of the times of infection. In this section, we specify the exact propagation model, the noisy observation model, and the two learning tasks this work tackles.

Propagation model: We consider a particular variant of the *inde-*

Table 4.1: Notations

$G = (V, E)$	Graph G, V set of nodes, E set of edges.
N	Number of nodes in the graph.
T_i^m	Random variable for the actual time of infection of node i during cascade m.
n_i^m	Noise of node i during cascade m.
$T_i'^m = T_i^m + n_i^m$	Random variable for the noisy time of infection of node i during cascade m.
I_i^m	Random boolean variable: $I_i'^m = True \Leftrightarrow$ node i was infected during cascade m
p_{ij}	Weight of edge (i, j), corresponding to the probability that i infects j.



(a) At t=0, node 1 is the source, in the infected state. It can possibly infect node 2, 3 and 4, all in the susceptible state.

(b) At t=1, nodes 3 and 4 are infected. Node 3 can infect susceptible node 5. Node 4 can infect susceptible node 2 but not removed node 1.

(c) At t=2, node 2 is infected. All its neighbors are in the removed state, so new node can be infected.

(d) At t=3, the cascade stops, even if node 5 remains in the susceptible state.

Figure 4.1: A complete cascade.

pendent cascade model, close to the one-step model introduced by [34] and further studied by [45]. The epidemic spreads in *discrete time* on a *weighted directed graph* $G = (V, E)$, where parents can infect their children with probability equal to the weight of the edge between them, but children cannot infect their parents. We allow bidirectional edges: it is possible that both $(i, j) \in E$ and $(j, i) \in E$, possibly with different weights. For each edge $(i, j) \in E$, the corresponding weight p_{ij} is such that $0 < p_{min} \leq p_{ij} \leq p_{max} < 1$.

This process is an instance of a $Susceptible \rightarrow Infected \rightarrow Removed$ (SIR) process. Each node starts out in the susceptible state. As in [2], each cascade m starts at a positive time [2] on a unique node source node, picked uniformly among the nodes of the graph. Once the source becomes infected, it is removed from the graph, and if it has children, each is infected at the next time step independently according to a probability specified by the weight of the edge shared with the source.

The process ends when there are no newly infected nodes (either because no infection happened during the previous time step, in which case some nodes may never be infected, or because all the nodes of the graph are removed). One realization of this process from start to finish is called a **cascade**. If two nodes are infected during the same cascade, we say that they are **co-infected** for this cascade. This process is illustrated in Figure 4.1.

[2]Most of the literature considers the initial time of infection to be 0. This is because when we have access to the exact times of infection, we can make this assumption without loss of generality. In our case, it would imply we know exactly when an outbreak started, which is usually not the case.

Observation model: Let T_i^m be a random variable corresponding to the time of infection of node i during cascade m, and let t_i^m be its realization (if i stays in the susceptible state during cascade m, we have $t_i^m = \infty$). We introduce three observation models.

In the **no-noise setting**, we have access to the exact times of infection T_i^m.

In the **limited-noise setting**, we never get to observe the exact times of infection T_i^m, but only a noisy version $T_i'^m = T_i^m + n_i^m$ (with realization $t_i'^m$), where all the n_i^m are i.i.d., and represent the noise added to the T_i^m. We assume n_i^m follows a *known* distribution $\mathcal{D}$. The only restriction we put on $\mathcal{D}$ is that it cannot have infinite value (*i.e.,* $t_i'^m = \infty \Leftrightarrow t_i^m = \infty$, and we know for a fact when nodes have been infected or not).

In the **extreme-noise setting**, we take the previous setting to the extreme, and we assume that instead of having access to the noisy times of infection $T_i'^m$, we only have access to the infection status of the nodes I_i^m. We know $I_i^m = True$ if i was infected during cascade m, and $I_i^m = False$ otherwise. Note that $T_i'^m < \infty \Leftrightarrow I_i^m = True$, so we can always deduce the infection status from the noisy times of infection. However, we cannot guess the noisy times of infection from the infection status: the (noisy) times of infections contain strictly more information than the infection status.

For these three settings, we call a **sample** the vector of all observations for the cascade m. In the no-noise setting, this is the extended-integer vec-

tor $\{t_i^m\}_{i\in V}$. In the limited-noise setting, this is the extended-integer vector $\{t_i'^m\}_{i\in V}$. In the extreme-noise setting, this is the boolean vector corresponding to the realization of $\{I_i^m\}_{i\in V}$. We also use the notation $T' = \{T_i'^m\}_{i\in V}^{m=1\dots M}$ (respectively $t' = \{t_i'^m\}_{i\in V}^{m=1\dots M}$) for the matrix representing the random variable (respectively the realizations) of all the samples.

Learning tasks: We focus on two different learning tasks. When we learn the **structure** of a graph, it means that for any two nodes i and j, we can decide whether or not there exists an edge between these two nodes (whatever its direction). When we learn the **weights** of the graph, it means that for every two nodes i and j, we learn the exact value[3] of both p_{ij} and p_{ji} up to precision ϵ.

4.1.3 Why is it a hard problem?

4.1.3.1 Counting approaches

Most approaches in the no-noise setting relate to counting. In our setting, for instance, a natural (and consistent) estimator for p_{ij} is to count how often an infection occurred along an edge, and divide it by how often such an infection could have happened:

$$\hat{p_{ij}} = \frac{\text{Number of times } j \text{ becomes infected one time step after } i}{\text{Number of times } i \text{ was infected before } j}.$$

In the no-noise setting, j could only have been infected by a node signaling exactly one time step before j. However, in the limited-noise setting,

[3]When $(i,j) \notin E$, we have $p_{ij} = 0$.

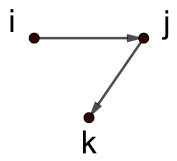
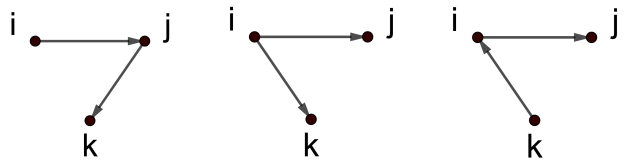


Figure 4.2: Possible scenarios which could have led to $T_i^{'} = 2$, $T_j^{'} = 3$ and $T_k^{'} = 4$. In the no-noise setting, this implies $T_i = 2, T_j = 3, T_k = 4$, and there is only one possible infection pattern.

j signaling its infection one time step after i could stem from a variety of scenarios:

- i could have indeed infected j: cases a), b) and c) of Figure **??** above.
- j could have infected i, but the noise flipped the order of signaling: cases d), e) and f).
- No infection happen between i and j, and the probability of infectin depends mainly on another node k: cases g), e) and f). This could happen for *any other node* k in the graph.

The natural estimator introduced earlier is therefore not consistent anymore; instead, it tends to a quantity which depends on p_{ij}, but also p_{ji}, and $p_{ik}, p_{ki}, p_{jk}, p_{kj}$ as well, for all the other nodes k in the graph. By counting the number of times j became infected one time step after i, we are not counting the number of infections along the edge (i, j) anymore, but instead a mixture of all the scenarios described above, which not only include the cases where j infected i, but also events in which the cascade spread through another node k, and the



(a)

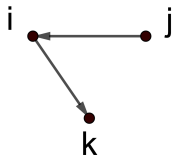
	T	n
i	0	2
j	1	2
k	2	2

(b)

	T	n
i	0	2
j	1	2
k	1	3

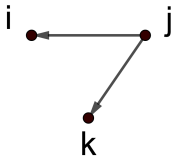
(c)

	T	n
i	1	1
j	2	1
k	0	4



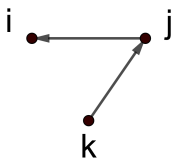
(d)

	T	n
i	1	1
j	0	3
k	2	2



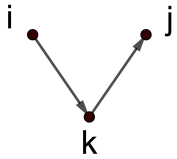
(e)

	T	n
i	1	1
j	0	3
k	1	3



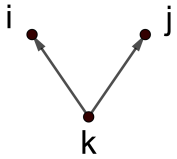
(f)

	T	n
i	2	0
j	1	2
k	0	4



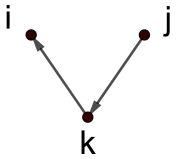
(g)

	T	n
i	0	2
j	2	1
k	1	3



(h)

	T	n
i	1	1
j	1	2
k	0	4



(i)

	T	n
i	2	0
j	0	3
k	1	3

Figure 4.3: Possible scenarios which could have led to $T_i^{'} = 2$, $T_j^{'} = 3$ and $T_k^{'} = 4$. We have $T_l^{'} = T_l + n_l$. In the limited-noise setting, there are nine possible infection patterns (many more scenarios with the same infection pattern, but different noise values, are not shown).

edge (i, j) was irrelevant to the process. Using this estimator, or any obvious (to us) extension of it, would not only imply learning the wrong weights for the edges, but also learning edges when there are no edges. Our first contribution is therefore to design a new set of estimators, from which we can deduce the value of p_{ij} (Sections 4.2.3 and 4.3.2).

Adding noise in the time of infection not only reverses the cascade chronology, it also exponentially increased the number of possible infection patterns that *could* have happened. Bounding the realm of possibilities is therefore our second step towards solving the problem (Section 4.2.1).

4.1.3.2 Max-likelihood approaches

Another common approach is to use likelihood-based methods. For instance, in [63], the authors develop a max-likelihood-based approach to learn the edges of the graph. They prove the log-likelihood has desirable properties: it decouples into only one local problem for each node, and this local problem is convex (up to the change of variable $\theta_{ij} = -\log(1 - p_{ij})$):

$$\mathcal{L}(T, P_{ij}) = \log \left(\frac{1}{N} \cdot \prod_{1 \leq i \leq N} \underbrace{\left(\prod_{j; t_j < t_i - 1} (1 - p_{ji}) \right)}_{\substack{\text{Probability that } i \text{ was not} \\ \text{infected before } t'_i.}} \cdot \underbrace{\left(1 - \prod_{j; t_j = t_i - 1} (1 - p_{ji}) \right)}_{\substack{\text{Probability that } i \text{ was} \\ \text{infected at } t'_i.}} \right)$$

$$\mathcal{L}(T, P_{ij}) = \log \left(\frac{1}{N} \right) + \sum_{i=1}^{N} \sum_{j; t_j < t_i - 1} \log(1 - p_{ji}) + \sum_{j; t'_j = t'_i - 1} \log(1 - p_{ji})$$

In our setting, the log-likelihood has none of these properties. It is not convex, and it is unclear any method other than brute force could find its maximum. Moreover, it does not decouple anymore, and even computing the log-likelihood itself takes exponential time.

$$\mathcal{L}(T, P_{ij}) = \log \left(\frac{1}{N} \cdot \underbrace{\sum_{(t_1,\dots,t_N) \leq (t'_1-1,\dots,t'_N-1)}}_{\substack{\text{This is the root} \\ \text{of the difficulty.}}} \cdot \prod_{1 \leq i \leq N} \underbrace{\left(n(t'_i - t_i) \right)}_{\substack{\text{Noise at node} \\ i \text{ is } t'_i - t_i.}} \right.$$
$$\left. \cdot \underbrace{\left(\prod_{j; t_j < t_i - 1} (1 - p_{ji}) \right)}_{\substack{\text{Probability that } i \text{ was not} \\ \text{infected before } t'_i.}} \cdot \underbrace{\left(1 - \prod_{j; t_j = t_i - 1} (1 - p_{ji}) \right)}_{\substack{\text{Probability that } i \text{ was} \\ \text{infected at } t'_i.}} \right)$$

When dealing with hidden variables, a common technique would be to use the Expectation-Maximization algorithm [18]. However, in our setting, the number of hidden states is $\prod_{i=1}^{N} t'_i$, which can be as large as $(N-1)^N$. This prohibits any realistic use of the Expectation-Maximization algorithm for networks with more than twelve nodes. Moreover, except for the recent contributions [51], very little is known about the theoretical convergence of the Expectation-Maximization algorithm.

4.1.4 Contributions

The contributions of this chapter are:

- To the best of our knowledge, we are the first to tackle the problem of learning the edges of a graph from noisy times of infection, a simple but natural extension of a well-known problem.
- We provide the first efficient algorithm for learning the structure and weights of a bidirectional tree in this setting. We also establish a tree-specific lower bound which shows that our algorithm is sample-optimal (up to log-factors) for learning the structure of the tree (Section 4.2).
- We prove it is possible to learn the structure of any bounded-degree graph in the extreme setting for which we only have access to the infection status (*i.e.,* whether or not a node was infected). Moreover, we can do so with almost optimal sample complexity, according to the bound established in [63].
- We provide polynomial algorithms for learning the weights of bounded degree graphs.
- Finally, we extend the results from bounded-degree graphs to general graphs. This proves the problem is solvable under any noise distribution, although the exponential sample complexity and running time prohibits any use of this algorithm in practice (Section 4.3.2).

4.2 Learning bidirectional trees

The bidirectional tree is the simplest example which illustrates some of the difficulties of the noisy setting. Indeed, for a directed tree, the true sequence of infections can be reconstructed, and we can use techniques from the no-noise setting. For a bidirectional tree, those techniques cannot be extended. However, the uniqueness of paths in the bidirectional tree still makes this problem considerably easier than the general setting. We therefore start by presenting a solution for the bidirectional tree. The key ideas here generalize to the neighborhood-based decomposition we introduce below, which forms our key conceptual approach for the general problem.

This section contains three contributions. First, we show how to learn the structure of a tree using only the infection status, *i.e.*, what we call the extreme-noise setting (Section 4.2.1). For each cascade, we only know which nodes were infected. We show this contains enough information to learn the structure of bidirectional tree. Second, we establish a lower bound for the no-noise setting, and show our algorithm has the same dependency in the number of nodes N as this lower bound (up to log-factors). In other words, for the task of learning the structure of any tree, an optimal algorithm in the no-noise setting would need as many cascades as our algorithm needs in the extreme-noise setting (up to log-factors).

Finally, we show how we can leverage this learned structure to learn the weights of the tree, this time when we have noisy access to the times of infection, *i.e.,* the limited-noise setting (Section 4.2.3). We provide sample

complexity for this task.

4.2.1 Tree structure

As illustrated in Section 4.1.3, the number of edges that *could* exist is much higher in the limited-noise setting than the number of actual edges in the tree. Our first key contribution is therefore to introduce a new estimator, $\hat{h}_{ij}$, which keeps track of the fraction of cascades for which i and j were both infected. This estimator can therefore be computed only with the infection status in the extreme-noise setting. Using this estimator, we show that in the specific case where the graph is a tree, we learn the structure of this tree, *i.e.* whether or not both $p_{ij} = p_{ij} = 0$.

Our algorithm for learning the edges of the tree relies on one central observation: $h_{i,j}$ achieves a kind of local maximum if there is an edge between i and j (Lemma 4.2.1). This observation relies heavily on the fact that there is uniqueness of paths on a tree. Let us now dive into the proof.

Definition 4.2.1. Let $\hat{h}_{i,j}$ be the fraction of cascades in which both i and j became infected. We have:

$$\hat{h}_{i,j} \to_{M\to\infty} \mathbb{P}(I_i^m \ \& \ I_j^m) := h_{i,j}.$$

We now show that the limit $h_{i,j}$ of the estimator $\hat{h}_{i,j}$ satisfies a local maximum property on the edges of the tree:

Lemma 4.2.1. *If i and j are not neighbors, let $(u_0, u_1, \ldots, u_d)$ be the path*

between them, with $u_0 = i$, $u_d = j$, *and* $d > 1$. *Then:*

$$\forall r \in \{0, d-1\},\ h_{i,j} < h_{u_r, ur+1}.$$

Proof. We consider the case in which both i and j have been infected. There is a unique source of infection and a unique path between i and j. Therefore, all the nodes on the path from i to j must have been infected as well. In particular, both u_r and u_{r+1} were infected. This shows $I_i^m \ \& \ I_j^m \Rightarrow I_{u_r}^m \ \& \ I_{u_{r+1}}^m$, so $\mathbb{P}(I_i^m \ \& \ I_j^m) \leq \mathbb{P}(I_{u_r}^m \ \& \ I_{u_{r+1}}^m)$, and therefore $h_{i,j} \leq h_{u_r, u_{r+1}}$.

What's more, every time u_r and u_{r+1} became infected, at least one more infection along an edge must have occurred in order for j to become infected as well. This occurred with probability at most $p_{max} < 1$. Therefore, $h_{i,j} = \mathbb{P}(I_i^m \ \& \ I_j^m) \leq p_{max} \cdot \mathbb{P}(I_{u_r}^m \ \& \ I_{u_{r+1}}^m) = p_{max} \cdot h_{u_r, u_{r+1}}$. We conclude $h_{i,j} < h_{u_r, u_{r+1}}$.

□

This simple lemma allows us to design Algorithm 1. Indeed, suppose we have access to all the limits $h_{i,j}$. By ordering them in decreasing order, we can deduce the structure of the tree by greedily adding every edge unless it forms a cycle[4].

[4]This algorithm is very similar in spirit to Kruskal's algorithm for finding the maximum spanning tree []

Algorithm 1 Learn the undirected edges of the tree.

1: **procedure** LEARNTREE($\{h_{i,j}\}_{i,j \in V}$) ▷ $h_{i,j}$ limit of $\hat{h}_{i,j}$.
2: $pairs\_h\_edge \leftarrow [(h_{i,j}, (i,j))$ for $1 \leq i \leq j \leq n_{nodes}]$
3: SORT($pairs\_h\_edge$) by decreasing order
4: $edges\_tree \leftarrow []$
5: **for** $(\sim, potential\_edge) \in pairs\_h\_edge$ **do**
6: **if** Adding $potential\_edge$ to $edges\_tree$ does not create a cycle **then**
7: Add $potential\_edge$ to $edges\_tree$
return $edges\_tree$

We show that if we have access to the limits $h_{i,j}$ of the estimators $\hat{h}_{i,j}$, the algorithm above correctly find the structure of the tree.

Lemma 4.2.2. *Algorithm 1 correctly finds all the $N-1$ pairs (i,j) such that there exists at least one directed edge between i and j.*

Proof. We show that in the for-loop at line 5, we add an edge to *edges_tree* if and only if this edge was a real edge in the original tree. We prove it by induction on the elements of the sorted list *pairs_h_edge*.

When no element has been selected, the proposition is trivially true. Suppose now that t elements of *pairs_h_edge* have been examined so far. Let $(\sim, (i,j))$ be the $t+1^{th}$ element. Two cases arise:

1. i and j are not neighbors. Let $(u_0, \ldots, u_d)$ be the path between them, with $u_0 = i$ and $u_d = j$. In this case, using Lemma 4.2.1, $\forall r$, $h_{u_r,u_{r+1}} > h_{i,j}$. In other words, all the pairs (u_r, u_{r+1}) have already been considered by the algorithm. By induction, we have kept all of them in *edges_tree*.

Therefore, adding the pair (i, j) would form a cycle. This pair is not kept in *edges_tree*, which is what we wanted since it is not an edge in the original tree.

2. i and j are neighbors. Suppose that adding this pair forms a cycle. Then there is a sequence $(v_0 = i, \ldots, v_d = j)$ of nodes such that $h_{v_k, v_{k+1}}$ were all bigger than $h_{i,j}$, and the pairs (v_k, v_{k+1}) were kept by the algorithm for all k. However, by uniqueness of paths in a tree, there exists a pair (v_a, v_{a+1}) such that the path connecting v_a and v_{a+1} in the original tree goes through (i, j). Using Lemma 4.2.1, this means $h_{i,j} > h_{v_a, v_{a+1}}$, which is a contradiction. Therefore, adding this pair in *edges_tree* does not form a cycle. This pair is kept in *edges_tree*.

Therefore, this algorithm keeps all the edges, and only the edges of the tree, so it recovers the tree structure.

□

We next quantify how many cascades M are needed for Algorithm 1 to be correct if we replace the $\{h_{i,j}\}_{i,j \in V}$ by their estimates $\{\hat{h}_{i,j}\}_{i,j \in V}$. We note that we do not require $\hat{h}_{i,j}$ to be close to their limit, but only need the order of the $\hat{h}_{i,j}$ to be the same as the order of the $h_{i,j}$. We identify events which guarantee that the order is the same (Corollary 4.2.4):

Definition 4.2.2. Let $\mathcal{H}_3 := \{(i, j, k) \in \{1, \ldots, N\}^3, p_{ij} + p_{ji} > 0 \ \& \ p_{jk} + p_{jk} > 0\}$ be the set of triplets of nodes such that at least one directed edge

exists between the first and the second node, as well as between the second and the third node.

Proposition 4.2.3. *If:*

$$\forall (i,j,k) \in \mathcal{H}_3,\ \hat{h}_{i,j} > \hat{h}_{i,k} \text{ and } \hat{h}_{j,k} > \hat{h}_{i,k},$$

then for all paths $(u_0, \ldots, u_d)$ *in the tree, with* $d > 1$*, we have:*

$$\forall r \in \{0, \ldots, d-1\},\ \hat{h}_{u_r,u_{r+1}} > \hat{h}_{u_0,u_d}.$$

Proof. For $r \in \{0, \ldots, d-2\}$, we have by hypothesis $\hat{h}_{u_r,u_{r+1}} > \hat{h}_{u_r,u_{r+2}}$. Now, we recall that $\hat{h}_{u_0,u_d}$ is the number of cascades for which both u_0 and u_d were infected. By uniqueness of paths in the tree, every time both u_0 and u_d were infected, both u_r and u_{r+2} must have been infected as well. This shows that $\hat{h}_{u_0,u_d} \leq \hat{h}_{u_r,u_{r+2}}$. Notice that this is a deterministic property, not an asymptotic property. Therefore, $\hat{h}_{u_r,u_{r+1}} > \hat{h}_{u_r,u_{r+2}} \geq \hat{h}_{u_0,u_d}$.

For $r = d-1$, we follow an identical reasoning, but with $\hat{h}_{u_r,u_{r+1}} > \hat{h}_{u_{r-1},u_{r+1}}$.

Corollary 4.2.4. *If:*

$$\forall (i,j,k) \in \mathcal{H}_3,\ \hat{h}_{i,j} > \hat{h}_{i,k} \text{ and } \hat{h}_{j,k} > \hat{h}_{i,k},$$

then the correctness of Algorithm 1 is preserved when given $\hat{h}$ *as input instead of* h*.*

In other words, Algorithm 1 outputs a correct set of undirected edges with finite samples.

Proof. According to Proposition 4.2.3, for all paths $(u_0, \ldots, u_d)$ in the tree, with $d > 1$, we have that $\forall r \in \{0, \ldots, d-1\}$, $\hat{h}_{u_r,u_{r+1}} > \hat{h}_{u_0,u_d}$. As shown in the proof of Lemma 4.2.2, this is the only property of the input needed in order to yield the correct output.

Proposition 4.2.5. *With* $M = \frac{N\left(\log\left(\frac{1}{\delta}\right)+2\log(N)\right)}{p_{min}(1-p_{max})}$ *cascades, with probability at least* $1-\delta$*, we have:*

$$\forall (i,j,k) \in \mathcal{H}_3,\ \hat{h}_{i,j} > \hat{h}_{i,k} \text{ and } \hat{h}_{j,k} > \hat{h}_{i,k}.$$

Proof. Let us consider one triplet (i,j,k) in $\mathcal{H}_3$. We recall that $\hat{h}_{i,k}$ is the number of cascades for which both i and k were infected. Since the only path from i to k is through j, we always have that $\hat{h}_{i,j} \geq \hat{h}_{i,k}$ and $\hat{h}_{j,k} \geq \hat{h}_{i,k}$. We notice that to obtain $\hat{h}_{i,j} > \hat{h}_{i,k}$, we only need one cascade for which both i and j got infected, but not k. We lower bound the probability $P_{\text{triplet identified}}$ of this cascade happening. For each cascade m, we have:

$$\begin{aligned}
P_{\text{triplet identified}} &= \mathbb{P}(I_i^m \ \&\ I_j^m \ \&\ \textsc{Not}(I_k^m)) \\
&\geq \mathbb{P}(i \text{ was a source, } i \text{ infected } j,\ j \text{ did not infect } k) \\
&\quad + \mathbb{P}(j \text{ was a source, } j \text{ infected } i,\ j \text{ did not infect } k) \\
&\geq \frac{1}{N} \cdot p_{ij} \cdot (1 - p_{jk}) + \frac{1}{N} \cdot p_{ji} \cdot (1 - p_{jk}) \\
&\geq \frac{1}{N} p_{min}(1 - p_{max}).
\end{aligned}$$

The probability that this event never occurs during the M cascades is upper

bounded by:

$$\begin{aligned}\mathbb{P}(\hat{h}_{i,j} = \hat{h}_{i,k}) &\leq \left(1 - \frac{1}{N} p_{min}(1 - p_{max})\right)^M \\ &\leq e^{-\frac{M}{N} p_{min}(1-p_{max})} \\ &\leq \frac{\delta}{N^2}.\end{aligned}$$

Now, there are $N-1$ edges in a tree, therefore $|\mathcal{H}_3| \leq (N-1)^2 < N^2$. By union bound:

$$\begin{aligned}\mathbb{P}\left(\bigcup_{(i,j,k) \in \mathcal{H}_3} \hat{h}_{i,j} = \hat{h}_{i,k}\right) &\leq \sum_{(i,j,k) \in \mathcal{H}_3)} \mathbb{P}(\hat{h}_{i,j} = \hat{h}_{i,k}) \\ &< N^2 \cdot \frac{\delta}{N^2} \\ &\leq \delta.\end{aligned}$$

Notice that $\mathcal{H}_3$ contains both (i,j,k) and (k,j,i). We have therefore proven that with probability at least $1-\delta$, when considering $M = \frac{N\left(\log\left(\frac{1}{\delta}\right) + 2\log(N)\right)}{p_{min}(1-p_{max})}$ cascades, we have $\forall (i,j,k) \in \mathcal{H}_3,\ \hat{h}_{i,j} > \hat{h}_{i,k}$ and $\hat{h}_{j,k} > \hat{h}_{i,k}$.

Putting together Proposition 4.2.5 and Corollary 4.2.4, we obtain our first theorem for learning the undirected edges with finite samples:

Theorem 4.2.6. *With $M = \frac{N\left(\log\left(\frac{1}{\delta}\right) + 2\log(N)\right)}{p_{min}(1-p_{max})}$ cascades, with probability at least $1-\delta$, we can learn the structure of a any bidirectional tree in the extreme-noise setting, i.e., when we only have access to the infection status of the nodes.*

4.2.2 Lower bound

In this section, we prove a lower bound for trees in the no-noise setting. With very minor adjustments, we adapt the lower bound of [63]. Since for a general tree, the max degree can be up to $N-1$, we design a lower bound which is independent from the max-degree. Let G be a tree drawn uniformly from $\mathcal{G}$, the set of all possible trees on N nodes, and $\hat{G}$ be the reconstructed graph from the times of infection. $G \leftrightarrow T \leftrightarrow \hat{G}$ therefore forms a Markov chain. We have:

$$\begin{aligned}
H(G) &= I(G;\hat{G}) + H(G|\hat{G}) \\
\text{(data processing inequality)} \quad &\leq I(G;T) + H(G|\hat{G}) \\
\text{(independent cascades)} \quad &\leq M \cdot I(T^1;T'^1) + H(G|\hat{G}) \\
(I(X,Y) \leq H(X)) \quad &= M \cdot \sum_{i=1}^{N} H(T_i^1) + H(G|\hat{G}) \\
\text{(Fano's inequality)} \quad &\leq M \cdot \sum_{i=1}^{N} H(T_i^1) + (1 + P_e \log(|\mathcal{G}|))
\end{aligned}$$

Since G is drawn uniformly from $\mathcal{G}$, $H(G) = \log(|\mathcal{G}|)$. There are N^{N-2} trees on N nodes, according to Cayley's formula [11], so $H(G) = (N-2) \cdot \log(N)$.

In conclusion:

$$M \geq \frac{(1-P_e)(N-2)\log(N) - 1}{N \cdot H(T_i^1)} = \Omega\left(\frac{\log(N)}{H(T_i^1)}\right)$$

Using the same kind of techniques as in [63], we can assume $H(T_i^1) \sim \frac{\log(N)}{N}$. Therefore:

Theorem 4.2.7. *In the no-noise setting, we need* $M = \Omega(N)$ *cascades to learn the tree structure.*

In our extreme-noise setting, when we have only access to the infection status of the nodes, we can learn the tree structure with the same sample complexity (up to log-factors) as the no-noise setting!

4.2.3 Tree weights

In this section, we assume we are in the limited-noise setting, and we have access to the times of infection. We also assume we have already learned the structure of the tree.

Once we have reduced the set of possible edges by learning the structure of the bidirectional tree, learning the weights of the edges is still non-trivial. Indeed, from $T_i'^m$ and $T_j'^m$, it is still impossible to know whether this sample is useful for estimating p_{ij} (case when i infected j), or whether we should use this sample for estimating p_{ji} instead (case when j infected i). What is more, we only get one sample per node and per cascade, so it is impossible to know what really happened during that cascade. However, knowing the distribution of the noise, it is possible to compute the probability that the noise maintained the order of infections. Using this information and the reduced set of known undirected edges, we can compute two sets of $N(N-1)$ estimators, from which it is possible to infer the weights of all edges in the tree.

We introduce these two sets of $N(N-1)$ estimators, or, in other words,

two estimators for each directed edge. These estimators tend to multivariate polynomials of the weights of most edges of the tree. Thus in general these polynomials have exponentially many terms; however, when i and j are neighbors, it is possible to express them concisely using a quantity $\mathcal{P}_{\setminus i}(\to i)$, which we define formally below. This succinct representation is the key idea we exploit to solve the resulting system of equations.

Once we know the structure of the bidirectional tree, we can consider the four estimators for each undirected edge (two estimators for each directed edge). They form a system of four equations and four unknowns, which we solve to obtain the weights of the edges.

Definition 4.2.3. $\mathcal{P}_{\setminus i}(\to j)$ is the probability that j became infected before any node on the path from j to i, including i, became infected.

We now introduce the estimators:

Definition 4.2.4. We introduce 2 sets of $N(N-1)$ estimators:

$\hat{f}_{i<j}$ = Fraction of infections for which i and j got infected, and i reported before j.

$\hat{g}_{i,\setminus j}$ = Fraction of infections for which i got infected, but j did not.

By the law of large numbers, as the number of cascades scales, $\hat{f}_{i<j}$ tends to $f_{i<j}$ and $\hat{g}_{i,\setminus j}$ to $g_{i,\setminus j}$, where

$$\begin{aligned} f_{i<j} &:= \mathbb{P}(T_i'^m < T_j'^m < \infty), \\ g_{i,\setminus j} &:= \mathbb{P}(T_i'^m < \infty, T_j'^m = \infty). \end{aligned}$$

We now compute the exact values of these two quantities. Let us assume the (unique) path between i and j has length d. We call $(u_0, \ldots, u_d)$ the set of nodes on the path from i to j, with $u_0 = i$ and $u_d = j$. We then have:

Lemma 4.2.8. *Recall $f_{i<j}$ and $g_{i,\backslash j}$ are the expectation of the estimators defined above. We have:*

$$\begin{aligned} f_{i<j} &= \mathcal{P}_{\backslash j}(\to i) \cdot p_{i\to j} \cdot s_{-d+1} + \mathcal{P}_{\backslash i}(\to j) \cdot p_{j\to i} \cdot s_{d+1} \\ &\quad + \sum_{l=1}^{d-1} \mathcal{P}_{\backslash i,\backslash j}(\to u_l) \cdot p_{u_l\to i} \cdot p_{u_l\to j} \cdot s_{2l-d+1}. \\ \hat{g}_{i,\backslash j} &= \mathcal{P}_{\backslash j}(\to i) \cdot (1 - p_{i\to j}) + \sum_{l=1}^{d-1} \mathcal{P}_{\backslash i,\backslash j}(\to u_l) \cdot p_{u_l\to i} \cdot (1 - p_{u_l\to j}). \end{aligned}$$

What's more, when i and j are neighbors (which implies $d = 1$), the expressions simplify to:

$$\begin{aligned} f_{i<j} &= \mathcal{P}_{\backslash j}(\to i) \cdot p_{ij} \cdot s_0 + \mathcal{P}_{\backslash i}(\to j) \cdot p_{ji} \cdot s_2 \\ g_{i,\backslash j} &= \mathcal{P}_{\backslash j}(\to i) \cdot (1 - p_{ij}). \end{aligned}$$

Proof.

$$
\begin{aligned}
f_{i<j} &= \mathbb{P}(T_i'^m < T_j'^m < \infty) \\
&= \mathbb{P}(i \text{ got infected before any nodes on the path from } i \text{ to } j, \\
&\quad \text{then infected } j, \text{ and the noise did not flip the times of infection}) \\
&\quad + \mathbb{P}(j \text{ got infected before any nodes on the path from } i \text{ to } j, \\
&\quad \text{then infected } i, \text{ and the noise flipped the times of infection}) \\
&\quad + \mathbb{P}(\text{One node on the path from } i \text{ to } j \text{ got infected before } i \text{ and } j, \\
&\quad \text{then infected both } i \text{ and } j, \text{ but } i \text{ reported before } j) \\
&= \mathcal{P}_{\not{j}}(\to i) \cdot p_{i\to j} \cdot s_{-d+1} \\
&\quad + \mathcal{P}_{\not{i}}(\to j) \cdot p_{j\to i} \cdot s_{d+1} \\
&\quad + \sum_{l=1}^{d-1} \mathcal{P}_{\not{i},\not{j}}(\to u_l) \cdot p_{u_l\to i} \cdot p_{u_l\to j} \cdot s_{2l-d+1}.
\end{aligned}
$$

This expression is involved in general. However, if i and j are neighbors, then there are no nodes u_l on the path between i and j, other than i and j themselves. What's more, $p_{i\to j} = p_{ij}$, and $d = 1$. Therefore:

$$
f_{i<j} = \mathcal{P}_{\not{j}}(\to i) \cdot p_{ij} \cdot s_0 + \mathcal{P}_{\not{i}}(\to j) \cdot p_{ji} \cdot s_2.
$$

Let us now focus on $g_{i,\not j}$:

$$\begin{aligned}
g_{i,\not j} &= \mathbb{P}(T_i'^m < \infty, T_j'^m = \infty) \\
&= \mathbb{P}(i \text{ got infected before any nodes on the path from } i \text{ to } j, \text{ but } j \text{ did not get infected}) \\
&\quad + \mathbb{P}(\text{One node on the path from } i \text{ to } j \text{ got infected before } i \text{ and } j, \text{ then infected } i \text{ but not } j) \\
&= \mathcal{P}_{\not j}(\to i) \cdot (1 - p_{i\to j}) \\
&\quad + \sum_{l=1}^{d-1} \mathcal{P}_{\not i,\not j}(\to u_l) \cdot p_{u_l \to i} \cdot (1 - p_{u_l \to j}).
\end{aligned}$$

As before, this expression is complex in general, but simplifies if i and j are neighbors, in which case:

$$g_{i,\not j} = \mathcal{P}_{\not j}(\to i) \cdot (1 - p_{ij}).$$

□

Using the simplified expression only for when i and j are neighbors, we obtain:

Proposition 4.2.9. *If we know (i, j) is an edge in the original tree, then the probability of infection along this edge is given by:*

$$p_{ij} = \frac{f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}{g_{i,\not j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}.$$

Proof. According to Lemma 4.2.8, we had four second-order equations, with 4 unknowns: p_{ij}, p_{ji}, $\mathcal{P}_{\not j}(\to i)$ and $\mathcal{P}_{\not i}(\to j)$. We solve it, and obtain the wanted result. See Appendix B.1 for details.

□

Combining all the pieces, we obtain our first theorem for infinite samples:

Theorem 4.2.10. *It is possible to learn the weights of a bidirectional tree in the limited-noise setting.*

Now that we have proven the problem is solvable, we establish the number of samples needed to learn the weights with the method above.

Lemma 4.2.11. *With* $M = \frac{N^2}{\epsilon^2} \log\left(\frac{6}{\delta}\right) \frac{\left((s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2\right)^2}{(s_0^2 - s_2^2)^2}$ *samples, with probability at least* $1 - \delta$*, we have:*

$$|\hat{p}_{ij} - p_{ij}| \leq \epsilon.$$

Proof. Using Hoeffding's inequality:

$$\mathbb{P}(|\hat{f}_{i<j} - f_{i<j}| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2},$$
$$\mathbb{P}(|\hat{f}_{j<i} - f_{j<i}| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2},$$
$$\mathbb{P}(|\hat{g}_{i,\setminus\{i,j\}} - g_{i,\setminus\{i,j\}}| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2}.$$

Choosing $M = \frac{1}{\epsilon_1^2} \log\left(\frac{6}{\delta}\right)$, we have that with probability at least $1 - \delta$, all the following hold:

$$|\hat{f}_{i<j} - f_{i<j}| \leq \epsilon_1,$$
$$|\hat{f}_{j<i} - f_{j<i}| \leq \epsilon_1,$$
$$|\hat{g}_{i,\setminus\{i,j\}} - g_{i,\setminus\{i,j\}}| \leq \epsilon_1.$$

Hence, with probability at least $1-\delta$, we have (see Appendix B.1 for details):

$$\hat{p}_{ij} - p_{ij} \leq \epsilon_1 \frac{(s_0^2 - s_2^2 + s_0 + s_2)p_{max}}{g_{i,j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2} + \epsilon_1 \frac{s_0 + s_2}{g_{i,j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2} + o(\epsilon_1).$$

We use the results from Lemma 4.2.8 to bound the denominator by $\frac{s_0^2 - s_2^2}{N}$. In the end, we obtain:

$$|\hat{p}_{ij} - p_{ij}| \leq \epsilon_1 N \frac{(s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2}{s_0^2 - s_2^2} + o(\epsilon_1).$$

We choose $\epsilon_1 = \frac{\epsilon}{N} \frac{s_0^2 - s_2^2}{(s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2}$. Therefore:
With $M = \frac{N^2}{\epsilon^2} \log\left(\frac{6}{\delta}\right) \frac{\left((s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2\right)^2}{(s_0^2 - s_2^2)^2}$ samples, with probability at least $1-\delta$, we have $|\hat{p}_{ij} - p_{ij}| \leq \epsilon$.

□

By a union bound on all the weights of the tree, knowing there are at most $2N(N-1) < 2N^2$ directed edges in a directed tree, we obtain the following sample complexity:

Theorem 4.2.12. *With $M = \frac{N^2}{\epsilon^2} \log\left(\frac{12N^2}{\delta}\right) \frac{\left((s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2\right)^2}{(s_0^2 - s_2^2)^2}$ cascades, with probability $1-\delta$, we can learn all the weights of the edges of a bidirectional tree in the limited-noise setting, i.e., when we only have access to the noisy times of infection.*

4.3 Bounded-degree graphs

In the previous section, the algorithm presented relies heavily on the uniqueness of paths. This property implies that we can deduce the edges from

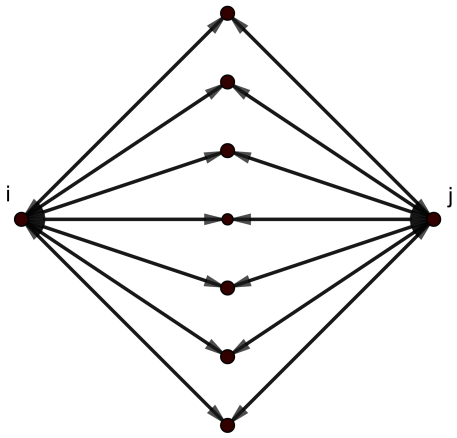


Figure 4.4: Two nodes can be co-infected frequently without sharing an edge.

the nodes which are co-infected the most often. However, this is not true for a general bounded-degree graph. In Figure 4.4, we can see that the two nodes i and j would be co-infected frequently despite not sharing an edge. This makes the task of finding the structure much more challenging than for the bidirectional tree.

In this section, we show how the main ideas for learning the structure of the bidirectional tree can be extended for learning the structure of general bounded-degree graphs, in the extreme-noise setting, with almost optimal sample complexity. The framework for learning the weights of the edges in the limited-noise setting is - to the best of our knowledge - not extendable to general bounded-degree graphs; we therefore develop a new algorithm to learn the weights for general bounded-degree graphs.

4.3.1 Bounded-degree structure

In the previous section, we introduced the estimator $\hat{h}_{i,j}$, which records the fraction of cascades for which both i and j are infected. From a local maximum property of this estimator, we deduced the structure of the tree, in a sample efficient fashion. Indeed, if there exists a path between i and k, and the first edge on this path is (i, j), then if i and k are infected, j must have been infected as well.

We want to build on this idea for a bounded-degree graph of maximum degree d. However, for such a graph, there may be multiple paths leading from i to j, and we cannot guarantee a single node will be infected each time. However, if i is a node, $\mathcal{N}_i$ is its neighborhood, and $k \notin \mathcal{N}_i$ is another node of the graph, we can guarantee that if both i and k are infected, there exists a node in $\mathcal{N}_i$ which is infected. Moreover, $\mathcal{N}_i$ is the set of smallest size for which at least one node is infected at the same time as i the most frequently. This leads us to a new set of estimators:

Definition 4.3.1. Let i be a node of the graph, and let S be a set such that $|S| < d$ and $i \notin S$. We define a new set of estimators:

$\hat{h}_{i,S}$ = Fraction of cascades for which i is infected, and at least one node of S is infected.

Let us assume that for each pair (i, S), we have access to the limit $h_{i,S}$ of $\hat{h}_{i,S}$. We now introduce an algorithm showing how to leverage these limits to learn the structure of any bounded-degree graph.

Algorithm 2 Learn the undirected edges of any graph of maximum degree d.

procedure LEARNGRAPH($\{h_{i,S}\}_{i\in V}^{|S|\leq d}$)
    $edges \leftarrow []$
    for $i = 1 \dots n_{nodes}$ **do**
        $S\_max\_i \leftarrow$ set such that h_{i,S_max_i} is maximal, and such that SIZE(S_max_i) is minimal.
        for n_j in S_max_i **do**
            Add edge (i, n_j) to $edges$
    return $edges$

We show this algorithm is correct.

Lemma 4.3.1. *Algorithm 3 correctly finds the neighborhood of each node.*

Proof. Let us recall that $h_{i,S}$ is the probability that node i and at least one node of S are co-infected. To prove the correctness of this algorithm, it suffices to prove:

$$\forall i \in V, \forall S \ s.t. \ |S| \leq d, \ \ h_{i,S} \geq h_{i,\mathcal{N}_i} \implies \mathcal{N}_i \subseteq S$$

Let pick a set S such that $\mathcal{N}_i \setminus S \neq \emptyset$, and let k be a node in $\mathcal{N}_i \setminus S$. We know $h_{i,S\cup\{k\}} \geq h_{i,S} + \mathbb{P}(i \text{ and } k \text{ are the only infected nodes})$. Since i and k are neighbors, $\mathbb{P}(i \text{ and } k \text{ are the only infected nodes}) > 0$, and therefore $h_{i,S\cup\{k\}} > h_{i,S}$. Following this line of reasoning, if $\mathcal{N}_i \setminus S \neq \emptyset$, S we can always increase the value of $h_{i,S}$ by adding a node of $\mathcal{N}_i$.

However, it is impossible to increase the value of $h_{i,\mathcal{N}_i}$, because if i and any other node of the graph are co-infected, we know one node of $\mathcal{N}_i$ is also

infected. Therefore, the algorithm is correct.

□

Unfortunately, we do not have direct access to $h_{i,S}$. We therefore study how many samples are needed to replace $h_{i,S}$ by its estimate $\hat{h}_{i,S}$ while preserving the correctness of the algorithm. Just like in Section 4.2.1, we notice that we do not need the $\{\hat{h}_{i,S}\}_{i\in V}^{|S|\leq d}$ to be close to their limit, we only need the indexes of their ordering to be the same. We notice that the neighborhood $\mathcal{N}_i$ of i is the set of smallest size which is the most often infected at the same time as i is ($\mathcal{N}_i = \arg\max_{|R|\leq d} h_{i,R}$). Therefore, we must have that for every set S, $\hat{h}_{i,S} \leq \hat{h}_{i,\mathcal{N}_i}$. However, some other sets might achieve the same value if we do not observe enough cascades. This could happen in two cases:

- Not all the nodes of the neighborhood were infected, and therefore a subset $T_1 \subset \mathcal{N}_i$ of the neighborhood is such that $\hat{h}_{i,T_1} = \hat{h}_{i,\mathcal{N}_i}$. Since $|T_1| < |\mathcal{N}_i|$, the algorithm would return T_1 and not $\mathcal{N}_i$, which would be a failure case.

- Some other node k is always infected every time a specific node $j \in \mathcal{N}_i$ is infected. The set $T_2 = \mathcal{N}_i \setminus \{j\} \cup \{k\}$ will therefore be such that $\hat{h}_{i,T_2} = \hat{h}_{i,\mathcal{N}_i}$, and the algorithm would not know which maximum to pick. This is a failure case as well.

We identify events which guarantee that the failure cases above cannot arise. Let $E_{i,j}^m$ be the event that i and j were the only infected nodes during cascade

m. $E_{i,j} = \bigcup_{1 \leq m \leq M} E_{i,j}^m$ is therefore the event that there exists a cascade for which only i and j were infected. If such a cascade exists, the set $S_{max} = \arg\max_{|R| \leq d} \hat{h}_{i,R}$ *must* contain j. If the event $E_{i,j}$ happens for every node j in the neighborhood of i, we can therefore guarantee Algorithm 3 is correct. We characterize the sample complexity needed for this below.

Proposition 4.3.2. *Let i be a node, and let j be one of its neighbors. Let S be a set of size $|S| \leq d$, such that $i \notin S$ and $j \notin S$. With probability at least $1 - \frac{\delta}{d \cdot N^{d+2}}$, among $M = \frac{(d+2) \cdot N \log(N) + N \log\left(\frac{d}{\delta}\right)}{p_{min}(1-p_{max})^{2(d-1)}}$ cascades, there exists a cascade in which i and j are infected, but no nodes of S are infected.*

Proof. We notice that if, during cascade m, the only infected nodes are i and j, no nodes of S are infected. This is exactly the event $E_{i,j}^m$ defined above.

$$\begin{aligned}
\mathbb{P}(E_{i,j}^m) &= \mathbb{P}(i \text{ and } j \text{ infected and } \forall s \in S, \text{s is not infected}) \\
&= \mathbb{P}(i \text{ and } j \text{ are the only infected nodes during cascade } m) \\
&\geq \mathbb{P}(\text{cascade started at } i,\ j \text{ was the only infected neighbor of } i, \\
&\quad \text{and } j \text{ did not infect any nodes}) \\
&\geq \frac{1}{N} \cdot p_{min} \cdot (1 - p_{max})^{2(d-1)}.
\end{aligned}$$

The probability that this never happens during M cascade is exactly the com-

plement of the event $E_{i,j}$ defined above:

$$\begin{aligned}\mathbb{P}(\textsc{Not}(E_{i,j})) &\leq \left(1 - \frac{1}{N} \cdot p_{min} \cdot (1 - p_{max})^{2(d-1)}\right)^M \\ &\leq e^{-M \cdot \frac{p_{min}}{N} \cdot (1-p_{max})^{2(d-1)}} \\ &\leq \frac{\delta}{d \cdot N^{d+2}}.\end{aligned}$$

□

Lemma 4.3.3. *With probability at least* $1-\delta$*, if we observe* $M = \frac{(d+2)\cdot N\log(N)+N\log\left(\frac{d}{\delta}\right)}{p_{min}(1-p_{max})^{2(d-1)}}$ *cascades, Algorithm 3 is correct, and has running time* $\mathcal{O}(M \cdot N^{d+2}) = \mathcal{O}(d \cdot N^{d+3}\log(N))$.

Proof. Let $\mathcal{S} = \{S \in \mathcal{P}(V), |S| \leq d\}$ be the set of sets of nodes of size at most d, let $\mathcal{S}_{\setminus i} = \{S \in \mathcal{S}, i \notin S\}$ be the set of sets of $\mathcal{S}$ which do not contain i, and let $\mathcal{N}_i = \{j, (i,j) \in E\}$ be the neighborhood of i. We pick a set $S \in \mathcal{S}$, such that $S \neq \mathcal{N}_i$.

We first prove that $\hat{h}_{i,S} \leq \hat{h}_{i,\mathcal{N}_i}$. Since the neighborhood of i separates i from the rest of the graph, and infected nodes are connected, we can conclude that every time i and another node of S are infected, one node in the neighborhood $\mathcal{N}_i$ of i is infected as well. Therefore, we cannot increase $\hat{h}_{i,S}$ without also increasing $\hat{h}_{i,\mathcal{N}_i}$. In particular, this means that even if $|S| > |\mathcal{N}_i|$ (for instance if $\mathcal{N}_i \subset S$), we have $\hat{h}_{i,S} \leq \hat{h}_{i,\mathcal{N}_i}$. We therefore know that $\mathcal{N}_i \in \arg\max_{R \in \mathcal{S}_{\setminus i}} \hat{h}_{i,R}$.

We now prove that with probability at least $1 - \delta$, $\mathcal{N}_i$ is the set of $\arg\max_{R \in \mathcal{S}_{\setminus i}} \hat{h}_{i,R}$ of minimal size. To do so, we notice that if there exists a cascade

such that i and $j \in \mathcal{N}_i$ are infected, but no node of S is infected, then this implies $\hat{h}_{i,S} < \hat{h}_{i,\mathcal{N}_i}$. Indeed, as shown above, every time we increase $\hat{h}_{i,S}$, we also increase $\hat{h}_{i,\mathcal{N}_i}$, and we know there exists one cascade for which we increased $\hat{h}_{i,\mathcal{N}_i}$ without increasing $\hat{h}_{i,S}$. We now calculate the probability P_{failure} that such a cascade does not exist for all nodes i in the graph, all nodes j in their neighborhood, and all sets S which do not include i or j.

$$\begin{aligned} P_{\text{failure}} &\leq \mathbb{P}(\exists i \in V, \exists j \in \mathcal{N}_i, \exists S \in \mathcal{S}_{\setminus i,j}, \text{every time } i \text{ and } j \text{ infected,} \\ &\qquad \text{a node of } S \text{ is also infected}) \\ &\leq \sum_{i \in V} \sum_{j \in \mathcal{N}_i} \sum_{S \in \mathcal{S}_{\setminus i,j}} \mathbb{P}(\text{every time } i \text{ and } j \text{ infected, a node of } S \text{ is also infected}). \end{aligned}$$

We use Proposition 4.3.2 to bound this quantity:

$$\begin{aligned} P_{\text{failure}} &\leq \sum_{i \in V} \sum_{j \in \mathcal{N}_i} \sum_{S \in \mathcal{S}_{\setminus i,j}} \frac{\delta}{d \cdot N^{d+2}} \\ &\leq N \cdot d \cdot N^{d+1} \cdot \frac{\delta}{d \cdot N^{d+2}} \\ &\leq \delta. \end{aligned}$$

We now know that with probability at least $1 - \delta$:

$$\forall R \in \bigcup_{j \in \mathcal{N}_i} \mathcal{S}_{\setminus i,j}, \quad \hat{h}_{i,R} < \hat{h}_{i,\mathcal{N}_i}.$$

This implies that no set of $\bigcup_{j \in \mathcal{N}_i} \mathcal{S}_{\setminus i,j}$ can belong in $\arg\max_{R \in \mathcal{S}_{\setminus i}} \hat{h}_{i,R}$. However, we have:

$$\mathcal{S}_{\setminus i} \setminus \bigcup_{j \in \mathcal{N}_i} \mathcal{S}_{\setminus i,j} = \{S \in \mathcal{S}, \mathcal{N}_i \subseteq S\}.$$

In particular, this means that $\mathcal{N}_i$ is the only set of $\mathcal{S}_{i} \setminus \bigcup_{j \in \mathcal{N}_i} \mathcal{S}_{i,j}$ of minimal size. This shows that with probability at least $1-\delta$, $\mathcal{N}_i$ is the set of $\arg\max_{R \in \mathcal{S}_i} \hat{h}_{i,R}$ of minimal size. This proves Algorithm 3 is correct, and that we can learn the structure of any graph of maximum degree d with $M = \frac{(d+2)\cdot N\log(N)+N\log\left(\frac{d}{\delta}\right)}{p_{min}(1-p_{max})^{2(d-1)}}$ cascades.

Since we do at most one operation by pair of (node, set) and by cascade, the running time is $\mathcal{O}(N \cdot N^{d+1} \cdot M)$, which is what we wanted to prove.

□

This leads to our theorem for learning the structure of any bounded-degree graph.

Theorem 4.3.4. *With probability at least $1-\delta$, in the extreme-noise setting, we can learn the structure of any graph of maximum degree d with $M = \mathcal{O}\left(\frac{d\cdot N\log\left(\frac{N}{\delta}\right)}{p_{min}(1-p_{max})^{2d}}\right)$ cascades in polynomial time.*

Let us now assume that $p_{max} \sim \frac{1}{d}$. This assumption is reasonable when you expect a constant number of infections by time step. For instance, it makes sense for real diseases, for which carriers have to meet to transmit it (we can only meet a constant number of people each day). It would not make sense for social networks, in which it is possible to reach many followers with each post.

Corollary 4.3.5. *With probability at least $1-\delta$, in the extreme-noise setting, if we assume $p_{max} \sim \frac{1}{d}$ and p_{min} constant, we can learn the structure of any graph*

of maximum degree d with $M = \mathcal{O}\left(d \cdot N \log\left(\frac{N}{\delta}\right)\right)$. This sample complexity is optimal up to log-factors, and almost matches the lower bound established in the no-noise setting.

Proof. We need at least $\mathcal{O}\left(d \cdot N \log\left(N\right)\right)$ samples to learn the structure of a bounded-degree graph with maximum degree d, according to the lower bound in [63].

4.3.2 Bounded-degree weights

For the remainder of this chapter, we state results in the limited-noise setting.

If we consider cascades of size k, the exact probability of infection between two nodes is a multivariate polynomial of degree N on $N(N-1)$ variables (the variables here would be the weights of the graph), with a sum of up to $2 \cdot \sum_{l=1}^{k} \frac{(N-2)!}{(N-k)!}$ terms. If the graph has more than five nodes, the resulting polynomial is of degree more than five.

For our algorithm, we therefore only use cascades of size 1 or 2. This is a waste of the data, since we simply discard cascades of larger size. However, we are not aware of techniques on how to utilize larger cascades in the limited-noise setting. Cascades of size 1 or 2 are simple enough that we can write explicitly their probability. This allows use to:

1. Design estimators for which we can calculate the exact limit.

2. Combine these estimators to transform a polynomial system of degree N to a polynomial system of degree 2.

3. Solve this system exactly and obtain the probabilities of infection.

4.3.3 Estimators

We start by designing a few estimators:

Definition 4.3.2. We introduce two sets of $N(N-1)$ estimators and one set of N estimators. These estimators can be computed even if we only have access to the noisy times of infection.

$\hat{h}^2_{i,j}$ = Fraction of cascades for which only i and j are infected.

$\hat{f}^2_{i<j}$ = Fraction of cascades for which only i and j are infected, and $t'_i < t'_j$.

$\hat{e}^1_i$ = Fraction of cascades for which only i is infected.

To simplify the coming notations, let us introduce s_k, which is the probability that the noise on j has delay at least k relative to the noise on i. Since the noise is i.i.d., this value is independent from i and j: $s_k = \mathbb{P}(n_j - n_i \geq k)$[5]. For instance, if i infected j during cascade m, the probability that the noise did not flip the order of infection (*i.e.* $T'^m_i < T^m_j$) is $\mathbb{P}(n_j \geq n_i) = s_0$. In

[5]For instance, for geometric noise of parameter q, we have: $s_k = \sum_{t_j=\max(0,k)}^{\infty} \sum_{t_i=0}^{t_j-k} (1-q)^{t_i+t_j} = (1-q)^{\max(0,k)} \left(1 - \frac{(1-q)^{1-\min(0,k)}}{2-q}\right)$.

the reverse case, the probability that the noise flipped the order of infection is $\mathbb{P}(T_j^m + n_j < T_i'^m + n_i) = \mathbb{P}(1 + n_j < n_i) = \mathbb{P}(n_i - n_j \geq 2) = s_2$.

We now compute the limits of those estimators.

Proposition 4.3.6. *As the number of cascades M goes to infinity, the estimators introduced above tend to the following limit:*

$$\hat{h}^2_{i,j} \to_{M\to\infty} \frac{1}{N}(p_{ij} + p_{ji}) \prod_{k\neq i,j} (1 - p_{ik})(1 - p_{jk}),$$
$$\hat{f}^2_{i<j} \to_{M\to\infty} \frac{1}{N}(p_{ij} \cdot s_0 + p_{ji} \cdot s_2) \prod_{k\neq i,j} (1 - p_{ik})(1 - p_{jk}),$$
$$\hat{e}^1_i \to_{M\to\infty} \frac{1}{N}(1 - p_{ij}) \prod_{k\neq i,j} (1 - p_{ik}).$$

Proof. The proof is very similar to 4.2.8. See details in Appendix B.2.2.

□

4.3.4 Solving the system

The limit of these estimators, as seen as a function of the probabilities of infection, is a complex polynomial on up to $2(N-1)$ variables. The crux of our algorithm is to combine those estimators in order to cancel out most of these variables, and create $N-1$ systems of two equations of degree 2 and two unknowns, which we then solve.

Proposition 4.3.7. *Let $\hat{V}_{ij} = \frac{\hat{f}^2_{i<j}}{\hat{h}^2_{i,j} + N \cdot \hat{e}^1_i \cdot \hat{e}^1_j}$. Then the limit of $\hat{V}_{ij}$ as the number of cascades M goes to infinity only depends on the variables p_{ij} and p_{ji} :*

$$\hat{V}_{ij} \to_{M\to\infty} \frac{p_{ij} \cdot s_0 + p_{ji} \cdot s_2}{1 + p_{ij} \cdot p_{ji}}.$$

Proof. Since all the estimators converge towards a constant, we can use Slutsky's lemma to find the limit of $\hat{V}_{ij}$. Let V_{ij} be the limit of $\hat{V}_{ij}$ as M goes to infinity. We notice that the $\prod_{k\neq i,j}(1-p_{ik})(1-p_{jk})$ parts cancel each other out:

$$\begin{aligned} V_{ij} &= \frac{f^2_{i<j}}{h^2_{i,j} + N \cdot e^1_i \cdot e^1_j} \\ &= \frac{\frac{1}{N}(p_{ij}\cdot s_0 + p_{ji}\cdot s_2)\left[\prod_{k\neq i,j}(1-p_{ik})(1-p_{jk})\right]}{\left(\frac{1}{N}(p_{ij}+p_{ji}) + N\frac{(1-p_{ij})(1-p_{ji})}{N^2}\right)\left[\prod_{k\neq i,j}(1-p_{jk})(1-p_{jk})\right]} \\ &= \frac{(p_{ij}\cdot s_0 + p_{ji}\cdot s_2)}{1+p_{ij}\cdot p_{ji}}. \end{aligned}$$

□

We can therefore use this equality to deduce the weights of all the edges of the graph:

Theorem 4.3.8. *For any graph, for any noise distribution having finite values, we can learn the weights of all the edges of the graph. In particular, we can compute a quantity which converges to the true weight of each edge:*

$$\hat{p}_{ij} = \frac{2(\hat{V}_{ji}s_2 - \hat{V}_{ij}s_0)}{(s_0^2 - s_2^2) + \sqrt{(s_0^2-s_2^2)^2 - 4(\hat{V}_{ji}s_2 - \hat{V}_{ij}s_0)(\hat{V}_{ij}s_2 - \hat{V}_{ji}s_0)}}.$$

Proof. We present a sketch of the proof here. The details can be found in Appendix B.2.1. We know $\hat{V}_{ij}$ tends to $V_{ij} = \frac{p_{ij}\cdot s_0 + p_{ji}\cdot s_2}{1+p_{ij}\cdot p_{ji}}$. Using both V_{ij} and V_{ji}, we can establish this second-degree equation:

$$V_{ji}s_2 - V_{ij}s_0 + \left(s_0^2 - s_2^2\right)p_{ij} + \left(V_{ij}s_2 - V_{ji}s_0\right)p_{ij}^2 = 0$$

We recall that by definition, $s_0 \geq s_2$. We also notice that if $p_{ij} = q_1$ and $p_{ji} = q_2$ is a pair of solutions of this system, then $p_{ij} = \frac{1}{q_2}$ and $p_{ji} = \frac{1}{q_1}$ forms the other pair of solution, which implies there is uniqueness of solutions in $[0, p_{max}]$. Since the real probabilities of infection satisfy this system, we also know the solution exists. Let $\Delta = (s_0^2 - s_2^2)^2 - 4(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)$. The only solution of this system in $[0, p_{max}]$ is then:

$$p_{ij} = \frac{2(V_{ji}s_2 - V_{ij}s_0)}{(s_0^2 - s_2^2) + \sqrt{\Delta}}.$$

□

4.3.5 Sample complexity

We establish the sample complexity needed to estimate p_{ij} with precision ϵ. To do so, we start by estimating V_{ij}. Note that we only consider the pair of nodes i and j if, among the M samples, there exists a cascade of size 2 in which i and j are the only infected nodes (*i.e.* $h_{i,j}^2 > 0$). Otherwise, we set $\hat{p}_{ij} = \hat{p}_{ji} = 0$ as our estimate for p_{ij}.

Proposition 4.3.9. *With probability* $1 - \frac{\delta}{N^2}$, *with* $M =$ *samples, we can estimate* V_{ij} *with precision* ϵ_V.

Proof. We present a sketch of the proof; the details can be found in Appendix

B.2.2. As in Proposition 4.2.11, we use Hoeffding's inequality:

$$\mathbb{P}(|\hat{f}^2_{i<j} - f^2_{i<j}| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2},$$
$$\mathbb{P}(|\hat{h}^2_{i,j} - h^2_{i,j}| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2},$$
$$\mathbb{P}(|\hat{e}^1_i - e^1_i| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2},$$
$$\mathbb{P}(|\hat{e}^1_j - e^1_j| > \epsilon_1) \leq 2e^{-2M\epsilon_1^2}.$$

We use this to bound above V_{ij}:

$$\hat{V}_{ij} \leq V_{ij}\left[1 + \frac{\epsilon_1}{f^2_{i<j}} + \epsilon_1 \frac{1 + N(e^1_i + e^1_j)}{h^2_{i,j} + Ne^1_i e^1_j} + o(\epsilon_1)\right]$$

By bounding below the denominators and bounding above the numerator, we finally obtain:

$$|\hat{V}_{ij} - V_{ij}| \leq \epsilon_1 \frac{4N}{p_{min}s_2(1-p_{max})^{2d}} + o(\epsilon_1).$$

Therefore, by union bound, and by choosing $\epsilon_1 \frac{4N}{p_{min}s_2(1-p_{max})^{2d}} = \epsilon_V$, and setting $2e^{-2M\epsilon_1^2} = \frac{\delta}{3N^2}$, we obtain:
With $M = \frac{1}{\epsilon_V^2}\frac{16N^2}{p^2_{min}s^2_2(1-p_{max})^{4d}}\frac{2\log(3N)-\log(\delta)}{2}$ samples, we can guarantee $|\hat{V}_{ij} - V_{ij}| \leq \epsilon_V$ with probability at least $1 - \frac{\delta}{N^2}$.

□

Once we have estimated V_{ij} with precision ϵ_V, estimating p_{ij} is unfortunately still not an easy task. Indeed, let $\Delta = (s_0^2 - s_2^2)^2 - 4(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)$. We have $p_{ij} = \frac{-(s_0^2 - s_2^2) + \sqrt{\Delta}}{2(V_{ij}s_2 - V_{ji}s_0)}$, which means that Δ has to be positive for this quantity to be defined. However, for general values of V_{ij} and V_{ji}, Δ can be negative. We therefore use the framework of constrained optimization to bound Δ away from 0.

Lemma 4.3.10. *Let* $\Delta = (s_0^2 - s_2^2)^2 - 4(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)$. *We have:*

$$\Delta \geq (s_0^2 - s_2^2)^2 \frac{(1 - p_{max})^2}{1 + p_{max}^2}.$$

Proof. We find the lower bound on Δ by reformulating the problem as a constrained optimization problem, and introducing the corresponding Lagrangian multipliers. The details can be found in Appendix B.2.1.

□

Now that we have established this bound on Δ, we can give the sample complexity needed to estimate p_{ij}.

Proposition 4.3.11. *Assuming we can estimate* V_{ij} *within precision* ϵ_V, *then we can estimate* p_{ij} *within precision* $\epsilon = \frac{6\epsilon_V(1+p_{max}^2)}{(s_0^2 - s_2^2)^2(1-p_{max})^2}$.

Proof. This is a simple derivation which can be found in Appendix B.2.2.

□

We finally piece everything together, and use a union bound on all the p_{ij} to obtain the final sample complexity of our algorithm for learning the weights of general bounded-degree graphs.

Theorem 4.3.12. *In the limited-noise setting, with probability at least* $1 - \delta$, *with* $M = \mathcal{O}\left(\frac{e^{4p_{max}(d+1)}}{p_{min}^2 s_2^2 (s_0^2 - s_2^2)^4} \frac{N^2}{\epsilon^2} \log\left(\frac{N}{\delta}\right)\right)$ *samples, we can learn the weights of any bounded-degree graph up to precision epsilon.*

Proof. We obtain the desired bound by combining Proposition 4.3.9 and Proposition 4.3.11. See details in Appendix B.2.2.

□

Once again, if we assume $p_{max} \sim \frac{1}{d}$ and p_{min} constant, we obtain the following sample complexity:

Corollary 4.3.13. *In the limited-noise setting, with probability at least* $1-\delta$*, if* $p_{max} \sim \frac{1}{d}$ *and* p_{min} *constant, we can learn the weights of any bounded-degree graph up to precision epsilon with* $M = \mathcal{O}\left(\frac{1}{s_2^2(s_0^2-s_2^2)^4}\frac{N^2}{\epsilon^2}\log\left(\frac{N}{\delta}\right)\right)$ *samples.*

4.4 General graphs

In the limited-noise setting, we notice that nothing prevents us from using the algorithm for learning bounded-degree weights for general graph. This proves this problem is solvable for *any* graph, and *any* noise distribution. As explained in Section 4.1.3, this is not an obvious result. However, if we do not assume $p_{max} \sim \frac{1}{N}$, the sample complexity is now exponential.

Theorem 4.4.1. *In the limited-noise setting, with probability at least* $1-\delta$*, it is possible to learn all the weights of any graph, for any noise distribution, with finite (but potentially exponential) sample complexity.*

4.5 Discussion

In this chapter, we presented the first results to learn the edges of a graph from noisy times of infection. We showed we learn the structure of any

bidirectional tree or any bounded-degree graph (note that not all trees are bounded-degree graphs) with optimal sample complexity (up to log-factors).

However, for learning the weights of a general bounded-degree graph, we only use cascades of size 1 or 2. If we are given an infinite number of cascades of size bigger than 2, our current algorithm cannot learn the weights of the graph. Future work could develop an algorithm without such a weakness.

All our results for learning the weights of the edges are in the limited-noise setting. Whether or not it is possible to learn the noise in the extreme-noise setting is an other question of interest for future work.

Finally, we have made no restriction on the distribution of the noise we add, other than it is finite. It would be interesting to study whether stronger restrictions on the noise (for instance Gaussian noise) would lead to stronger results. It would also be interesting to allow infinite noise, and develop algorithms which are robust to errors in the infection status of a node (our current algorithms can return wrong graph structure with only one adversarially chosen false positive).

Chapter 5

Uncertainty about what is infecting nodes

5.1 Introduction

In this chapter[1], we follow the research direction introduced in the previous chapter and continue to study graph inference. This time we introduce uncertainty about what is infecting the nodes. Specifically, we assume that there exist two spreading mechanisms which produce similar symptoms. Theoretically, we model this as epidemics spreading on a mixture of two different graphs (induced by the spreading mechanisms), and we aim to reconstruct both graphs from epidemic cascades.

It turns out that this problem is not always identifiable. Our first contribution is therefore to establish the first *necessary and sufficient* conditions for this problem to be solvable in polynomial time on edge-separated graphs. When the conditions are met, *i.e.*, when the graphs are connected with at least three edges, we provide an efficient algorithm for learning the weights of both graphs with optimal sample complexity (up to log factors). We give complementary results and provide sample-optimal (up to log factors) algorithms for

[1]This chapter covers the material prebiously published in *Learning Mixtures of Graphs from Epidemic Cascades*. My main contributions were defining the setting, and solving part of theoretical challenges.

mixtures of *directed* graphs of out-degree at least three, and for mixtures of undirected graphs of unbalanced and/or unknown priors.

As described in the previous chapter, learning the graph from times of infection during multiple epidemics has been extensively studied. However, this line of research always assumes that the epidemic cascades are all of the same kind, and spread on one unique graph which entirely captures the dynamics of the spread. In reality, our observations of cascades are far more granular: different kinds of epidemics spread on the same nodes but through different mechanisms, i.e., different spreading graphs. Epidemic cascades we observe are often a *mixture* of different kinds of epidemics. Without knowledge of the *label* of the epidemic, can we recover the individual spreading graphs? For a concrete example, let us consider the ubiquitous Twitter graph. Individuals usually have multiple interests, and will share tweets differently according to the underlying topics of the tweets. For instance, two users may have aligned views on football and diametrically opposed political views, and hence may retweet each others' football tweets but not political posts. Interesting settings are those where the epidemic labels (in this simple case, *football* and *politics*) is not observable. While *football* and *politics* may be easy to distinguish via basic NLP, the majority of settings will not enjoy this property (e.g., she retweets football posts relating to certain teams, outcomes, or special plays). In fact, the focus on recovering the spreading graph stems precisely from the desire to study very poorly-understood epidemics whose spreading mechanisms, symptoms, etc. remain elusive. Examples outside the Twitter realm (*e.g.*, human

epidemics with multiple spreading vectors) abound.

In such cases, applying existing techniques for estimating the spreading graph would recover the union of graphs in the mixture. For Twitter and other social networks, this is essentially already available. However, this union is typically not informative enough to predict the spread of tweets, and may even be misleading.

We address precisely this problem. We consider a mixture of epidemics that spread on two unknown weighted graphs when, for each cascade, the kind of epidemic (and hence the spreading graph) remains hidden. We aim to accurately recover the weights of both the graphs from such cascades.

5.1.1 Relevant work

Mixture models in general have attracted significant focus. Even for the most basic models, e.g., Gaussian mixture models, or mixed regression, rigorous recovery results have proved elusive, and only recently has there been significant progress (e.g., [7, 12, 16, 19, 50, 52, 83, 85, 86]). This work reveals some similarities to prior work. For example, here too, moment-based approaches play a critical role; moreover, here too, there are conditions on separation of the two classes needed for recovery. Interestingly, however, the technical key to our work is much more combinatorial in nature, rather than appealing to more general-purpose tools (like tensor decomposition or EM). As we outline below, the crux of the proof of correctness of our algorithm is a combination of a characterization of *forbidden graphs* that cannot be learned, and a

decomposition-reduction of a general graph to smaller subgraphs that can be learned and later patched to produce a globally consistent solution.

5.1.2 Contributions

To the best of our knowledge, this is the first chapter to study the inverse problem of learning mixtures of weighted undirected graphs from epidemic cascades. We address the following questions:

Recovery: Under the assumption that the underlying graphs are connected, have at least three edges and under some separability condition (detailed in the next section), we prove the problem is solvable and give an efficient algorithm to recover the weights of *any* mixture of connected graphs with equal priors on the same set of vertices.

Identifiability: We show the problem is not solvable in polynomial time of one if the condition mentioned above is violated. The problem is unidentifiable when one of the graphs of the mixture has a connected component with less than three edges. Moreover, there exist (many) graphs which violate the separability condition, and for which any algorithm would require at least exponential (in the number of nodes) sample complexity.

Sample Complexity: We prove a lower bound on the sample complexity of the problem, and show that our algorithm always matches the lower bound up to log factors in terms of the number of nodes N. It also matches the bound exactly in terms of the dependency in the separation parameter $\frac{1}{\Delta}$ if the graphs have min-degree at least 3.

Extensions: We give similar guarantees for the case of directed graphs of min-degree at least 3, and of undirected graphs with unbalanced and/or unknown mixtures priors. Finally, we discuss how to obtain numerical solutions for $K > 2$ mixtures.

5.2 Preliminaries

We consider an instance of the *independent cascade model* [34, **?**]. We observe independent epidemics spreading on a mixture of two graphs. In this section, we specify the dynamics of the spreading process, the observation model, and the learning task.

5.2.1 Mixture Model

We consider two *weighted* graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on the *same set of vertices* V. Unless specified otherwise, the graphs considered are *undirected*: $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$. Note that p_{ij} (q_{ij}) is 0 if there is no edge between i and j in G_1 (G_2).

We say that the mixture is Δ-separated if:

$$\min_{(i,j)\in E_1 \cap E_2} |p_{ij} - q_{ij}| \geq \Delta > 0.$$

We denote the minimum edge weight by $p_{min} := \min_{(i,j)\in E_1} \min_{(k,l)\in E_2} \min(p_{ij}, q_{kl})$. Note that p_{min} is positive.

5.2.2 Dynamics of the Spreading Process

We observe M independent identically distributed epidemic cascades, which come from the following generative model.

Component Selection: At the start of a cascade, an i.i.d. Bernoulli random variable $b \in \{1, 2\}$ with parameter α ($\Pr[b = 1] = \alpha$) decides the component of the mixture, *i.e.*, the epidemic spreads on graph G_b. We say that the mixture is balanced if $\alpha = 0.5$, and we call α and $1 - \alpha$ the priors of the mixture. Unless specified otherwise, the results presented are for balanced mixtures.

Epidemic Spreading: Once the component of the mixture G_b is fixed, the epidemic spreads in *discrete time* on graph G_b according to a regular one-step *Susceptible* $\rightarrow$ *Infected* $\rightarrow$ *Removed* (SIR) process [63, **?**]. At $t = 0$, the epidemic starts on a *unique* source, chosen uniformly at random among the nodes of V. The source is in the Infected state, while all the other nodes are in the Susceptible state. Let I_t (resp. R_t) be the set of nodes in the Infected (resp. Removed) state at time t. At each time step $t \in \mathbb{N}$, all nodes in the Infected state try to infect their neighbors in the Susceptible state, before transitioning to the Removed state during this same time step (*i.e.*, $R_{t+1} = R_t \cup I_t$) [2]. If i is in the Infected state at time t, and j is in the Susceptible state at the same time (*i.e* $i \in I_t, j \in S_t$), then i infects j with probability p_{ij} if $b = 1$, and q_{ij}

[2]Once a node is in the Removed state, the spread of the epidemic proceeds as if this node were no longer on the graph.

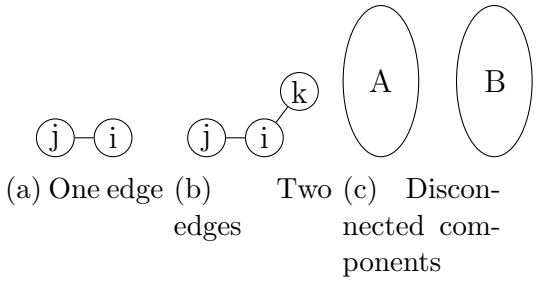


Figure 5.1: Unsolvable structures

if $b = 2$, with $0 \leq p_{ij} \leq 1$. Note that multiple nodes in the Infected state can infect the same node in the Susceptible state. The process ends at the first time step such that all nodes are in the Susceptible or Removed state (*i.e.*, no node is in the Infected state).

One realization of such a process, from randomly picking the component of the mixture and the source at $t = 0$ to the end of the process, is called a cascade.

5.2.3 Observation Model

For each cascade we do not have the knowledge of the underlying component, that is, we do not observe b and we treat this as a missing label. For each cascade, we have access to the complete list of infections: we know which node infected which node at which time (one node can have been infected by multiple nodes). This list constitutes a sample from the underlying mixture model.

5.2.4 Learning Objective

Our goal is to learn the weights of all the edges of the underlying graphs of the mixture, up to precision $\epsilon < \min(\Delta, p_{min})$. Specifically, we want to provide $\hat{p}_{ij}$ and $\hat{q}_{ij}$ for all vertex pairs $i, j \in V$ such that $\max_{i,j \in V^2} \max(|p_{ij} - \hat{p}_{ij}|, |q_{ij} - \hat{q}_{ij}|) < \epsilon$.

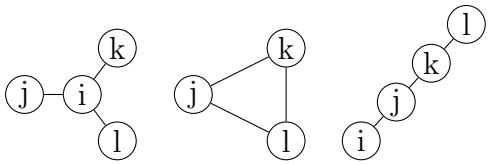
5.2.5 When is this problem solvable?

Prior to presenting our main results, we offer some intuition. We show that it is not always possible to learn the weights of both components of the mixture, even for settings that appear deceptively easy.

Indeed, it is impossible to learn the graph on two nodes i and j, with only one directed edge from i to j (see Figure 5.1a). To see this, consider a balanced mixture, for which edge (i, j) has weight β in G_1, and weight $1 - \beta$ in G_2. Node i will infect node j half of the time, independently of the value of β. This shows that we cannot recover the original weights, and the mixture problem is not solvable. If we add another edge, and i is now connected to a new node k (see Figure 5.1b), the problem is still not solvable (see Supplementary Material).

Surprisingly, if i has a third neighbor l (see Figure 5.2a), it becomes possible to learn the weights of the mixture. Learning this local structure is one of the main building blocks of our algorithm.

One might think that four nodes are needed for this problem to be



(a) Star (b) Triangle (c) Line

Figure 5.2: Solvable local structure

solvable. However, we can learn the edges of a triangle (see Figure C.2c). Similarly, the intuition that nodes need to be of degree at least three is misleading. If a line has more than three nodes (see Figure 5.2c), it is solvable. The line on four nodes is the other local structure which forms the foundation of our algorithm.

On the other end, the setting for which there exists (at least) two parts of the graph A and B for which cascades never overlap is a general unsolvable setting (see Figure 5.1c). We write $A_i = A \cap E_i, B_i = B \cap E_i$. Let $E'_1 = A_1 \cup B_2$, $E'_2 = A_2 \cup B_1$. We notice that a mixture spreading on edges E_1 and E_2 yields the same cascade distribution as a mixture on E'_1 and E'_2. Therefore, the solution is not unique.

The three simple shapes in Figure 5.2 form the core of this chapter. Our key insight is in showing that any graph that can be built up using these three building blocks (i.e., each node belongs in at least one of these structures) is solvable. This effective decomposition succeeds in reducing a general problem to a small number of sub-problems, for which we provide a solution.

5.3 Main Results

In this section we present our main results on the impossibility and recoverability of edge weights for a balanced mixture.

5.3.1 Balanced Mixture of Undirected Graphs

Below, we say it is impossible to recover the edge weights if two or more mixtures of graphs could have produced the same cascades distribution.

Impossibility Result Under Infinite Samples

Condition 1. *The graph $G = (V, E_1 \cup E_2)$ is connected and has at least three edges: $|E_1 \cup E_2| \geq 3$.*

Claim 1. *Suppose Condition 1 is violated. Then it is impossible to recover the edge weights corresponding to each graph (even with infinite samples).*

Impossibility Result Under Polynomial Samples

Condition 2. *The mixture is Δ-separated: for every edge $(i, j) \in E_1 \cup E_2$, $p_{ij} \neq q_{ij}$.*

Claim 2. *Suppose Condition 2 is violated. Then there exist (many) graphs for which we need at least exponential (in the number of nodes N) samples to recover the edge weights.*

Recoverability Result with Finite Samples

Theorem 5.3.1. *Suppose Conditions 1 and 2 are true. Then there exists an algorithm that runs on epidemic cascades over a balanced mixture of two undirected, weighted graphs* $G_1 = (V, E_1)$ *and* $G_2 = (V, E_2)$*, and recovers the edge weights corresponding to each graph up to precision* ϵ *with probability at least* $1 - \delta$*, in time* $O(N^2)$ *and sample complexity* $O\left(\frac{N}{\epsilon^2 \cdot \Delta^4} \log(\frac{N}{\delta})\right)$*, where* $N = |V|$*.*

Remark on Partial Recovery: An important element of our results is that if Conditions 1 and 2 are not satisfied for the entire graph we can still recover the biggest subgraph which follows these conditions. In particular, if the graph we obtain by removing all non Δ-separated edges is still connected, we can detect and learn all the edges of the graph (see Supplementary Material for more details). This is important, as it effectively means that we are able to learn the mixtures in the parts of the graph that matter most. On a practical note, this also means that our algorithm is resistant to the presence of bots in the network that would repost everything indifferently.

5.3.2 Extensions

Extension to Directed Graphs Interestingly, the techniques used to prove the theorem above can be immediately applied to learn mixtures of directed graphs of out-degree at least three (see Supplementary Material for complete proof). Note that the better dependency in $\frac{1}{\Delta}$ comes from the assumption on

the degree [3]. Since many applications on social networks can ignore nodes of out-degree less than three, as thoses nodes have very little impact on any diffusion phenomena, this result is of independent interest:

Theorem 5.3.2. *Suppose Conditions 1 and 2 are true. Then there exists an algorithm that runs on epidemic cascades over a balanced mixture of two* ***directed****, weighted graphs of minimum out-degree three* $G_1 = (V, E_1)$ *and* $G_2 = (V, E_2)$*, and recovers the edge weights of each graph up to precision* ϵ *with probability at least* $1 - \delta$*, in time* $O(N^2)$ *and sample complexity* $O\left(\frac{N}{\epsilon^2 \cdot \Delta^2} \log(\frac{N}{\delta})\right)$*, where* $N = |V|$*.*

Extension to Unbalanced/Unknown Priors If the mixture is unbalanced, but the priors are known, we can adapt our algorithm to learn the mixture *under the same conditions as above*, at the price of a higher dependency in $\frac{1}{\Delta}$. If the priors are unknown, we can only recover graphs of min-degree at least three.

5.3.3 Lower Bounds

We provide two lower bounds for mixtures of two graphs, one for undirected graphs, one for directed graphs.

Theorem 5.3.3. *When learning the edge weights of a balanced mixture on two* Δ*-separated graphs on* N *nodes up to precision* $\epsilon < \Delta$*, we need:*

[3] This immediately implies a better dependency in $\frac{1}{\Delta}$ for learning undirected graphs of minimum degree three.

1. $\Omega\left(\frac{N}{\Delta^2}\right)$ *samples for undirected graphs, which proves our algorithm is optimal in* N *up to log factors in this setting.*

2. $\Omega\left(N\log(N) + \frac{N\log\log(N)}{\Delta^2}\right)$ *samples for directed graphs of minimum out-degree three, which proves our algorithm has optimal dependency in* N *and in* $\frac{1}{\Delta^2}$ *in this setting.*

5.4 Balanced Mixture of Undirected Graphs

In this section, we provide our main algorithm (Algorithm 3) that recovers the edge weights of the graphs under the conditions presented in Theorem 5.3.1.

Algorithm 3 Learn the weights of undirected edges

Input Vertex set V
Output Edge weights for the two epidemics graphs

$E \leftarrow$ LearnEdges(V) ▷ **Learn the edges**
$S, W \leftarrow$ Learn2Nodes(V, E) ▷ **Initialize**
while $S \neq V$ **do**
  Select $u \in S, v \in V \backslash S$ such that $(u, v) \in E$
  if deg(v) ≥ 3 **then** ▷ **Use star primitive**
    $W \leftarrow W \cup$ LearnStar(v, E, W)
  if deg(v) $= 2$ **then** ▷ **Use line primitive**
    Set $w \in S$ such that $(u, w) \in E$
    Set $t \in V$ such that $(v, t) \in E$ and $t \neq u$
    if $t \notin S$ **then**
      $W \leftarrow W \cup$ LearnLine(t, v, u, w, S, W)
  $S \leftarrow S \cup \{v\}$
Return W

5.4.1 Overview of Algorithm 3

First, the algorithm learns the edges of the underlying graph using the procedure LearnEdges. To detect whether an edge (u, v) exists in $E_1 \cup E_2$, we use a simple estimator (Section 5.4.2.1). This also provides us with the degree of each node with respect to $E_1 \cup E_2$.

With the knowledge of the structure of the graph, to learn the edge weights adjacent to a node, our algorithm uses two main procedures, LearnStar and LearnLine. If a node is of degree at least three (*e.g.*, node u in Figure 5.3), procedure LearnStar recovers all the edge weights (i.e., the weights of the two mixtures for these edges) adjacent to this node *independently* of the rest of the graph. Otherwise, if a node is of degree two (*e.g.*, node u in Figure 5.4), procedure LearnLine learns all the edge weights adjacent to this node *independently*. Both procedures use carefully designed estimators that exploit the respective structures. We present the above estimators for balanced mixtures in Section 5.4.2. We require Condition 2 for the existence of the proposed estimators.

Our main algorithm maintains a set of *learned nodes*. A node is a *learned node* if the weights for all the edges adjacent to it have been learned. The algorithm begins with learning two connected nodes (two nodes with an edge in between) using procedure Learn2Nodes. Next it proceeds iteratively, by learning the weights of the edges connected to one *unlearned neighbor* of the *learned nodes* using the two procedures discussed above. The algorithm terminates when all the nodes in V are learned.

5.4.2 Learning Edges, Star Vertices, and Line Vertices

In this section, we show how we recover the weights for local structures using moment matching methods. Our proof relies on a few crucial ideas. First, we introduce local estimators, which can be computed from observable events in the cascade and are polynomials of the weights of the mixture. General systems of polynomial equations are hard to solve. However, we found ways of combining these specific estimators to decouple the problem, and obtain $\mathcal{O}(|E|)$ systems of six polynomial equations of maximum degree three, with six unknowns. Finally, we show how to elegantly obtain a closed-form solution for these systems.

5.4.2.1 Learning the Edges in $E_1 \cup E_2$

We recall that I_0 is the random variable indicating the set containing the unique source of the epidemic for a cascade. If an epidemic cascade starts from node u, then for any node a that is infected in time step 1 there is an edge $(u, a) \in E_1 \cup E_2$. This provides us with the average weight of the edge (u, a) as X_{ua},

Claim 3. *If u and a are two distinct nodes of V such that $(u, a) \in E_1 \cup E_2$, then:*

$$X_{ua} := \Pr(u \to a \mid u \in I_0) = \frac{p_{ua} + q_{ua}}{2} \geq \frac{p_{min}}{2}.$$

Furthermore, there exists an edge between u and a in $E_1 \cup E_2$, if and only if $X_{ua} \geq \frac{p_{min}}{2} > 0$.

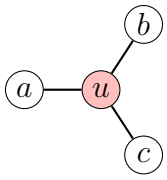


Figure 5.3: A star vertex u, with edges $(u, a), (u, b)$ and (u, c) in $E_1 \cup E_2$.

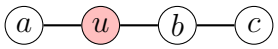


Figure 5.4: A line vertex u, with edges $(u, a), (u, b)$ and (b, c) in $E_1 \cup E_2$.

The above claim can be leveraged to design algorithm LearnEdges, which takes as inputs all the X_{ua} for all pairs (u, a), and returns all the edges of $E_1 \cup E_2$ (see Supplementary Materials).

Conditioning on Source Node: We notice that the expression of X_{ua} is a function of weights of edges (u, a). Here, conditioning on the event "$u \in I_0$" is crucial. Indeed, if the source had been any node other than u, the probability that a was in the Susceptible state when u was infected would have depended on the (unknown) weights of the paths connecting the source and node a. We could not have obtained the simple expression above.

5.4.2.2 Star vertex

A star vertex is a vertex $u \in V$ of degree at least three in $E_1 \cup E_2$ (Figure 5.3). We consider:

$Y_{ua,ub}$: the probability the star vertex u infects neighbors a and b, conditioned on u being the source vertex.

Claim 4. *For u and a, b and c as in Figure 5.3:*

$$Y_{ua,ub} = \Pr(u \to a, u \to b \mid u \in I_0) = \frac{p_{ua}p_{ub} + q_{ua}q_{ub}}{2}.$$

We emphasize that the conditioning is once again crucial to obtain such a simple form for $Y_{ua,ub}$. Further, we make a key observation that for $i \neq j \in \{a, b, c\}$,

$$Y_{ui,uj} - X_{ui}X_{uj} = \frac{(p_{ui} - q_{ui})(p_{uj} - q_{uj})}{4}. \tag{5.1}$$

This directly leads to the closed-form expressions for the weights of the edges adjacent to the star vertex u.

Lemma 5.4.1. *Suppose Conditions 1 and 2 are true and $\alpha = 1/2$. Let $s_{ua} \in \{-1, 1\}$. The weight of any edge (u, a) connected to a star vertex u, with distinct neighbors a, b and c in $E_1 \cup E_2$, is given by:*

$$p_{ua} = X_{ua} + s_{ua}\sqrt{\frac{(Y_{ua,ub} - X_{ua}X_{ub})(Y_{ua,uc} - X_{ua}X_{uc})}{Y_{ub,uc} - X_{ub}X_{uc}}},$$

$$q_{ua} = X_{ua} - s_{ua}\sqrt{\frac{(Y_{ua,ub} - X_{ua}X_{ub})(Y_{ua,uc} - X_{ua}X_{uc})}{Y_{ub,uc} - X_{ub}X_{uc}}}.$$

Furthermore, any two signs s_{ui} and s_{uj}, for $i \neq j$ and $i, j \in \{a, b, c\}$, satisfy $s_{ui}s_{uj} = \mathsf{sgn}(Y_{ui,uj} - X_{ui}X_{uj})$.

Resolving Mixture Ambiguity: Separating the weights of both graphs in the mixture is not enough to learn the mixture: we also have to assign the two weights to the right component of the mixture. The identity $s_{ui}s_{uj} = \mathsf{sgn}(Y_{ui,uj} - X_{ui}X_{uj})$ allows us to identify three weights belonging to

the same mixture component: p_{ua}, p_{ub} and p_{uc}. If one of these weights had been learned before, it is immediate to assign the two new weights to the same component. This leads to the following algorithm:

LearnStar: This algorithm takes as input a star vertex u, the set of edges of $E_1 \cup E_2$, and all the X_{ui} and $Y_{ui,uj}$ for all (i, j) distinct neighbors of u, and returns all the weights of the edges connected to u in both mixtures using the above closed-form expressions (see Supplementary Materials).

5.4.2.3 Line vertex

We now consider a node u that has degree exactly two in $E_1 \cup E_2$ and forms a line structure. Specifically, let u, a, b and c be four distinct nodes of V, such that $(a, u), (u, b)$ and (b, c) belong in $E_1 \cup E_2$. We call such a node u a line vertex (see Figure 5.4).

To recover the weights of the edges adjacent to a line vertex, only considering events in the first two timesteps is insufficient. Contrary to a star vertex, for a line vertex we have only one second moment.

We circumvent the problem by considering:
1) $Y_{ub,bc}$: the probability of the event when (in Figure 5.4) u infects only b, and in turn b infects c, conditioned on u being the source.
2) $Z_{ua,ub,bc}$: the probability of the event when (in Figure 5.4) u infects both a and b, and in turn b infects c, conditioned on u being the source.

Claim 5. *For a line vertex u and nodes a, b and c as in Figure 5.4:*

- $Y^{|}_{ua,ub} = Pr(u \to a, u \to b \mid u \in I_0) = \frac{p_{ua}p_{ub}+q_{ua}q_{ub}}{2}$,
- $Y^{|}_{ub,bc} = \Pr(u \to b, b \to c \mid u \in I_0) = \frac{p_{ub}p_{bc}+q_{ub}q_{bc}}{2}$,
- $Z^{|}_{ua,ub,bc} = \Pr(u \to a, u \to b, b \to c \mid u \in I_0)$

$= \frac{p_{ua}p_{ub}p_{bc}+q_{ua}q_{ub}q_{bc}}{2}$.

The result for $Y^{|}_{ua,ub}$ is similar to Claim 4. However, the proof for $Y^{|}_{ub,bc}$ and $Z^{|}_{ub,bc,bc}$ not only requires u to be the source, but also relies on the fact that u is of degree 2, which implies $p_{uc} = q_{uc} = 0$.

We note that $Y^{|}_{bc,ua}$ does not exist, as c cannot be infected if b is not. So we cannot immediately replicate the star vertex proof. Let us define $R^{|} := X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}$. However, we prove the following equality, which acts as a surrogate for $(Y^{|}_{ua,bc} - X_{ua}X_{bc})$:

$$R^{|} = \frac{1}{4}(p_{ua} - q_{ua})(p_{bc} - q_{bc}). \tag{5.2}$$

As in Lemma 5.4.1, we now obtain the closed-form expressions for the weights associated with the line vertex u. For the sake of notation consistency, we define $Y^{|}_{bc,ua} := (R^{|} + X_{bc}X_{ua})$ (it has no probabilistic interpretation).

Lemma 5.4.2. *Suppose Conditions 1 and 2 hold and $\alpha = 1/2$, then for s_{ua}, s_{ub}, and s_{bc} in $\{-1, 1\}$, the weights of the edges for a line structure are*

given by:

$$\forall(e1,e2,e3) \in \{(ua,ub,bc),(ub,bc,ua),(bc,ua,ub)\},$$
$$p_{e1} = X_{e1} + s_{e1}\sqrt{\frac{(Y^{|}_{e1,e2} - X_{e1}X_{e2})(Y^{|}_{e3,e1} - X_{e3}X_{e1})}{Y^{|}_{e2,e3} - X_{e2}X_{e3}}},$$
$$q_{e1} = X_{e1} - s_{e1}\sqrt{\frac{(Y^{|}_{e1,e2} - X_{e1}X_{e2})(Y^{|}_{e3,e1} - X_{e3}X_{e1})}{Y^{|}_{e2,e3} - X_{e2}X_{e3}}}.$$

Furthermore, for all $e1, e2 \in \{ua, ub, uc\}$ *and* $e1 \neq e2$, $s_{e1}s_{e2} = \mathsf{sgn}(Y_{e1,e2} - X_{e1}X_{e2})$.

LearnLine: In the same fashion as for the star vertex, we can use the expression in Lemma 5.4.2 to design an algorithm LearnLine, which takes as input a line vertex u, the set of the edges of $E_1 \cup E_2$, and the limit of the estimators X_{ua}, X_{ub}, X_{bc}, $Y^{|}_{ua,ub}$, $Y^{|}_{ub,bc}$, $Z^{|}_{ua,ub,bc}$ for a, b and c as in Figure 5.4, and returns the weights of the edges $(u,a),(u,b)$ and (b,c) in both mixtures (see Supplementary Materials).

Learn2Nodes Our main algorithm is initialized by learning weights associated with edges connected to two nodes using subroutine Learn2Nodes. As this algorithm is very similar in spirit to our general algorithm, we leave the details to the Supplementary Materials.

5.4.3 Correctness of Algorithm 3

To prove the correctness of the main algorithm, we show the following invariant:

Lemma 5.4.3. *At any point in the algorithm, the entire neighborhood of any node of S has been learned and recorded in W:*

$$\forall u \in S,\ \forall v \in V,\ (u,v) \in E \implies (u,v) \in W.$$

Proof. We prove the above by induction on the iteration of the while loop. Due to the correctness of LEARN2NODES (proven in Supplementary material), after calling this function, W contains all edges adjacent to the two vertices in S. Hence the base case is true. Let us assume that after k iterations of the loop, the induction hypothesis holds.

We consider three cases in the $(k+1)$-th iteration:

• $\mathbf{deg(v) \geq 3}$: We recover all edges adjacent to the star vertex v by using LEARNSTAR (correct due to Lemma 5.4.1). Sign consistency is ensured using edge $(u,v) \in W$ since $u \in S$.

• $\mathbf{deg(v) = 2}$: There exists $w \in S$ such that $(u,w) \in E$ since $|S| \geq 2$ and is connected. Since $deg(v) = 2$, there exists $t \neq u$ such that $(t,v) \in E$. Now if $t \in S$ then $(t,v) \in W$ and we are done. If $t \notin S$ then v is a line vertex for $t - v - u - w$. By using LEARNLINE we recover all edges on the line (correct due to Lemma 5.4.2). Sign consistency is ensured through edge (v,u).

• $\mathbf{deg(v) = 1}$: Since $u \in S$, we have $(u,v) \in W$, so we are done.

Thus by induction, after every iteration of the for loop, the invariant is maintained. □

Theorem 5.4.4. *Suppose Conditions 1 and 2 are true; Algorithm 3 learns the edge weights of the two balanced mixtures in the setting of infinite samples.*

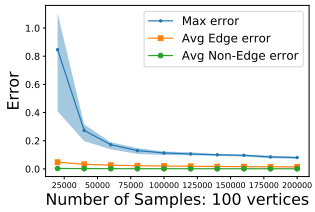
Proof. Since at every iteration, the size of S increases by 1, after at most $|V|$ iterations, we have $S = V$. Using Lemma 5.4.3, we also have $W = E_1 \cup E_2$. □

5.4.4 Finite Sample Complexity

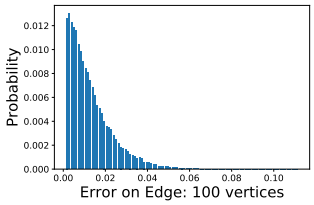
In this section, we investigate the error in estimating the quantities X_{ui}, $Y_{ui,uj}$ for $i, j \in \{a, b, c\}$ in the case of a star vertex, and X_{e1}, $Y^{|}_{e1,e2}$ and $Z_{ua,ub,bc}$ for $e1, e2 \in \{ua, ub, bc\}$ in the case of a line vertex, using a finite number of cascades. We further investigate the effect of the error in these quantities on the accuracy of the recovered weights.

We use a simple count-based estimator. Specifically, for events $\mathcal{E}_1$ and $\mathcal{E}_2$, we estimate the probability $\Pr(\mathcal{E}_1|\mathcal{E}_2) = \frac{\sum_{m=1}^{M} \mathbb{1}_{\mathcal{E}_1 \cap \mathcal{E}_2}}{\sum_{m=1}^{M} \mathbb{1}_{\mathcal{E}_2}}$. As a concrete example, we have the estimator for X_{ua} as $\hat{X}_{ua} := \frac{\sum_{m=1}^{M} \mathbb{1}_{u \to a, u \in I_0^m}}{\sum_{m=1}^{M} \mathbb{1}_{u \in I_0^m}}$. Here I_0^m denotes the source of the m-th cascade and $u \to a$ denotes that u infects a. We can argue, using the law of large numbers and Slutsky's Lemma, that the above approach provides us with balanced estimators.

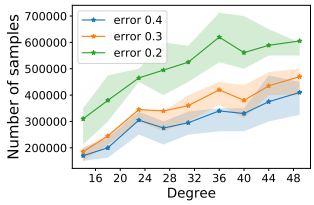
We first establish high probability error bounds for the base estimators with a finite number of cascades, for both the star vertex and the line vertex. Finally, using the above guarantees, we provide our main sample complexity



(a) Maximum and average absolute error as a function of the number of cascades on $\mathcal{G}(N,p)$ graphs, with $N = 100$ vertices, $p = \sqrt{N}$. The first graph has 1044 edges, the second one 1026.



(b) Normalized histogram of absolute error after 200000 cascades on the same $\mathcal{G}(N,p)$ graphs, with $N = 100$ vertices, $p = \sqrt{N}$, and 100 bins. The first graph has 1044 edges, the second one 1026.



(c) Number of cascades needed to reach given precision, as a function of the degree. All experiments are on random d-regular graphs on 50 nodes, with 375 to 1225 undirected edges.

Figure 5.5: Experimental sample complexity, error distribution, and dependency in the degree.

result for the balanced mixture problem. See Supplementary Material for proofs.

Theorem 5.4.5. *Suppose Condition 1 and 2 hold. With* $M = \mathcal{O}\left(\frac{1}{p_{min}^6 \Delta^4} \frac{N}{\epsilon^2} \log\left(\frac{N}{\delta}\right)\right)$ *samples, Algorithm 3 learns the edge weights of a balanced mixture on two graphs within precision* ϵ *with probability at least* $1 - \delta$.

5.5 Extensions

5.5.1 Extension to Directed Graphs

We notice that in the case of directed graphs of minimum out-degree three, we can simply use the star structure to learn all the directed edges. This, however, would not be enough to ensure mixture consistency; we therefore need

to also use two new structures to solve the problem. The algorithm is very similar to Algorithm 3, and the structures are very similar to the structures encountered so far. Precise details are left for the Supplementary Material.

5.5.2 Extension to Unbalanced/Unknown Priors

While previous results only considered balanced mixtures, *i.e.* with parameter $\alpha = 0.5$, we focus here on unbalanced mixtures ($\alpha \neq 0.5$ known) and on mixtures of unknown priors (α unknown).

We first note that the main algorithm for recovering the graph does not depend on the prior once the correct LearnStar and LearnLine primitives are provided. Therefore, we show how to design these primitives.

Unbalanced Mixtures We can easily extend Equation 5.1 for star vertices in the case of unbalanced mixtures. Specifically, we have for all $i \neq j \in \{a, b, c\}$:

$$Y_{ui,uj} - X_{ui}X_{uj} = \alpha(1-\alpha)(p_{ui} - q_{ui})(p_{uj} - q_{uj}).$$

However, Equation 5.2 does not extend easily (see Supplementary Material for details) in the general case:

Theorem 5.5.1. *Suppose Conditions 1 and 2 are true. Then there exists an algorithm that runs on epidemic cascades over an* ***unbalanced*** *mixture of two undirected, weighted graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, with $|V| = N$, and recovers the edge weights corresponding to each graph up to precision ϵ in time $O(N^2)$ and sample complexity:*

- $O\left(\frac{N\log N}{\epsilon^2}\text{poly}(\frac{1}{\Delta})\text{poly}(\frac{1}{\min(\alpha,1-\alpha)})\right)$ *in general.*
- $O\left(\frac{N\log N}{\epsilon^2\Delta^2}\text{poly}(\frac{1}{\min(\alpha,1-\alpha)})\right)$ *for graphs of minimum degree three.*

Mixtures of Unknown Priors If the graph has at least one star vertex, we can learn the entire mixture by learning the parameter α from this node, and use the results from above to learn the rest of the graph once α has been recovered. Details can be found in Supplementary Material.

5.5.3 Extension to Mixtures of $K > 2$ Graphs

For graphs of minimum degree $2K-1$, writing the equations using first and second moments (*i.e.* the X_{ua} and $Y_{ua,ub}$) as above yields at least as many equations as unknowns. Using Quadratic Constraints Quadratic Programming, we can obtain numerical solutions. Note that the constraints are not convex, so there is no guarantee this problem is solvable in polynomial time. As there is also no immediate reduction to a NP-hard problem, we do not know the complexity of learning mixtures of $K > 2$ graphs.

5.6 Experiments

We validate our results on synthetic data. We first draw random graphs from a distribution (specified below), and each sample is a simulation of a cascade spreading on it. Once the graphs are drawn, the experiments are run 10 times. The shaded area represents the 25^{th} to 75^{th} percentiles.

Our first two experiments are on Erdös-Renyi $\mathcal{G}(N, p)$ graphs. In Figure 5.5a, we investigate the maximum error on learned edges and compare it with the average error. We find that the "Max error" curve follows the dependence predicted by our theory, of $\epsilon = \mathcal{O}\left(1/\sqrt{N \log(N)}\right)$. It is also worth noting that the average error on the existing edges is one order of magnitude smaller than the max error. By plotting the normalized histogram of the absolute value of the errors (Figure 5.5b), we confirm that only a few edges keep the maximum error high. Finally, by varying the degree on random d-regular graphs with a fixed number of vertices (Figure 5.5c), we see that the sample complexity is multiplied by 2 as the number of edges grows from $\Theta(N)$ to $\Theta(N^2)$, as predicted since the dependence in the degree is logarithmic. Therefore, our algorithm is as sample-efficient (up to small constants) on dense graphs as on sparse graphs.

5.7 Conclusion

We provided an efficient algorithm for learning the edge weights of a balanced mixture of two undirected graphs from epidemic cascades, as well as matching lower bounds (up to log factors). We extended our results to directed graphs of min-degree three, and unbalanced/unknown mixtures.

Our algorithm is robust, in the sense that it has partial recovery guarantees, and it is unaffected by adversarial examples which would consist of adding nodes/edges. Due to its local structure, it is also easily parallelizable.

Learning mixtures of more than two graphs, or mixtures of directed

graphs without restriction on the minimum degree, are still open problems, and are left for future work.

Appendices

Appendix A

Uncertainty about who is infected

A.0.1 Properties of the binary tree

We first establish a few properties of the complete binary tree.

Proposition A.0.1. *The* CutWidth *of the complete binary tree is smaller than* $\log(N)$.

Proof. Consider the crusade [23] implied by a Deep First Search over the tree. This crusade has a maximal cut of $\log(N) - 1$. Thus, by definition, the CutWidth is lower than $\log(N) - 1$.

Proposition A.0.2. *There are* r^3 *subtrees containing* $\frac{N}{r^3}$ *nodes, and they are at distance* $\mathcal{O}(\log\log(N))$ *from the root.*

Proof. In the complete binary tree, there are 2^k subtrees at distance k from the root that contain $\frac{N}{2^k}$ nodes. The results follows for $k = \frac{3\log(r)}{\log(2)}$.

□

A.0.2 Some probabilities

A.0.2.1 Geometric variables

Proposition A.0.3. *The minimum of i independent geometric random variables of parameter μ is a geometric random variable of parameter $1-(1-\mu)^i$.*

Proof. Let $Geo(i,\mu)$ be the minimum of i independent geometric random variables. Then:

$$\begin{aligned}\mathbb{P}(Geo(i,\mu) \geq k) &= ((1-\mu)^{k-1})^i \\ &= ((1-\mu)^i)^{k-1}\end{aligned}$$

We recognize the probability distribution of a geometric variable with parameter $1-(1-\mu)^i$.

Lemma A.0.4. *As $\tau \to 0$, it takes less than $\frac{\log(k)}{\tau}$ time steps in expectation to infect k new nodes.*

Proof. Every new infection increases the cut by 1. Let $Geo(i,\mu)$ be the minimum of i geometric random variables of parameter μ, and let T_k be the time it takes to infect the k new nodes. We have:

$$T_k = \sum_{i=1}^{k-1} Geo(i,\mu)$$

Using Claim A.0.3, $Geo(i,\mu)$ is a geometric variable of parameter $1-(1-\mu)^i$.

Therefore:

$$\begin{aligned}\mathbb{E}(T_k) &= \mathbb{E}\left(\sum_{i=1}^{k-1} Geo(i,\mu)\right)\\ &= \sum_{i=1}^{k-1} \frac{1}{1-(1-\mu)^i}\\ &=_{\tau\to 0} \sum_{i=1}^{k-1} \frac{1}{i\tau}\\ &\leq_{\tau\to 0} \frac{\log(k)}{\tau}\end{aligned}$$

□

A.0.2.2 Some curing probabilities

Proposition A.0.5. *If all the budget at a given time step is spent, the probability that no nodes are cured in this time step is* $1-\delta$.

Proof. Let r_i be the budget attributed to the node i. Then:

$$\begin{aligned}P_{\text{NoCuring}} &= \prod_{i=1}^{N} 1-\delta_i = \prod_{i=1}^{N} e^{-r_i\tau}\\ &= e^{-\left(\sum_{i=1}^{N} r_i\right)\tau} = e^{-r\tau}\\ &= 1-\delta.\end{aligned}$$

□

Lemma A.0.6. *The probability $P_{\mathrm{m}}^{\mathrm{path}}$ that m nodes are reinfected along a path, such that no node on the m-length path is cured before they all become infected, is lower bounded by $\left(\frac{\mu(1-\delta)}{\delta+\mu(1-\delta)}\right)^{m+1}$.*

Proof. Using Proposition A.0.5,

$$\begin{aligned}
P_{\mathrm{m}}^{\mathrm{path}} &\geq \sum_{t=0}^{\infty} \binom{m+t}{m} \mu^{m+1}(1-\mu)^t(1-\delta)^{m+t} \\
&\geq (\mu(1-\delta))^{\mathrm{m}} \cdot \mu \cdot \sum_{t=0}^{\infty} \binom{m+t}{m} ((1-\mu)(1-\delta))^t \\
&\geq (\mu(1-\delta))^m \cdot \mu \cdot \frac{1}{(1-(1-\mu)(1-\delta)))^{m+1}} \\
&\geq \left(\frac{\mu(1-\delta)}{\delta+\mu(1-\delta)}\right)^{m+1}.
\end{aligned}$$

□

Corollary A.0.7. *The probability $P_{\mathrm{m}}^{\mathrm{startPath}}$ that m nodes are reinfected along a path, such that no node on the m-length path is cured before they all become infected, and such that there is an infection on the first time step, is lower bounded by $\mu \cdot \left(\frac{\mu(1-\delta)}{\delta+\mu(1-\delta)}\right)^{m}$.*

Proof. Taking into account that the first time step is an infection:

$$\begin{aligned}
P_{\mathrm{m}}^{\mathrm{startPath}} &\geq \sum_{t=0}^{\infty} \binom{m-1+t}{m-1} \mu^{m+1}(1-\mu)^t(1-\delta)^{m+t} \\
&\geq \mu \cdot \left(\frac{\mu(1-\delta)}{\delta+\mu(1-\delta)}\right)^{m}.
\end{aligned}$$

□

Proposition A.0.8. *Let $T_{\frac{N}{2r^4}}$ be the random variable representing the time to cure half of the $\frac{N}{r^4}$ last nodes. Then:*

$$\mathbb{P}\left(T_{\frac{N}{2r^4}} \leq \frac{N}{4r^5\delta}\right) \leq e^{-\frac{N}{8r^5}}.$$

Proof. The difficulty here lies in the fact that we want to obtain exponential concentration inequalities on a sum of geometric variables, which are unbounded. Therefore, we cannot directly use a Chernoff's bound. Following an idea from [10], we represent geometric variables as the sum of Bernoulli variables. Each variable is then bounded, which makes the analysis possible.

Let X_i^t be 1 if node i was cured at time t, and 0 otherwise. Let X^t be r with probability δ, and 0 otherwise. We notice $\mathbb{P}(X_i^t = 1) \leq \delta$, and $\forall t, \sum_{i=1}^{N} X_i^t \leq r$. By using Chernoff's bound on a sum of $\frac{N}{4r^5\delta}$ Bernoulli variables of parameter δ, we can therefore bound the probability that curing $\frac{N}{2r^4}$

nodes happens in a short time (here less than $\frac{N}{4r^5\delta}$ time):

$$\begin{aligned}
\mathbb{P}(T_{\frac{N}{2r^4}} \leq \frac{N}{4r^5\delta}) &= \mathbb{P}(\sum_{t=1}^{\frac{N}{4r^5\delta}} \sum_{i=1}^{N} X_i^t \geq \frac{N}{2r^4}) \\
&\leq \mathbb{P}(\sum_{t=1}^{\frac{N}{4r^5\delta}} X^t \geq \frac{N}{2r^4}) \\
&\leq \mathbb{P}(\sum_{t=1}^{\frac{N}{4r^5\delta}} \frac{X^t}{r} \geq \frac{N}{2r^5}) \\
&\leq \mathbb{P}\left(\sum_{t=1}^{\frac{N}{4r^5\delta}} \frac{X^t}{r} \geq \frac{N}{4r^5\delta} \cdot \delta \cdot (1+1)\right) \\
&\leq \mathbb{P}\left(\sum_{t=1}^{\frac{N}{4r^5\delta}} \frac{X^t}{r} \geq \mathbb{E}\left[\sum_{t=1}^{\frac{N}{4r^5\delta}} \frac{X^t}{r}\right] \cdot (1+1)\right) \\
&\leq e^{-\frac{\frac{N}{4r^5} \cdot 1^2}{3}} \\
&\leq e^{-\frac{N}{12r^5}}.
\end{aligned}$$

□

Proposition A.0.9. *Conditioned on reaching a cut of* $3r$ *in a minimal tree in less than* $\frac{30\log(r)}{\tau}$ *time steps, the probability of not infecting any nodes outside of the escape* $P_{\text{NoOtherInfections}}$ *is bounded by:*

$$\begin{aligned}
P_{\text{NoOtherInfections}} &\leq e^{-\frac{360\log^2(r)\mu}{\tau}} \\
&\leq_{\tau \to 0} e^{-360\log^2(r)}.
\end{aligned}$$

Proof. For an infection to not be part of the *Escape*, it has to happen because of a node which is either on the path to the root, or on the path to a minimal

tree. As calculated before, there are $12\log(r)$ such nodes, which all have at most one edge not on the path (the two others were used either to get infected, or to infect the next node on the path). What's more, each of these nodes was infected for at most $\frac{30\log(r)}{\tau}$ time steps. The probability of not infecting any node along those edges during all these time steps is therefore:

$$\begin{aligned} P_{\text{NoOtherInfections}} &\leq \left((1-\mu)^{12\log(r)}\right)^{\frac{30\log(r)}{\tau}} \\ &\leq (1-\mu)^{\frac{360\log^2(r)}{\tau}} \\ &\leq e^{-\frac{360\log^2(r)\mu}{\tau}}. \end{aligned}$$

As τ goes to 0:

$$\begin{aligned} P_{\text{NoOtherInfections}} &\leq e^{-\frac{360\log^2(r)\mu}{\tau}} \\ &\leq_{\tau\to 0} e^{-360\log^2(r)}. \end{aligned}$$

□

A.0.2.3 Moment generating function of the random walk

Proposition A.0.10. *There exists $x^* > 0$ such that the Moment Generating Function (MGF) of G_t evaluated at x^* is 1.*

Proof. Since G_t is a sum of independent random variables, and since the MGF of a Bernouilli random variable of parameter p is equal to $MGF(x) = pe^x + (1-p)$:

$$\begin{aligned} MGF_{G_t}(x) &= MGF_{C_t}(x) \cdot MGF_{I_t}(x) \\ &= (\delta e^x + (1-\delta))^r \cdot (\mu e^{-x} + (1-\mu))^{\frac{r^3}{3}} \end{aligned}$$

We can see that:

$$MGF(0) = 1, \qquad MGF'(0) < 0, \qquad MGF(r) \to_{r\to\infty} \infty$$

Therefore, by the Intermediate Value Theorem:

$$\exists x^* > 0, MGF(x^*) = 0$$

There is no closed form solution for x^*, but we can get an approximation when $\tau \to 0$.

□

Proposition A.0.11. *When $\tau \to 0$, we have a closed form solution: $x^* = \log(r) - \log(3)$.*

Proof. When $\tau \to 0$:

$$\begin{aligned}
1 = MGF_{G_t}(x^*) &= (\delta e^{x^*} + (1-\delta))^r \cdot (\mu e^{-x^*} + (1-\mu))^{\frac{r^3}{3}} \\
&=_{\tau\to 0} (r\tau e^{x^*} + (1 - r\tau))^r \cdot (\tau e^{-x^*} + (1-\tau))^{\frac{r^3}{3}} \\
&=_{\tau\to 0} (1 + r\tau(e^{x^*} - 1))^r (1 + \tau(e^{-x^*} - 1))^{\frac{r^3}{3}} \\
&=_{\tau\to 0} (1 + r^2\tau(e^{x^*} - 1))(1 + \tau\frac{r^3}{3}(e^{-x^*} - 1)) \\
&=_{\tau\to 0} 1 + \tau\left(r^2(e^{x^*} - 1) - \frac{r^3}{3}(1 - e^{-x^*})\right) \\
&=_{\tau\to 0} 1 + \tau\left(r^2 \cdot e^{-x^*}(e^{2x^*} - e^{x^*} - \frac{r \cdot e^{x^*}}{3} + \frac{r}{3})\right)
\end{aligned}$$

If we want to nullify the first order in τ, we need:

$$(e^{x^*})^2 - (1 + \frac{r}{3})(e^{x^*}) + \frac{r}{3} = 0$$

This is a second order polynomial, which gives us the solution $e^{x^*} = 1$ (trivial solution for $x^* = 0$), and $e^{x^*} = \frac{r}{3}$, which gives us a non-trivial solution:

$$x^* = \log(r) - \log(3) > 0$$

□

A.0.3 Some calculus

A.0.3.1 monotonicity results

Proposition A.0.12. *The function $k(x) = \frac{x}{1-e^{-rx}}$ is increasing in x. In particular, for all $x \leq 0$, we have $k(x) \geq k(0) = \frac{1}{r}$.*

Proof. $x \to x$ and $x \to \frac{1}{1-e^{-rx}}$ are both increasing functions of x, so $k(x)$ is also increasing.

□

A.1 A policy achieving the upper bound

The main contribution of Chapter 3 proves a lower bound on the budget in the Partial information setting. We prove that for budget $r = \mathcal{O}(\log(N))$, there exists no strategy which allows polynomial expected curing time, unless $D(p||q)/\tau$ goes to infinity. Moreover, our result implies that if $D(p||q)/\tau = 0$,

then for budget $r = \mathcal{O}(\text{poly}(\log(N)))$, there exists no strategy which allows polynomial expected curing time.

We now study the converse problem. In this section we exhibit a policy which:

- Does not require any knowledge of the state (works even in the Blind Curing setting);
- Achieves linear expected curing time;
- Needs $r \sim \mathcal{O}(e^{\frac{4}{c}} \cdot N^c)$ budget, for any $c > 0$.

A.1.1 Description of the policy

We consider the ordering O of the nodes given by a Depth First Search on a binary tree. We split the graph into 3 sets: A_{sus}, A_{inf} and A_{buff}. Intuitively, these sets respectively represent the set of the nodes we believe are cured, the set of nodes we believe are infected, and the buffer zone in the middle.

We run through the following algorithm. As we show in Section A.1.4, the probability that we fail to cure the graph in one pass of the algorithm below (what we call one *iteration*), is at most $2/N$, and hence the expected time to cure, given our budget, is linear.

To initialize each pass of the algorithm, we set $t = 0$, and also initialize the sets $A^0_{\text{sus}} = A^0_{\text{buff}} = \emptyset$, $A^0_{\text{inf}} = V$.
Every $\frac{1}{\tau}$ time steps, we:

- move a node from A_{inf} to A_{buff}, following the ordering O
- remove all the nodes from A_{buff} which are at distance greater than $c\log(N)$ from any node of A_{inf}, and place them in A_{sus}

Then, during $\frac{1}{\tau}$ time steps, we:

- cure all the nodes of A_{buff} with constant budget c_1;
- cure the new node with budget $(1 + c_2)\log(N)$, where c_2 constant.

This gives a total budget of $2^{c\log(N)} \cdot c_1 + c_2\log(N) = N^{c\log(2)} \cdot c_1 + c_2\log(N)$.

At time step $t = \frac{N}{\tau}$, when $A_{\text{inf}}^{\frac{N}{\tau}} = \emptyset$, we keep curing $A_{\text{buff}}^{\frac{N}{\tau}}$ for an additional $\frac{c\log(N)}{\tau}$ time steps.

One pass through the set of actions described above is called an **iteration**. We show below that the probability of failing to cure the entire graph in one iteration is bounded by $2/N$, and hence we can cure the graph in linear expected time. Equivalently, in time $\gamma \cdot N$, one can get a $(1-\epsilon)$-probability guarantee that the graph is cured, for any $\epsilon > 0$.

A.1.2 Properties of the policy

Proposition A.1.1. *Every node of the graph spends at least* $\frac{c\log(N)}{\tau}$ *time steps in* A_{buff}.

Proof. We notice A_{inf} is connected at all time. Therefore, when a node i is removed from A_{inf} and added to A_{buff}, it is at distance 1 from a node of A_{inf}. Every subsequent node transferred from A_{inf} to A_{buff} can only increase the distance between i and A_{inf} by 1. Since a new node is transferred every $\frac{1}{\tau}$ time steps, and all the nodes at distance no greater than $c\log(N)$ from A_{inf} are kept in A_{buff}, every node i of the graph spends at least $\frac{c\log(N)}{\tau}$ time steps in A_{buff}.

□

Proposition A.1.2. *Let T_{cured} be the time it takes to cure the graph, and $P_{\text{OneIteration}}$ be an lower bound on the probability that the graph is cured in one iteration. Then:*

$$\mathbb{E}[T_{\text{cured}}] \leq \frac{N + c\log(N)}{P_{\text{OneIteration}}}.$$

Proof. T_{cured} is stochastically dominated by an exponential variable with parameter $P_{\text{OneIteration}}$, which in turn has expectation $\frac{1}{P_{\text{OneIteration}}}$. One iteration lasts exactly $\frac{N}{\tau} + \frac{c\log(N)}{\tau}$ time steps, and one time step lasts τ time, so an iteration lasts $N + c\log(N)$ time.

□

A.1.3 Analysis

Definition A.1.1. We call an **epoch** $\frac{1}{\tau}$ consecutive time steps.

If there is at least one infected node at the end of the policy, then either one of the following events must have happen:

1. One node was not cured when it entered the buffer zone, and then proceeds to make its way to A_{sus}.

2. There was a path of infection from a node of A_{inf} to a node of A_{sus}.

We calculate the probability of the two events above happening during one epoch:

A.1.3.1 Case 1: One node was not cured when it entered the buffer zone, and then proceeds to make its way to A_{sus}.

The probability of this event is lower than the probability that one node was not cured during one epoch when it entered the buffer zone:

$$\begin{aligned}\mathbb{P}(\text{Case1}) &\leq (1-(1-e^{-(1+c_2)\log(N)\tau}))^{\frac{1}{\tau}} \\ &\leq e^{-(1+c_2)\log(N)} \\ &\leq \frac{1}{N^{1+c_2}}.\end{aligned}$$

A.1.3.2 Case 2: There was a path of infection from a node of A_{inf} to a node of A_{sus}.

In the case 2:

1. One node n_s of $A_{buf}^{t_0}$ needs to become infected at time step t_0.

2. One node n_e of $A_{\text{sus}}^{t_0+t}$ becomes infected after t time steps.

3. Every $\frac{1}{\tau}$ time steps, the nodes of A_{sus} can become closer to where the infected node by a distance 1. Therefore, $c\log(N) - \lfloor \tau \cdot t \rfloor - 1$ additional infections need to happen along the unique path between n_s and n_e.

Let us calculate the probability p_1 that b becomes infected at time t_0, and then proceed to infect $c\log(N) - \lfloor \tau \cdot t \rfloor$ additional nodes along a cured path in t time steps, with the head of the infection not being cured:

$$p_1 = \mu \cdot \binom{t}{\max(0, c\log(N) - \lfloor \tau \cdot t \rfloor - 1)} \mu^{c\log(N) - \lfloor \tau \cdot t \rfloor} (1-\beta)^t.$$

Let us now sum over all time steps t, to get the probability p_2 that an infection reaches A_{sus} with exactly d infections, starting from one time step:

$$
\begin{aligned}
p_2 &= \mu \cdot \sum_{t=\frac{c\log(N)}{1+\tau}}^{\infty} \binom{t}{\max(0, c\log(N) - \lfloor \tau \cdot t \rfloor - 1)} \mu^{c\log(N) - \lfloor \tau \cdot t \rfloor} (1-\beta)^t \\
&\leq \mu \cdot \sum_{t=\frac{c\log(N)}{1+\tau}}^{\infty} \binom{t}{c\log(N)-1} \frac{(c\log(N))!}{(c\log(N) - \lfloor \tau \cdot t \rfloor)!} \frac{t!}{(t+\tau t)!} \\
&\quad \cdot \mu^{c\log(N) - \lfloor \tau \cdot t \rfloor} (1-\beta)^t \\
&\leq \mu \cdot \sum_{t=\frac{c\log(N)}{1+\tau}}^{\infty} \binom{t}{c\log(N)-1} \left(\frac{c\log(N)}{t + \lfloor \tau t \rfloor} \right)^{\lfloor \tau t \rfloor} \mu^{c\log(N) - \lfloor \tau \cdot t \rfloor} (1-\beta)^t \\
&\leq \mu \cdot \sum_{t=c\log(N)}^{\infty} \binom{t}{c\log(N)-1} \mu^{c\log(N) - \tau \cdot t} (1-\beta)^t \\
&= \mu \cdot \sum_{t'=0}^{\infty} \binom{c\log(N) - 1 + t'}{c\log(N) - 1} \mu^{c\log(N)} \left(\frac{1-\beta}{\mu^\tau} \right)^{c\log(N) - 1 + t'} \\
&= \mu^{c\log(N)+1} \cdot \left(\frac{1-\beta}{\mu^\tau} \right)^{c\log(N)-1} \frac{1}{\left(1 - \left(\frac{1-\beta}{\mu^\tau}\right)\right)^{c\log(N)+1}} \\
&\sim_{\tau \to 0} \tau^{c\log(N)+1} \cdot 1^{c\log(N)-1} \frac{1}{\left(1 - \left(\frac{1-c_1 \cdot \tau}{\tau^\tau}\right)\right)^{c\log(N)}} + o(\tau) \\
&\sim_{\tau \to 0} \frac{\tau^{c\log(N)+1}}{(c_1\tau)^{c\log(N)}} + o(\tau) \\
&\sim_{\tau \to 0} \frac{\tau}{c_1^{c\log(N)}} + o(\tau) \\
&\sim_{\tau \to 0} \frac{\tau}{N^{c \cdot \log(c_1)}} + o(\tau),
\end{aligned}
$$

where we have used that $\left(\frac{c\log(N)}{t+\lfloor \tau t \rfloor}\right) < 1$, that $\mu < 1$, so $\mu^{-\lfloor \tau \cdot t \rfloor} \leq \mu^{-\tau \cdot t}$, that $\sum_{k=0}^{\infty} \binom{m+k}{k} a^k = \frac{1}{(1-a)^{m+1}}$ when $|a| < 1$, and that $\tau^\tau \to_{\tau \to 0} 1$.

Now, if we select a starting node n_s and an end node n_e, there is only one path between them in a tree. Such an infection can start $\frac{1}{\tau}$ times during

one epoch. We can therefore apply a union bound:

$$\begin{aligned}\mathbb{P}(\text{Case2}) &\leq \sum_{t_0=1}^{\frac{1}{\tau}} \sum_{n_s,n_e} p_2 \\ &\leq \frac{N^2}{\tau} \cdot p_2 \\ &\leq \frac{1}{N^{c\cdot\log(c_1)-2}}.\end{aligned}$$

A.1.4 Combining the results for all time steps

At each epoch, the probability of failure is upper bounded by $\mathbb{P}(\text{Case1})+\mathbb{P}(\text{Case2})$. The probability of failing during one iteration, which lasts $N + c\log(N)$ epochs, is therefore:

$$\begin{aligned}\mathbb{P}(\text{OneIterationFail}) &\leq (N + c\log(N)) \cdot (\mathbb{P}(\text{Case1}) + \mathbb{P}(\text{Case2})) \\ &\leq 2N \cdot (\frac{1}{N^{c\cdot\log(c_1)-2}} + \frac{1}{N^{1+c_2}}).\end{aligned}$$

Therefore, if we choose $c_2 = 1$, and $c_1 = e^{\frac{4}{c}}$, we have:

$$\begin{aligned}\mathbb{P}(\text{OneIterationFail}) &\leq 2N \cdot (\frac{1}{N^{4-2}} + \frac{1}{N^{1+1}}) \\ &\leq \frac{2}{N}.\end{aligned}$$

We have, therefore, an upper bound, as stated in Theorem 3.2.2. We repeat here, and complete the proof.

THEOREM 3.2.2. *In the Blind Curing setting, for all $c > 0$, we can cure the binary tree in expected linear time with budget $\mathcal{O}(e^{\frac{4}{c}} \cdot N^c)$.*

Proof. If we choose $c_2 = 1$, and $c_1 = e^{\frac{4}{c}}$, we have:

$$\begin{aligned}\mathbb{P}(\text{OneIterationFail}) &\leq 2N \cdot (\frac{1}{N^{4-2}} + \frac{1}{N^{1+1}}) \\ &\leq \frac{2}{N}.\end{aligned}$$

Therefore, with budget $e^{\frac{4}{c}} \cdot N^c + \log(N)$, for all $c > 0$:

$$\begin{aligned}\mathbb{E}[T_{\text{cured}}] &\leq \frac{N + c\log(N)}{1 - \frac{2}{N}} \\ &\leq 4N.\end{aligned}$$

A.2 Numerical experiments

In this section, we add some numerical experiments to illustrate the difficulty of the problem. We introduce two curing strategies: Naive Curing, which cures randomly a subset of nodes which signal themselves as infected (they raise a flag), and the strategy from Section A.1, Blind Protection, which prevents the infection from spreading by curing every node near the infected set. We hope these two strategies can provide insight into the difficulty of curing the complete binary tree in our model.

It is important to understand that the results we present are strategy-specific, which means better results could possibly be achieved with better strategies. Devising optimal strategies is however outside of the scope of this work.

A.2.1 Impact of the lack of information

In this section, we illustrate the dramatic impact of the lack of information on the Naive Curing strategy. In the following experiment, we consider a binary tree on 31 nodes. We use a budget $r = 16 > \frac{N}{2}$. If p_ϵ is the probability of error, at each time step, an infected node raises a flag with probability $1-p_\epsilon$, and a susceptible node raises a flag with probability p_ϵ. We set the size of a time step to be $\tau = 0.1$.

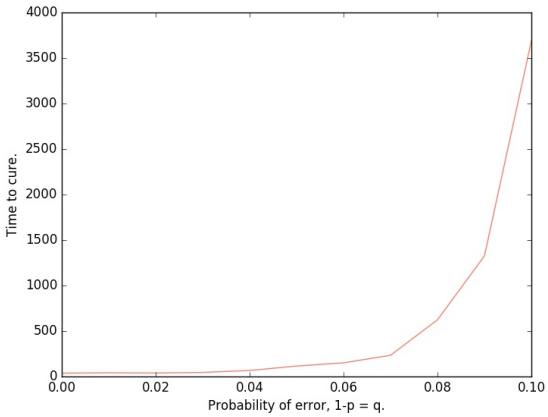


Figure A.1: Time to cure as a function of the probability of error for the Naive Curing strategy.

The results can be seen in Figure A.1. The time to cure increases faster than exponentially with the probability of error. We can see that even with 10% of error, it takes more than 3500 time steps with budget $r = \frac{N}{2}$ on 31 nodes.

A.2.2 Impact of size of the graph

We now consider the strategy described in Appendix A.1. For the purpose of these experiments, keeping the same notation as the previous section, we set $c_1 = 10$ (this is the budget for every node which we "protect"), and $c_2 = 1$ (we cure any new node with budget $(1 + c_2)\log(N)$). We still have $\tau = 0.1$. For this experiment, we investigate the time it takes to cure the graph for a budget equal to different exponents of the number of nodes.

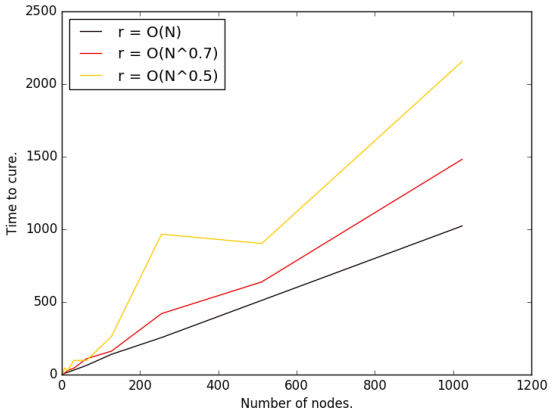


Figure A.2: Time to cure as a function of the number of nodes for the Blind Protection strategy. The plots are the average of 20 runs.

The results are shown in Figure A.2. As theory predicts, the time to cure increases more slowly than $4 \cdot N$, where N is the number of nodes for budget $r = \mathcal{O}(N^c)$, for all $c > 0$ constant.

As a reminder, in the Blind Curing setting, it is impossible to cure the complete binary tree in less than superpolynomial time for budget $r = \mathcal{O}(\log^\alpha(N))$, for all $\alpha > 0$ constant.

Appendix B

Uncertainty about when nodes are infected

B.1 Bidirectional tree

We include here the full calculations for learning the weights of the bidirectional tree.

Proposition B.1.1. *If we know (i, j) is an edge in the original tree, then the probability of infection along this edge is given by:*

$$p_{ij} = \frac{f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}{g_{i,\not j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}.$$

Proof. According to Lemma 4.2.8, we have:

$$\begin{aligned}
f_{i<j} &= \mathcal{P}_{\not j}(\to i) \cdot p_{ij} \cdot s_0 + \mathcal{P}_{\not i}(\to j) \cdot p_{ji} \cdot s_2 \\
g_{i,\not j} &= \mathcal{P}_{\not j}(\to i) \cdot (1 - p_{ij}) \\
f_{j<i} &= \mathcal{P}_{\not i}(\to j) \cdot p_{ji} \cdot s_0 + \mathcal{P}_{\not j}(\to i) \cdot p_{ij} \cdot s_2 \\
g_{j,\not i} &= \mathcal{P}_{\not i}(\to j) \cdot (1 - p_{ji}).
\end{aligned}$$

We have 4 second-order equations, with 4 unknowns: p_{ij}, p_{ji}, $\mathcal{P}_{\not j}(\to i)$ and

$\mathcal{P}_{\not i}(\to j)$. We solve it:

$$\begin{aligned}
f_{i<j} + f_{j<i} &= (\mathcal{P}_{\not j}(\to i) \cdot p_{ij} + \mathcal{P}_{\not i}(\to j) \cdot p_{ji}) \cdot (s_0 + s_2) \\
f_{i<j} - f_{j<i} &= (\mathcal{P}_{\not j}(\to i) \cdot p_{ij} - \mathcal{P}_{\not i}(\to j) \cdot p_{ji}) \cdot (s_0 - s_2) \\
\implies \mathcal{P}_{\not j}(\to i) \cdot p_{ij} &= \frac{1}{2}\left(\frac{f_{i<j} + f_{j<i}}{s_0 + s_2} + \frac{f_{i<j} - f_{j<i}}{s_0 - s_2}\right) \\
&= \frac{f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}{s_0^2 - s_2^2} \\
\mathcal{P}_{\not j}(\to i) &= g_{i,\not j} + \mathcal{P}_{\not j}(\to i) \cdot p_{ij} \\
&= g_{i,\not j} + \frac{f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}{s_0^2 - s_2^2} \\
\implies p_{ij} &= \frac{\mathcal{P}_{\not j}(\to i) \cdot p_{ij}}{\mathcal{P}_{\not j}(\to i)} \\
p_{ij} &= \frac{f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}{g_{i,\not j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2}.
\end{aligned}$$

□

Lemma B.1.2. *With* $M = \frac{N^2}{\epsilon^2} \log\left(\frac{6}{\delta}\right) \frac{\left((s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2\right)^2}{(s_0^2 - s_2^2)^2}$ *samples, with probability at least* $1 - \delta$*, we have:*

$$|\hat{p}_{ij} - p_{ij}| \leq \epsilon.$$

Proof. Using Hoeffding's inequality:

$$\begin{aligned}
\mathbb{P}(|\hat{f}_{i<j} - f_{i<j}| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}, \\
\mathbb{P}(|\hat{f}_{j<i} - f_{j<i}| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}, \\
\mathbb{P}(|\hat{g}_{i,\not j} - g_{i,\not j}| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}.
\end{aligned}$$

Choosing $M = \frac{1}{\epsilon_1^2}\log\left(\frac{6}{\delta}\right)$, we have that with probability at least $1-\delta$, all the following hold:

$$|\hat{f}_{i<j} - f_{i<j}| \leq \epsilon_1,$$

$$|\hat{f}_{j<i} - f_{j<i}| \leq \epsilon_1,$$

$$|\hat{g}_{i,\not j} - g_{i,\not j}| \leq \epsilon_1.$$

Hence, with probability at least $1-\delta$, we have:

$$\begin{aligned}
\hat{p}_{ij} &= \frac{\hat{f}_{i<j}\cdot s_0 - \hat{f}_{j<i}\cdot s_2}{\hat{g}_{i,\not j}\cdot(s_0^2 - s_2^2) + \hat{f}_{i<j}\cdot s_0 - \hat{f}_{j<i}\cdot s_2} \\
&\leq \frac{(f_{i<j}+\epsilon_1)\cdot s_0 - (f_{j<i}-\epsilon_1)\cdot s_2}{(g_{i,\not j}-\epsilon_1)\cdot(s_0^2 - s_2^2) + (f_{i<j}-\epsilon_1)\cdot s_0 - (f_{j<i}+\epsilon_1)\cdot s_2} \\
&= \frac{f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2 + \epsilon_1(s_0+s_2)}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2 - \epsilon_1(s_0^2 - s_2^2 + s_0 + s_2)} \\
&= p_{ij}\frac{1}{1 - \frac{\epsilon_1(s_0^2 - s_2^2 + s_0 + s_2)}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2}} \\
&\quad + \epsilon_1\frac{s_0+s_2}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2} + o(\epsilon_1) \\
&= p_{ij}\left(1 + \frac{\epsilon_1(s_0^2 - s_2^2 + s_0 + s_2)}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2}\right) \\
&\quad + \epsilon_1\frac{s_0+s_2}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2} + o(\epsilon_1) \\
\hat{p}_{ij} - p_{ij} &\leq \epsilon_1\frac{(s_0^2 - s_2^2 + s_0 + s_2)p_{max}}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2} \\
&\quad + \epsilon_1\frac{s_0+s_2}{g_{i,\not j}\cdot(s_0^2 - s_2^2) + f_{i<j}\cdot s_0 - f_{j<i}\cdot s_2} + o(\epsilon_1).
\end{aligned}$$

Using the results from Lemma 4.2.8, we have:

$$f_{i<j} = \mathcal{P}_{\not j}(\to i)\cdot p_{ij}\cdot s_0 + \mathcal{P}_{\not i}(\to j)\cdot p_{ji}\cdot s_2,$$

$$g_{i,\not j} = \mathcal{P}_{\not j}(\to i)\cdot(1-p_{ij}).$$

We use it to simplify the denominator:

$$\begin{aligned}
\text{denominator} &= g_{i,\not j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2 \\
&= \left(\mathcal{P}_{\not j}(\to i) \cdot (1 - p_{ij})\right) \cdot (s_0^2 - s_2^2) \\
&\quad + \left(\mathcal{P}_{\not j}(\to i) \cdot p_{ij} \cdot s_0 + \mathcal{P}_{\not i}(\to j) \cdot p_{ji} \cdot s_2\right) \cdot s_0 \\
&\quad - \left(\mathcal{P}_{\not i}(\to j) \cdot p_{ji} \cdot s_0 + \mathcal{P}_{\not j}(\to i) \cdot p_{ij} \cdot s_2\right) s_2 \\
&= \mathcal{P}_{\not j}(\to i) \cdot (s_0^2 - s_2^2) \\
&\geq \frac{s_0^2 - s_2^2}{N}.
\end{aligned}$$

Plugging back above:

$$\begin{aligned}
\hat{p}_{ij} - p_{ij} &\leq \epsilon_1 \frac{(s_0^2 - s_2^2 + s_0 + s_2)p_{max}}{g_{i,\not j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2} \\
&\quad + \epsilon_1 \frac{s_0 + s_2}{g_{i,\not j} \cdot (s_0^2 - s_2^2) + f_{i<j} \cdot s_0 - f_{j<i} \cdot s_2} + o(\epsilon_1) \\
&\leq \epsilon_1 N \frac{(s_0^2 - s_2^2 + s_0 + s_2)p_{max}}{s_0^2 - s_2^2} + \epsilon_1 N \frac{s_0 + s_2}{s_0^2 - s_2^2} + o(\epsilon_1).
\end{aligned}$$

By symmetry, we obtain:

$$|\hat{p}_{ij} - p_{ij}| \leq \epsilon_1 N \frac{(s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2}{s_0^2 - s_2^2} + o(\epsilon_1).$$

By choosing $\epsilon_1 = \frac{\epsilon}{N} \frac{s_0^2 - s_2^2}{(s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2}$, we therefore have:
With $M = \frac{N^2}{\epsilon^2} \log\left(\frac{6}{\delta}\right) \frac{\left((s_0^2 - s_2^2 + s_0 + s_2)p_{max} + s_0 + s_2\right)^2}{(s_0^2 - s_2^2)^2}$ samples, with probability at least $1 - \delta$, we have $|\hat{p}_{ij} - p_{ij}| \leq \epsilon$.

□

B.2 Bounded-degree graphs

B.2.1 Solving the system

Lemma B.2.1. *Let* $\Delta = (s_0^2 - s_2^2)^2 - 4(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)$. *We have:*

$$\Delta \geq (s_0^2 - s_2^2)^2 \frac{(1 - p_{max})^2}{1 + p_{max}^2}.$$

Proof. Finding a lower bound for Δ can be achieved through minimizing Δ, or maximizing $(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)$. We want to solve:

$$\begin{aligned}
\underset{V_{ij}, V_{ji}}{\text{maximize}} \quad & (V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0) \\
\text{subject to} \quad & V_{ij} = \frac{p_{ij}s_0 + p_{ji}s_2}{1 + p_{ij}p_{ji}}, \\
& V_{ji} = \frac{p_{ji}s_0 + p_{ij}s_2}{1 + p_{ij}p_{ji}}, \\
& p_{ij} \geq 0, \\
& p_{max} - p_{ij} \geq 0, \\
& p_{ji} \geq 0, \\
& p_{max} - p_{ji} \geq 0.
\end{aligned}$$

To do so, we introduce Lagrangian multipliers. By replacing V_{ij} and V_{ji} with their actual value, the optimization problem above only has affine constraints, so it satisfies the linearity constraint qualification for the Karush-Kuhn-Tucker conditions. In other words, all the partial derivatives of the Lagrangian are

equal to 0 for an optimal point.

$$\begin{aligned}
\mathcal{L} &= \mathcal{L}(V_{ij}, V_{ji}, p_{ij}, p_{ji}, \lambda_1, \lambda_2, \mu_1, \mu_2, \mu_3, \mu_4) \\
&= (V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0) \\
&\quad - \lambda_1(v_{ij}(1 + p_{ij}p_{ji}) - p_{ij}s_0 + p_{ji}s_2) \\
&\quad - \lambda_2(v_{ji}(1 + p_{ij}p_{ji}) - p_{ji}s_0 + p_{ij}s_2) \\
&\quad - \mu_1 p_{ij} - \mu_2(p_{max} - p_{ij}) \\
&\quad - \mu_3 p_{ji} - \mu_4(p_{max} - p_{ji}).
\end{aligned}$$

We calculate the gradients of $\mathcal{L}$.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_{ij}} &= V_{ji}(s_0^2 + s_2^2) - 2V_{ij}s_0s_2 - \lambda_1(1 + p_{ij}p_{ji}), \\
\frac{\partial \mathcal{L}}{\partial V_{ji}} &= V_{ij}(s_0^2 + s_2^2) - 2V_{ji}s_0s_2 - \lambda_2(1 + p_{ij}p_{ji}), \\
\frac{\partial \mathcal{L}}{\partial p_{ij}} &= -\lambda_1(V_{ij}p_{ji} - s_0) - \lambda_2(V_{ji}p_{ji} - s_2) - \mu_1 + \mu_2, \\
\frac{\partial \mathcal{L}}{\partial p_{ji}} &= -\lambda_1(V_{ij}p_{ij} - s_2) - \lambda_2(V_{ji}p_{ij} - s_0) - \mu_3 + \mu_4.
\end{aligned}$$

From now on, we find the set X^0 of points for which all the partial derivatives are null. We know the solution of the maximization problem is the point of X^0 which maximizes the objective function.

Let us assume an interior point solution exists. For this point, all the gradients of $\mathcal{L}$ are equal to 0. Since it is an interior point, we also have $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 0$ by complementary slackness. Solving this system, we

obtain:

$$\lambda_1 = (s_0^2 - s_2^2)\frac{p_{ji}s_0 - p_{ij}s_2}{(1 + p_{ij}p_{ji})^2},$$
$$\lambda_2 = (s_0^2 - s_2^2)\frac{p_{ij}s_0 - p_{ji}s_2}{(1 + p_{ij}p_{ji})^2}.$$

Plugging this in above, the condition $\frac{\partial \mathcal{L}}{\partial p_{ij}} = 0$ becomes $p_{ij}p_{ji}(1 - p_{ji}) = 0$. However, this is impossible for an interior point, since $0 < p_{ij}, p_{ji} < p_{max} < 1$. Therefore, the extrema of Δ are attained when at least one constraint is active.

We notice that if the conditions $p_{ij} = 0$ or $p_{ji} = 0$ are active, then $(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0) = 0$. Let us suppose (without loss of generality, by symmetry of the problem) that we have $p_{ij} = p_{max}$. The objective function is then increasing in p_{ji}. Therefore, Δ is minimized when $p_{ij} = p_{ji} = p_{max}$, which implies $v_{ij} = V_{ji} = \frac{p_{max}(s_0+s_2)}{1+p_{max}^2}$. In this case:

$$\begin{aligned}
\Delta &\geq (s_0^2 - s_2^2)^2 - 4V_{ij}^2(s_0 - s_2)^2 \\
&\geq (s_0^2 - s_2^2)^2 - 4\left(\frac{p_{max}(s_0 + s_2)}{1 + p_{max}^2}\right)^2 (s_0 - s_2)^2 \\
&\geq (s_0^2 - s_2^2)^2 \left[1 - 4\frac{p_{max}}{1 + p_{max}^2}\right] \\
&\geq (s_0^2 - s_2^2)^2 \frac{(1 - p_{max})^2}{1 + p_{max}^2}.
\end{aligned}$$

This expression is always positive, which is what we wanted.

□

Theorem B.2.2. *For any graph, for any noise distribution having finite values, we can learn the weights of all the edges of the graph. In particular, we*

can compute a quantity which converges to the true weight of each edge:

$$\hat{p}_{ij} = \frac{2(\hat{V}_{ji}s_2 - \hat{V}_{ij}s_0)}{(s_0^2 - s_2^2) + \sqrt{(s_0^2 - s_2^2)^2 - 4(\hat{V}_{ji}s_2 - \hat{V}_{ij}s_0)(\hat{V}_{ij}s_2 - \hat{V}_{ji}s_0)}}.$$

Proof.

$$\begin{aligned}
\hat{V}_{ij} &\to_{M\to\infty} V_{ij} \\
&:= \frac{p_{ij}\cdot s_0 + p_{ji}\cdot s_2}{1 + p_{ij}\cdot p_{ji}} \\
p_{ji} &= \frac{V_{ij} - p_{ij}\cdot s_0}{s_2 - V_{ij}\cdot p_{ij}}
\end{aligned}$$

We can plug this in V_{ji}:

$$\begin{aligned}
V_{ji} &= \frac{p_{ji}\cdot s_0 + p_{ij}\cdot s_2}{1 + p_{ij}\cdot p_{ji}} \\
V_{ji}\left[1 + p_{ij}\cdot\frac{V_{ij} - p_{ij}\cdot s_0}{s_2 - V_{ij}\cdot p_{ij}}\right] &= \frac{V_{ij} - p_{ij}\cdot s_0}{s_2 - V_{ij}\cdot p_{ij}}\cdot s_0 + p_{ij}\cdot s_2
\end{aligned}$$

After some shuffling around, we obtain the second-degree equation:

$$V_{ji}s_2 - V_{ij}s_0 + \left(s_0^2 - s_2^2\right)p_{ij} + \left(V_{ij}s_2 - V_{ji}s_0\right)p_{ij}^2 = 0$$

We recall that by definition, $s_0 \geq s_2$. We also notice that if $p_{ij} = q_1$ and $p_{ji} = q_2$ is a pair of solutions of this system, then $p_{ij} = \frac{1}{q_2}$ and $p_{ji} = \frac{1}{q_1}$ forms the other pair of solution, which implies there is uniqueness of solutions in $[0, p_{max}]$. Since the real probabilities of infection satisfy this system, we also know the solution exists. Let $\Delta = (s_0^2 - s_2^2)^2 - 4(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)$.

The only solution of this system in $[0, p_{max}]$ is:

$$\begin{aligned} p_{ij} &= \frac{-(s_0^2 - s_2^2) + \sqrt{\Delta}}{2(V_{ij}s_2 - V_{ji}s_0)} \\ &= \frac{(s_0^2 - s_2^2)^2 - (s_0^2 - s_2^2)^2 - 4(V_{ji}s_2 - V_{ij}s_0)(V_{ij}s_2 - V_{ji}s_0)}{2(V_{ji}s_0 - V_{ij}s_2)\left((s_0^2 - s_2^2) + \sqrt{\Delta}\right)} \\ &= \frac{2(V_{ji}s_2 - V_{ij}s_0)}{(s_0^2 - s_2^2) + \sqrt{\Delta}} \end{aligned}$$

□

B.2.2 Sample complexity

Proposition B.2.3. *As the number of cascades M goes to infinity, the estimators below tend to the following limit:*

$$\begin{aligned} \hat{h}^2_{i,j} &\to_{M\to\infty} \frac{1}{N}(p_{ij} + p_{ji}) \prod_{k\neq i,j} (1 - p_{ik})(1 - p_{jk}) \\ \hat{f}^2_{i<j} &\to_{M\to\infty} \frac{1}{N}(p_{ij} \cdot s_0 + p_{ji} \cdot s_2) \prod_{k\neq i,j} (1 - p_{ik})(1 - p_{jk}) \\ \hat{e}^1_i &\to_{M\to\infty} \frac{1}{N}(1 - p_{ij}) \prod_{k\neq i,j} (1 - p_{ik}) \end{aligned}$$

Proof. Using the law of large numbers:

$$\begin{aligned}
\hat{f}^2_{i<j} &\to_{M\to\infty} \mathbb{E}[\hat{f}^2_{i<j}] \\
&= \mathbb{P}(i \text{ source, infects } j, \text{ no other infections, delay } 0) \\
&\quad + \mathbb{P}(j \text{ source, infects } i, \text{ no other infections, delay } 2) \\
&= \frac{1}{N} p_{ij} \prod_{k\neq i,j} (1-p_{ik}) \prod_{k\neq i,j} (1-p_{jk}) \cdot s_0 \\
&\quad + \frac{1}{N} p_{ji} \prod_{k\neq i,j} (1-p_{jk}) \prod_{k\neq i,j} (1-p_{ik}) \cdot s_2 \\
&= \frac{1}{N} (p_{ij} \cdot s_0 + p_{ji} \cdot s_2) \prod_{k\neq i,j} (1-p_{ik})(1-p_{jk}).
\end{aligned}$$

In the same vein, we have:

$$\begin{aligned}
\hat{h}^2_{i,j} &\to_{M\to\infty} \mathbb{E}[\hat{h}^2_{i,j}] \\
&= \mathbb{P}(i \text{ source, infects } j, \text{ no other infections}) \\
&\quad + \mathbb{P}(j \text{ source, infects } i, \text{ no other infections}) \\
&= \frac{1}{N} (p_{ij} + p_{ji}) \prod_{k\neq i,j} (1-p_{ik})(1-p_{jk}) \\
\hat{e}^1_i &\to_{M\to\infty} \mathbb{E}[\hat{e}^1_i] \\
&= \mathbb{P}(i \text{ source, no other infections}) \\
&= \frac{1}{N} \prod_{k\neq i} (1-p_{ik}) \\
&= \frac{1}{N} (1-p_{ij}) \prod_{k\neq i,j} (1-p_{ik}).
\end{aligned}$$

□

Proposition B.2.4. *With probability* $1 - \frac{\delta}{N^2}$*, with* $M =$ *samples, we can estimate* V_{ij} *with precision* ϵ_V*.*

Proof. As in Proposition 4.2.11, we use Hoeffding's inequality:

$$\begin{aligned}
\mathbb{P}(|\hat{f}^2_{i<j} - f^2_{i<j}| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}, \\
\mathbb{P}(|\hat{h}^2_{i,j} - h^2_{i,j}| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}, \\
\mathbb{P}(|\hat{e}^1_i - e^1_i| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}, \\
\mathbb{P}(|\hat{e}^1_j - e^1_j| > \epsilon_1) &\leq 2e^{-2M\epsilon_1^2}.
\end{aligned}$$

We use this to bound above V_{ij}:

$$\begin{aligned}
\hat{V}_{ij} &= \frac{\hat{f}^2_{i<j}}{\hat{h}^2_{i,j} + N \cdot \hat{e}^1_i \cdot \hat{e}^1_j} \\
&\leq \frac{f^2_{i<j} + \epsilon_1}{h^2_{i,j} + \epsilon_1 + N \cdot (e^1_i + \epsilon_1) \cdot (e^1_j + \epsilon_1)} \\
&= V_{ij} \frac{1 + \frac{\epsilon_1}{f^2_{i<j}}}{1 + \frac{\epsilon_1 + N\epsilon_1(e^1_i + e^1_j)}{h^2_{i,j} + Ne^1_i e^1_j}} \\
&= V_{ij} \left[1 + \frac{\epsilon_1}{f^2_{i<j}} + \epsilon_1 \frac{1 + N(e^1_i + e^1_j)}{h^2_{i,j} + Ne^1_i e^1_j} + o(\epsilon_1) \right]
\end{aligned}$$

We bound below the denominators:

$$\begin{aligned}
f^2_{i<j} &\geq \frac{1}{N}(p_{ij}s_0 + p_{ji}s_2)(1 - p_{max})^{2d} \\
&\geq \frac{p_{min}s_2}{N}(1 - p_{max})^{2d} \\
h^2_{i,j} + Ne^1_i e^1_j &\geq \frac{1}{N}(1 + p_{ij}p_{ji})(1 - p_{max})^{2d} \\
&\geq \frac{1}{N}(1 - p_{max})^{2d}
\end{aligned}$$

We bound above the numerator:

$$\begin{aligned} N(e_i^1 + e_i^1) &= N\left(\frac{1}{N}\prod_{k\neq i}(1-p_{ik}) + \frac{1}{N}\prod_{k\neq j}(1-p_{jk})\right) \\ &\leq N\left(\frac{1}{N} + \frac{1}{N}\right) \\ &\leq 2. \end{aligned}$$

Plugging in above:

$$\begin{aligned} \hat{V}_{ij} &= V_{ij}\left[1 + \frac{\epsilon_1}{f_{i<j}^2} + \epsilon_1\frac{1 + N(e_i^1 + e_j^1)}{h_{i,j}^2 + Ne_i^1e_j^1} + o(\epsilon_1)\right] \\ &\leq V_{ij}\left[1 + \frac{\epsilon_1}{\frac{p_{min}s_2}{N}(1-p_{max})^{2d}} + \frac{\epsilon_1(1+2)}{\frac{1}{N}(1-p_{max})^{2d}} + o(\epsilon_1)\right]. \end{aligned}$$

Using $V_{ij} \leq 1$, and by symmetry:

$$\begin{aligned} |\hat{V}_{ij} - V_{ij}| &= \epsilon_1\frac{N}{(1-p_{max})^{2d}}\left[\frac{1}{p_{min}s_2} + 3\right] + o(\epsilon_1) \\ &=\leq \epsilon_1\frac{N}{(1-p_{max})^{2d}}\left[\frac{1 + 3p_{min}s_2}{p_{min}s_2}\right] + o(\epsilon_1) \\ &\leq \epsilon_1\frac{4N}{p_{min}s_2(1-p_{max})^{2d}} + o(\epsilon_1). \end{aligned}$$

Therefore, by union bound, and by choosing $\epsilon_1\frac{4N}{p_{min}s_2(1-p_{max})^{2d}} = \epsilon_V$, and setting $2e^{-2M\epsilon_1^2} = \frac{\delta}{3N^2}$, we obtain:

With $M = \frac{1}{\epsilon_V^2}\frac{16N^2}{p_{min}^2s_2^2(1-p_{max})^{4d}}\frac{2\log(3N)-\log(\delta)}{2}$ samples, we can guarantee $|\hat{V}_{ij} - V_{ij}| \leq \epsilon_V$ with probability at least $1 - \frac{\delta}{N^2}$. □

Proposition B.2.5. *Assuming we can estimate V_{ij} within precision ϵ_V, then we can estimate p_{ij} within precision $\epsilon = \frac{6\epsilon_V(1+p_{max}^2)}{(s_0^2-s_2^2)^2(1-p_{max})^2}$.*

Proof. If we know $|\hat{V}_{ij} - V_{ij}| < \epsilon_V$ and $|\hat{V}_{ji} - V_{ji}| < \epsilon_V$:

$$\begin{aligned}\hat{p}_{ij} &= \frac{2(\hat{V}_{ji}s_2 - \hat{V}_{ij}s_0)}{(s_0^2 - s_2^2) + \sqrt{(s_0^2 - s_2^2)^2 - 4(\hat{V}_{ji}s_2 - \hat{V}_{ij}s_0)(\hat{V}_{ij}s_2 - \hat{V}_{ji}s_0)}} \\ &\leq \frac{2(V_{ji}s_2 - V_{ij}s_0) + 2\epsilon_V(s_0 + s_2)}{(s_0^2 - s_2^2)^2 + \sqrt{\Delta - 4\epsilon_V(s_0 + s_2)^2(V_{ij} + V_{ji})}}.\end{aligned}$$

We recall $s_0 + s_2 \leq 1$, $V_{ij} \leq 1$, $p_{ij} \leq 1$, and $1 + p_{max}^2 \leq \Delta \leq 1$ (Lemma 4.3.10). Hence:

$$\begin{aligned}\hat{p}_{ij} &\leq \frac{2(V_{ji}s_2 - V_{ij}s_0) + 2\epsilon_V}{(s_0^2 - s_2^2)^2 + \sqrt{\Delta}\sqrt{1 - 8\frac{\epsilon_V}{\Delta}}} \\ &\leq p_{ij}\left(1 + \frac{4\epsilon_V}{\sqrt{\Delta}\left((s_0^2 - s_2^2)^2 + \sqrt{\Delta}\right)}\right) + \frac{2\epsilon_V}{(s_0^2 - s_2^2)^2 + \sqrt{\Delta}} + o(\epsilon_V) \\ &\leq p_{ij} + \frac{6\epsilon_V}{\Delta}.\end{aligned}$$

By symmetry, and using the bound on Δ stated above, we conclude that if we know V_{ij} and V_{ji} up to precision ϵ_V, we know p_{ij} up to precision $\epsilon = \frac{6\epsilon_V(1+p_{max}^2)}{(s_0^2 - s_2^2)^2(1-p_{max})^2}$.

□

Theorem B.2.6. *In the limited-noise setting, with probability at least $1 - \delta$, with $M = \mathcal{O}\left(\frac{e^{4p_{max}(d+1)}}{p_{min}^2 s_2^2 (s_0^2 - s_2^2)^4} \frac{N^2}{\epsilon^2} \log\left(\frac{N}{\delta}\right)\right)$ samples, we can learn the weights of any bounded-degree graph up to precision epsilon.*

Proof. With probability at least $1 - \frac{\delta}{N^2}$, using
$M = \frac{1}{\epsilon_V^2} \frac{16N^2}{p_{min}^2 s_2^2 (1-p_{max})^{4d}} \frac{2\log(3N) - \log(\delta)}{2}$ samples, we can guarantee $|\hat{V}_{ij} - V_{ij}| \leq \epsilon_V$

with probability at least $1 - \frac{\delta}{N^2}$ samples, knowing $\frac{1}{\epsilon_V} = \frac{6(1+p_{max}^2)}{\epsilon(s_0^2-s_2^2)^2(1-p_{max})^2}$. This gives us a sample complexity of:

$$\begin{aligned} M &\geq \frac{18}{\epsilon^2(s_0^2 - s_2^2)^4(1-p_{max})^{4(d+1)}} N^2 \left[\frac{4}{p_{min}s_2}\right]^2 \log\left(\frac{9N^2}{\delta}\right) \\ &\geq \frac{1152 \cdot e^{4p_{max}(d+1)}}{p_{min}^2 s_2^2 (s_0^2 - s_2^2)^4} \frac{N^2}{\epsilon^2} \log\left(\frac{9N^2}{\delta}\right) \\ &= \mathcal{O}\left(\frac{e^{4p_{max}(d+1)}}{p_{min}^2 s_2^2 (s_0^2 - s_2^2)^4} \frac{N^2}{\epsilon^2} \log\left(\frac{N}{\delta}\right)\right). \end{aligned}$$

□

Appendix C

Uncertainty about what is infecting nodes

C.1 Necessary Conditions

C.1.1 We need at least three edges

Let $G = (V, E_1 \cup E_2)$ be the union of the graphs from both mixtures. In this subsection, we prove it is impossible to learn the weights of E_1 and E_2 if G has less than three edges:

One edge: For a graph on two nodes, we have already seen that the cascade distribution are identical if $p_{12} = \beta = 1 - q_{12}$, for any value of β, which proves the problem is not solvable.

Two edges: When we have two nodes and two edges, we can without loss of generality assume that node 1 is connected to node 2 and node 3. Then, if:

- $p_{12} = \beta$
- $q_{12} = 1 - \beta$
- $p_{13} = \frac{\frac{1}{2} - \frac{\beta}{2} + \frac{1}{4}}{\frac{1}{2} - \beta}$
- $q_{13} = \frac{\frac{1}{4} - \frac{\beta}{2}}{\frac{1}{2} - \beta}$

The cascade distribution is identical for any value of $\beta < \frac{1}{2}$. By simple calculations, we can show the following,

- Fraction of cascades with only node 1 infected: $\frac{1}{12}$.
- Fraction of cascades with only node 2 infected: $\frac{1}{6}$.
- Fraction of cascades with only node 3 infected: $\frac{1}{6}$.
- Fraction of cascades where 3 infected 1, but 1 did not infect 2: $\frac{1}{12}$.
- Fraction of cascades where 3 infected 1, 1 infected 2: $\frac{1}{12}$.
- Fraction of cascades where 1 infected 3, but 1 did not infect 2: $\frac{1}{12}$.
- Fraction of cascades where 1 infected 2, but 1 did not infect 3: $\frac{1}{12}$.
- Fraction of cascades where 1 infected 3 and 2: $\frac{1}{12}$.
- Fraction of cascades where 2 infected 1, but 1 did not infect 3: $\frac{1}{12}$.
- Fraction of cascades where 2 infected 1, then 1 infected 3: $\frac{1}{12}$.

Since the distribution of cascades is the same for any value of $\beta < \frac{1}{2}$, the problem is not solvable.

C.1.2 We need Δ-separation

Separability is necessary for the existence of sample efficient algorithms. Specifically, we show that there exist (many) graphs where separability is

violated, and for which the sample complexity is exponential in the size of the graph.

Indeed, consider a graph G composed of two subgraphs A and B, connected by a path P of length d. Suppose the path has the same weight in both mixtures, and for the edges $e \in P$, $\max_{e \in P} p_e < 1$. Similar to the disconnected graph, we write $A_i = A \cap E_i$, and $B_i = B \cap E_i$. To learn the graph completely we need to differentiate between the mixture on E_1 and E_2, and the mixture on $E_1' = A_1 \cup P \cup B_2$ and $E_2' = A_2 \cup P \cup B_1$.

The path P is not informative in the above differentiation as both the mixture in the path have same weights. Therefore, we need at least one cascade covering *at least one edge in A and one edge in B*. Since P is of length d, this happens with probability at most $e^{-\Omega(d)}$. To see such a cascade, we need at least $e^{\Omega(d)}$ cascades in expectation. Therefore, setting $d = cN$, for some constant $c > 0$, we prove that exponential number of samples are necessary for any algorithm to recover the graph if the Δ-separated Condition is violated.

C.1.3 Dealing with mixtures which are not Δ-separated

In this section, we show how to detect and deduce the weights of edges which have the same weight across both component of the mixture. We assume both G_1 and G_2 follow Conditions 1 and 2 if we remove all non-distinct edges, and in particular remain connected.

Suppose there exists an edge (i, j) in the graph, such that $p_{ij} = q_{ij} > 0$. Then in particular, there exists another edge connecting i to the rest of the

graph G_1 through node k, such that $p_{ik} \neq q_{ik}$. Then:

Lemma C.1.1. *Suppose G_1 and G_2 follow assumption 2 after removing all non-distinct edges. We can detect and learn the weights of non-distinct edges the following way:*

$$\textit{If } X_{ij} > 0, \textit{ and } \forall k \in V,\ X_{ik} > 0 \implies Y_{ik,ij} - X_{ik}X_{ij} = 0, \textit{ then}$$

$p_{ij} = q_{ij} = X_{ij}$.

Proof. Since G_1 is connected on three nodes or more even when removing edge (i, j), we know there exists a node l such either l is connected to either i or k. Therefore, either $Y_{ik,il} - X_{ik}X_{il} > 0$ or $Y_{ki,kl} - X_{ki}X_{kl} > 0$. In both these cases, we deduce $p_{ik} \neq q_{ik}$. This in turns allow us to detect that $p_{ij} = q_{ij}$. Once this edge is detected, it is very easy to deduce its weight, since $p_{ij} = X_{ij} = q_{ij}$ by definition. □

C.2 Proofs for unbalanced mixtures

C.2.1 Estimators - proofs

Lemma C.2.1. *Under Conditions 1 and 2, in the setting of infinite samples, the weights of the edges for a line structure are then given by:*

$$p_{ua} = X_{ua} + s_{ua}\sqrt{\frac{(Y^{|}_{ua,ub} - X_{ua}X_{ub})R^{|}}{Y^{|}_{ub,bc} - X_{ua}X_{bc}}},\ q_{ua} = X_{ua} - s_{ua}\sqrt{\frac{(Y^{|}_{ua,ub} - X_{ua}X_{ub})R^{|}}{Y^{|}_{ub,bc} - X_{ua}X_{bc}}},$$

$$p_{bc} = X_{uc} + s_{bc}\sqrt{\frac{(Y^{|}_{ub,bc} - X_{ua}X_{bc})R^{|}}{Y^{|}_{ua,ub} - X_{uc}X_{ua}}},\ q_{bc} = X_{uc} - s_{bc}\sqrt{\frac{(Y^{|}_{ub,bc} - X_{ua}X_{bc})R^{|}}{Y^{|}_{ua,ub} - X_{uc}X_{ua}}}$$

$$p_{ub} = X_{ub} + s_{ub}\sqrt{\frac{(Y^{|}_{uc,ua} - X_{uc}X_{ua})(Y^{|}_{ub,bc} - X_{ua}X_{bc})}{R^{|}}},$$

$$q_{ua} = X_{ua} - s_{ua}\sqrt{\frac{(Y^{|}_{ua,ub} - X_{uc}X_{ua})(Y^{|}_{ub,bc} - X_{ua}X_{bc})}{R^{|}}},$$

where $R^{|} = X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}$, *and for* $s_{ua} \in \{-1, 1\}$.

Proof. In this case, there is no edge between u and c, which implies that $p_{uc} = q_{uc} = 0$. Hence, we cannot use a variation of the equation above for finding the edges of a star structure without dividing by zero. Therefore, we need to use $Z^{|}_{ua,ub,bc}$. Let $R^{|} = X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}$. We notice

a remarkable simplification:

$$
\begin{aligned}
R^{|} &= X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}} \\
&= \frac{p_{ua}+q_{ua}}{2}\cdot\frac{p_{bc}+q_{bc}}{2} + \frac{\frac{p_{ua}p_{ub}p_{bc}+q_{ua}q_{ub}q_{bc}}{2} - \frac{p_{ua}+q_{ua}}{2}\cdot\frac{p_{ua}p_{bc}+q_{ua}q_{bc}}{2} - \frac{p_{bc}+q_{bc}}{2}\cdot\frac{p_{ua}p_{ua}+q_{ua}q_{ua}}{2}}{\frac{p_{ub}+q_{ub}}{2}} \\
&= \frac{1}{4}(p_{ua}p_{bc} + p_{ua}q_{bc} + q_{ua}p_{bc} + q_{ua}q_{bc}) + \frac{2}{p_{ub}+q_{ub}}\left[\frac{p_{ua}p_{ub}p_{bc}+q_{ua}q_{ub}q_{bc}}{2}\right. \\
&\quad -\frac{1}{4}(p_{ua}p_{ub}p_{bc} + q_{ua}p_{ub}p_{bc} + p_{ua}q_{ub}q_{bc} + q_{ua}q_{ub}q_{bc}) \\
&\quad \left.-\frac{1}{4}(p_{ua}p_{ub}p_{bc} + p_{ua}p_{ub}q_{bc} + q_{ua}q_{ub}p_{bc} + q_{ua}q_{ub}q_{bc})\right] \\
&= \frac{1}{4}(p_{ua}p_{bc} + p_{ua}q_{bc} + q_{ua}p_{bc} + q_{ua}q_{bc}) - \frac{1}{2(p_{ub}+q_{ub})}\left[q_{ua}p_{ub}p_{bc} + p_{ua}q_{ub}q_{bc} + p_{ua}p_{ub}q_{bc} + q_{ua}q_{ub}q_{bc}\right] \\
&= \frac{1}{4}(p_{ua}p_{bc} + q_{ua}p_{bc} + q_{ua}q_{bc} + p_{ua}q_{bc}) - \frac{1}{2(p_{ub}+q_{ub})}\left[(p_{ub}+q_{ub})(q_{ua}p_{bc} + p_{ua}q_{bc})\right] \\
&= \frac{1}{4}(p_{ua}p_{bc} + q_{ua}q_{bc} - p_{ua}q_{bc} - q_{ua}p_{bc}) \\
&= \frac{1}{4}(p_{ua} - q_{ua})(p_{bc} - q_{bc})
\end{aligned}
$$

We can then use the same proof techniques as in Lemma 5.4.1, and finally obtain:

$$
|p_{ua} - q_{ua}| = \sqrt{\frac{(Y^{|}_{ua,ub} - X_{ua}X_{ua})\left(X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}\right)}{Y^{|}_{ub,bc} - X_{ua}X_{bc}}}.
$$

This gives us the required result. □

C.2.2 Resolving Sign Ambiguity across Base Estimators

The following lemma handles the sign ambiguity (s_{ua}) introduced above.

Lemma C.2.2. *Suppose Condition 1 and 2 are true, in the setting of infinite samples, for edges $(u,a),(u,b)$ with $a \neq b$ for any vertex u with degree ≥ 2, the sign pattern s_{ua}, s_{ub} satisfy the following relation.*

$$s_{ua}s_{ub} = \mathsf{sgn}(Y_{ua,ub} - X_{ua}X_{ub}).$$

Proof. From previous analysis, we have $\mathsf{sgn}(p_{ua} - q_{ua}) = s_{ua}$. Therefore:

$$\begin{aligned}\mathsf{sgn}(Y_{ua,ub} - X_{ua}X_{ub}) &= \mathsf{sgn}\left(\frac{(p_{ua} - q_{ua})(p_{ub} - q_{ub})}{4}\right) \\ &= s_{ua}s_{ub}.\end{aligned}$$

□

Thus fixing sign of one edge gives us the signs of all the other edges adjacent to a star vertex. A similar relationship can be established among the edges of a line vertex, using $\mathsf{sgn}\left(X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}\right)$.

C.2.3 Main algorithm - proofs

Here we will present in detail the sub-routines required by our algorithm and the essential lemmas needed for our main proof.

LearnEdges This procedure detects the edges in the underlying graph using the estimate X_{uv}.

Algorithm 4 LearnEdges

Input Vertex set V
Output Edges of the graph

Set $E \leftarrow \emptyset$
for $u < v \in V$ **do**
    Compute $\hat{X}_{uv}$
    if $\hat{X}_{uv} \geq \epsilon$ **then**
        $E \leftarrow E \cup \{(u, v)\}$
Return E

Claim 6. LearnEdges(V) *outputs* E *such that* $E = E_1 \cup E_2$.

Proof. For each pair of nodes $u, v \in V$, if $(u, v) \in E_1 \cup E_2$ then $X_{uv} \neq 0$ since $X_{uv} = 0$ if and only if $p_{uv} = q_{uv} = 0$, which is equivalent to the edge (u, v) not belonging in the mixture. □

LearnStar This procedure returns the weights of the outgoing edges of a star vertex using the star primitive discussed before.

Algorithm 5 LearnStar

Input Star vertex $u \in V$, edge set E, weights W
Output Weights of edges adjacent to u

Use star primitive with star vertex u and learn all adjacent edges weights W^*.
if $W = \emptyset$ **then**
    Fix sign of any edge and ensure sign consistency.
else
    Set $v \in V$ such that $(u, v) \in W$.
    Use s_{uv} to remove sign ambiguity
Return W^*.

Lemma C.2.3. *If $deg(u) \geq 3$,* LearnStar(u, S, W) *recovers p_{ua}, q_{ua} for all a such that $(u, a) \in E$.*

Proof. The proof follows from using Lemma 5.4.1 on star vertex u (degree of $u \geq 3$) and using Lemma C.2.2 to resolve sign ambiguity through fixing an edge or s_{uv} ($(u, v) \in W$ hence know sign). □

LearnLine This procedure returns the weights of the edges of a line $a - b - c - d$ rooted at vertex b of degree 2 using the line primitive discussed before.

Algorithm 6 LearnLine

Input Line $a - b - c - d$ with $deg(b) = 2$, edge set E, weights W
Output Weights of edges $(a, b), (b, c), (c, d)$

Use line primitive on $a - b - c - d$ rooted at b and learn all edges weights $W^{|}$.
if $W = \emptyset$ **then**
    Fix sign of any edge and ensure sign consistency.
else
    Find edge $e \in \{(a, b), (b, c), (c, d)\}$ such that $e \in W$.
    Use s_e to remove sign ambiguity.
Return $W^{|}$.

Lemma C.2.4. *If $deg(b) = 2$,* LearnLine(a, b, c, d, S, W) *recovers $p_{ab}, q_{ab}, p_{bc}, q_{bc}, pcd, q_{cd}$.*

Proof. The proof follows from using Lemma 5.4.2 on line $a - b - c - d$ rooted at vertex b (degree of $b = 2$) and using Lemma C.2.2 to resolve sign ambiguity by fixing an edge or using s_e. □

Learn2Nodes This procedure chooses a pair of connected vertices in our graph and outputs the weights of all outgoing edges of each of the two vertices. We initialize our algorithm using this procedure.

Algorithm 7 LEARN2NODES

Input Vertex set V, Edge Set E

Output Set of 2 vertices V, Weight of all edges adjacent to the vertices W

$W = \emptyset$
Set $u = \arg\max_{a \in V} deg(a)$
Set $v = \arg\min_{a \in V, (u,a) \in E} deg(a)$
if $deg(u) \geq 3$ **then**
    $W \leftarrow$ LEARNSTAR(u, E, W)
    if $deg(v) = 3$ **then**
        $W \leftarrow W \cup$ LEARNSTAR(v, E, W)
    else if $deg(v) = 2$ **then**
        Let $t \in V$ be such that $(t, v) \in E$ and $t \neq u$
        Let $w \in V$ be such that $(w, u) \in E$ and $w \neq v, t$
        if $v = t$ **then**
            $W \leftarrow W \cup$ LEARNLINE(t, v, u, w, W)
else
    w be such that $(w, u) \in E$ and $w \neq v$
    if $deg(v) = 2$ **then**
        Let $t \in V$ be such that $(t, v) \in E$ and $t \neq u$
        $W \leftarrow$ LEARNLINE(w, u, v, t, W)
    else
        Let $t \in V$ be such that $(t, w) \in E$ and $t \neq w$
        $W \leftarrow$ LEARNLINE(v, u, w, t, W)
Return $(u, v), W$

Lemma C.2.5. *Under Conditions 1 and 2,* LEARN2NODES(V) *outputs two connected nodes* (u, v) *and weights of all edges adjacent to* u, v.

Proof. We will break the proof down into cases based on the degree of chosen

vertices u, v as follows,

- $deg(u) \geq 3$: By Lemma C.2.3, we can recover all the edges of u and fix a sign.
  - $deg(v) \geq 3$: By Lemma C.2.3, we can recover all the edges of v and ensure sign consistency by using the edge (u, v).
  - $deg(v) = 2$: Since $deg(v) = 2$, there exists a vertex $t \neq u$ such that $(t, u) \in E$. Since $deg(u) \geq 3$, there must exist $w \neq t, u$ such that $(u, w) \in E$. Now we have line primitive $t - v - u - w$ with $deg(v) = 2$ and Lemma C.2.4 guarantees recovery of the edge weights.
  - $deg(v) = 1$, then we already know all the edges adjacent to v.
- $deg(u) = 2, deg(v) = 2$: Since the max degree of the graph is 2 and it is connected then it can either be a line or a cycle. There are at least 4 nodes in the graph, thus there exist $w \neq v$ such that $(w, u) \in E$ and $t \neq u, w$ such that $(v, t) \in E$. This gives a path $w - u - v - t$ with $deg(u) = 2$ and Lemma C.2.4 guarantees recovery of all edges.
- $deg(u) = 2, deg(v) = 1$: As in the previous case, the underlying graph is a line. Therefore there exist path $v - u - w - t$ and we can similarly apply Lemma C.2.4 to guarantee recovery of all edges.

□

C.2.4 Finite sample complexity - proofs

In this section, we provide explicit proof for the sample complexity of our algorithm. To do so, we bound below the number of cascades starting on each node through Bernstein inequality, and use this number to obtain concentration of all the estimators.

Definition C.2.1. Among M cascades, let M_u be the number of times node u is the source.

Claim 7. *With M samples, every node is the source of the infection at least $\frac{M}{2N}$ times with probability at least $1 - e^{-\frac{3M}{26N}}$.*

Proof. Among M cascade, the expectation of M_u is $\frac{M}{N}$, since the source is chosen uniformly at random among the N vertices of V. Since M_u can be seen as the sum of Bernoulli variable of parameter $\frac{1}{N}$, we can use Bernstein's inequality to bound it below:

$$\begin{aligned}\Pr(M_u < \frac{M}{2N}) &= \Pr\left(\frac{M}{N} - M_u > \frac{M}{2N}\right) \\ &\leq e^{-\frac{\left(\frac{M}{2N}\right)^2}{2M\frac{1}{N}(1-\frac{1}{N})+\frac{1}{3}\frac{M}{2N}}} \\ &\leq e^{-\frac{3M}{26N}}.\end{aligned}$$

Claim 8. *Let u either be a star vertex, with neighbors a, b and c, or be part of a line structure rooted in u, with neighbors a, b, and c neighbor of b. Suppose $M_u \geq \frac{M}{2N}$. Then with $M = \frac{N}{\epsilon^2}\log\left(\frac{12N^2}{\delta}\right)$ samples, with probability at least $1 - \frac{\delta}{6N^2}$, we can guarantee any of the following:*

1. $\forall r \in a, b, c, \ \left|\hat{X}_{ur} - X_{ur}\right| \leq \epsilon_1$.

2. $\forall r \neq s \in \{a, b, c\}, \ |\hat{Y}^*_{ur,us} - \hat{Y}^*_{ur,us}| \leq \epsilon_1$.

3. $|\hat{Y}^{|}_{ua,ub} - \hat{Y}^{|}_{ua,ub}| \leq \epsilon_1$ *and* $|\hat{Y}^{|}_{ua,ab} - \hat{Y}^{|}_{ua,ab}| \leq \epsilon_1$.

4. $|\hat{Z}^{|}_{ua,ub,bc} - Z^{|}_{ua,ub,bc}| \leq \epsilon_1$.

Proof. By Hoeffding's inequality:

$$\begin{aligned}
\Pr(|\hat{X}_{ur} - X_{ur}| > \epsilon_1) &= \Pr\left(\left|\sum_{m=1}^{M_u} \mathbb{1}_{\{u \to r \ | \ u \in I_0\}} - M_u \cdot X_{ur}\right| > M_u \cdot \epsilon_1\right) \\
&\leq \Pr\left(\left|\sum_{m=1}^{\frac{M}{2N}} \mathbb{1}_{\{u \to r \ | \ u \in I_0\}} - \frac{M}{2N} \cdot X_{ur}\right| > \frac{M}{2N} \cdot \epsilon_1\right) \\
&\leq 2e^{-2\frac{M}{2N}\epsilon_1^2}.
\end{aligned}$$

Therefore, the quantity above is smaller than $\frac{\delta}{6N^2}$ for $M \geq \frac{N}{\epsilon_1^2} \log\left(\frac{12N^2}{\delta}\right)$. The proof is almost identical for the other quantities involved.

Claim 9. *If we can estimate $X_{ua}, Y^*_{ua,ub}, Y^{|}_{ua,ab}$ and $Z^{|}_{ua,ub,bc}$ within ϵ_1, we can estimate p_{ua} within precision $\epsilon = \frac{41}{p^3_{min}\Delta^2} \cdot \epsilon_1$.*

Proof. If u is of degree three or more, we use a star primitive to estimate it.

Let a, b and c be three of its neigbors:

$$\begin{aligned}
\hat{p}_{ua} &= \hat{X}_{ua} + s_{ua}\sqrt{\frac{(\hat{Y}_{ua,ub} - \hat{X}_{ua}\hat{X}_{ub})(\hat{Y}_{ua,uc} - \hat{X}_{ua}\hat{X}_{uc})}{\hat{Y}_{ub,uc} - \hat{X}_{ub}\hat{X}_{uc}}} \\
&\leq X_{ua} + \epsilon_1 \\
&\quad + s_{ua}\left(\frac{(Y_{ua,ub} - X_{ua}X_{ub} + s_{ua}\left(1 + X_{ua} + X_{ub}\right)\epsilon_1)(Y_{ua,uc} - X_{ua}X_{uc} + s_{ua}\left[1 + X_{ua} + X_{uc}\right)\epsilon_1)}{Y_{ub,bc} - X_{ub}X_{uc} - s_{ua}\left(1 + X_{ub} + X_{uc}\right)\epsilon_1)}\right] \\
&\leq X_{ua} + \epsilon_1 + s_{ua}\sqrt{\frac{(Y_{ua,ub} - X_{ua}X_{ub})(Y_{ua,uc} - X_{ua}X_{uc})}{Y_{ub,uc} - X_{ub}X_{uc}}}\left(\frac{(1 + s_{ua}\frac{3\epsilon_1}{\frac{\Delta^2}{4}})^2}{1 - s_{ua}\frac{3\epsilon_1}{\frac{\Delta^2}{4}}}\right)^{\frac{1}{2}} \\
&\leq p_{ua} + \epsilon_1 + p_{ua}\left(\frac{12}{\Delta^2} + \frac{6}{\Delta^2}\right)\cdot\epsilon_1 + o(\epsilon_1) \\
&\leq p_{ua} + \frac{19}{\Delta^2}\cdot\epsilon_1 + o(\epsilon_1).
\end{aligned}$$

Where we have used $Y_{ur,us} - X_{ur}X_{us} \geq \frac{\Delta^2}{4}$, $s_{ua}^2 = 1$, $p_{ua} \leq 1$, $1 \leq \frac{1}{\Delta^2}$. We then conclude by symmetry.

If u is of degree two, we use a line primitive to estimate it:

$$\begin{aligned}
\hat{p}_{ua} &= \hat{X}_{ua} + s_{ua}\sqrt{\frac{(\hat{Y}^{|}_{ua,ub} - \hat{X}_{ua}\hat{X}_{ub})\left(\hat{X}_{ua}\hat{X}_{bc} + \frac{\hat{Z}^{|}_{ua,ub,bc} - \hat{X}_{ua}\hat{Y}^{|}_{ub,bc} - \hat{X}_{bc}\hat{Y}^{|}_{ua,ub}}{\hat{X}_{ub}}\right)}{\hat{Y}^{|}_{ub,bc} - \hat{X}_{ua}\hat{X}_{bc}}} \\
&\leq X_{ua} + \epsilon_1 + s_{ua}\sqrt{\frac{(Y^{|}_{ua,ub} - X_{ua}X_{ub} + 3s_{ua}\epsilon_1)\left(X_{ua}X_{bc} + 2\epsilon_1 + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub} + 5s_{ua}\epsilon_1}{X_{ub} - s_{ua}\epsilon_1}\right)}{Y^{|}_{ub,bc} - X_{ua}X_{bc} - 3s_{ua}\epsilon_1}}
\end{aligned}$$

As shown in the proof of Lemma 5.4.2, we have:

$$\begin{aligned} Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub} &= \frac{1}{2}(p_{ub}+q_{ub})(q_{ua}p_{bc}+p_{ua}q_{bc}) \\ &\geq \frac{p^3_{min}}{2} \\ \left(X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}\right) &= \frac{1}{4}(p_{ua}-q_{ua})(p_{bc}-q_{bc}) \\ &\geq \frac{\Delta^2}{4}. \end{aligned}$$

Therefore:

$$\begin{aligned} \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub} + 5s_{ua}\epsilon_1}{X_{ub} - s_{ua}\epsilon_1} &\leq \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}} \left[\frac{1 + s_{ua}\frac{5\epsilon_1}{\frac{p^3_{min}}{2}}}{1 - s_{ua}\frac{\epsilon_1}{\frac{p_{min}}{2}}}\right] \\ &\leq \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}} + s_{ua}\left(\frac{12}{p^3_{min}}\right)\epsilon_1 + o(\epsilon_1) \end{aligned}$$

We also have:

$$\begin{aligned} X_{ua}X_{bc} + 2\epsilon_1 + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub} + 5s_{ua}\epsilon_1}{X_{ub} - s_{ua}\epsilon_1} &\leq \left(X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y_{\cdot}}{X_{ub}}\right. \\ &\quad \left. \cdot \left(1 + s_{ua}\frac{\frac{14}{p^3_{min}}}{\frac{\Delta^2}{4}}\epsilon_1\right) + o(\epsilon_1).\right. \end{aligned}$$

Combining all the above inequalitites:

$$\hat{p}_{ua} \leq X_{ua} + \epsilon_1 + s_{ua}\sqrt{\frac{(Y^{|}_{ua,ub} - X_{ua}X_{ub})\left(X_{ua}X_{bc} + \frac{Z^{|}_{ua,ub,bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}}{X_{ub}}\right)}{Y^{|}_{ub,bc} - X_{ua}X_{bc}}}$$

$$\cdot \left[\frac{\left(1 + \frac{3\epsilon_1}{\frac{\Delta^2}{4}}\right)\left(1 + s_{ua}\frac{\frac{14}{p^3_{min}}}{\frac{\Delta^2}{4}}\epsilon_1\right)}{1 - s_{ua}\frac{3\epsilon_1}{\frac{\Delta^2}{4}}}\right]^{\frac{1}{2}}$$

$$\leq p_{ua} + \epsilon_1 + p_{ua}s^2_{ua}\left(\frac{6}{\Delta^2} + \frac{28}{p^3_{min}\Delta^2} + \frac{6}{\Delta^2}\right)\cdot\epsilon_1 + o(\epsilon_1)$$

$$\leq p_{ua} + \frac{41}{p^3_{min}\Delta^2}\cdot\epsilon_1 + o(\epsilon_1).$$

We can conclude by symmetry.

Since $\frac{41}{p^3_{min}\Delta^2}\cdot\epsilon_1 \geq \frac{19}{\Delta^2}\cdot\epsilon_1$, we conclude that we can know p_{ua} within precision $\epsilon = \frac{41}{p^3_{min}\Delta^2}\cdot\epsilon_1$ regardless of the degree of u.

Theorem C.2.6. *Under Conditions 1 and 2,, with probability* $1-\delta$*, with* $M = N\cdot\frac{41^2}{p^6_{min}\Delta^4\cdot\epsilon^2}\log\left(\frac{12N^2}{\delta}\right) = \mathcal{O}\left(\frac{N}{\epsilon^2}\log\left(\frac{N}{\delta}\right)\right)$ *samples, we can learn all the edges of the mixture of the graphs within precision* ϵ*.*

Proof. We pick $\epsilon = \frac{41}{p^3_{min}\Delta^2}\cdot\epsilon_1$. We use Claim 7 to bound the quantity $\Pr(M_u < \frac{M}{2N})$, and Claim 8 and 9 to bound $\Pr(|\hat{p}_{ua} - p_{ua}| > \frac{41}{p^3_{min}\Delta^2}\cdot\epsilon_1 | M_u \geq \frac{M}{2N})$. For

(u, a) edge of the graph:

$$\begin{aligned}
\Pr(|\hat{p}_{ua} - p_{ua}| > \epsilon) &\leq \Pr(|\hat{p}_{ua} - p_{ua}| > \epsilon | M_u < \frac{M}{2n}) \cdot \Pr(M_u < \frac{M}{2N}) \\
&\quad + \Pr(|\hat{p}_{ua} - p_{ua}| > \epsilon | M_u \geq \frac{M}{2n}) \cdot \Pr(M_u \geq \frac{M}{2N}) \\
&\leq 1 \cdot 2e^{-2\frac{M}{2N}} + \Pr(|\hat{p}_{ua} - p_{ua}| > \epsilon | M_u \geq \frac{M}{2N}) \cdot 1 \\
&\leq \frac{\delta}{12N^2} + \Pr(|\hat{p}_{ua} - p_{ua}| > \frac{41}{p_{min}^3 \Delta^2} \cdot \epsilon_1 | M_u \geq \frac{M}{2N}) \\
&\leq \frac{\delta}{12N^2} + \frac{\delta}{12N^2} \\
&\leq \frac{\delta}{6N^2}.
\end{aligned}$$

We conclude by union bound on the six estimators involved for all the pairs of nodes in the graph, for a total of at most $6N^2$ estimators.

C.2.5 Complete graph on three nodes

In this section, we prove it is possible to recover the weights of a mixture on three nodes, as long as there are at least three edges in $E_1 \cup E_2$. Since no node is of degree 3, no node is a star vertex, and since there are less than four nodes, no node is a line vertex, and we can not use the techniques developped above for connected graphs on four vertices or more. However, we can still use very similar proofs techniques. Suppose the vertices of V are 1, 2 and 3.

Definition C.2.2. We reuse the quantities defined for star vertices:

- For i, j distinct in $\{1, 2, 3\}$, $\hat{X}_{ij} = \frac{\frac{1}{M}\sum_{m=1}^{M} \mathbb{1}_{i \to j, i \in I_0^m}}{\frac{1}{M}\sum_{m=1}^{M} \mathbb{1}_{i \in I_0^m}} \to_{M \to \infty} X_{ij} = \frac{p_{ij}+q_{ij}}{2}$.

- For i, j, k distinct in $\{1,2,3\}$, $Y_{ij,ik} = \frac{\frac{1}{M}\sum_{m=1}^{M} \mathbb{1}_{i\to j, i\to k, i\in I_0^m}}{\frac{1}{M}\sum_{m=1}^{M} \mathbb{1}_{u\in I_0^m}} \to_{M\to\infty} Y_{ij,ik} = \frac{p_{ij}p_{ik}+q_{ij}q_{ik}}{2}$.

Even though neither 1, 2 or 3 is a star vertex, we can write the same kind of system of equations as a star vertex would satisfy. In particular:

$$\frac{|p_{ij}-q_{ij}|}{2} = \sqrt{\frac{(Y_{ij,ik}-X_{ij}ik)(Y_{ji,jk}-X_{ji}X_{jk})}{Y_{ki,kj}-X_{ki}X_{kj}}}.$$

Resolving the sign ambiguity as previoulsy (Lemma C.2.2), this finally yields:

$$p_{ij} = X_{ij} + s_{ij}\sqrt{\frac{(Y_{ij,ik}-X_{ij}ik)(Y_{ji,jk}-X_{ji}X_{jk})}{Y_{ki,kj}-X_{ki}X_{kj}}},$$
$$q_{ij} = X_{ij} + s_{ij}\sqrt{\frac{(Y_{ij,ik}-X_{ij}ik)(Y_{ji,jk}-X_{ji}X_{jk})}{Y_{ki,kj}-X_{ki}X_{kj}}}.$$

C.3 Lower Bounds

C.3.1 Directed lower bound

We consider the task of learning all the edges of any mixture of graphs up to precision $\epsilon < \Delta$. To do so, we have to be able to learn a mixture on a specific graph, which we present below.

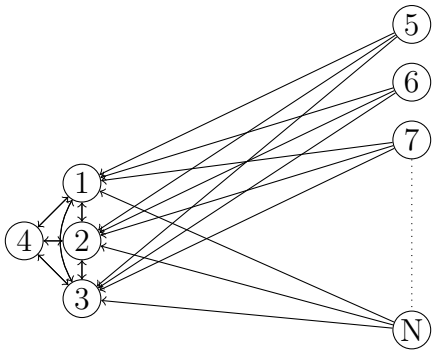


Figure C.1: Lower-bound directed graph

The example we focus on is the directed graph of min-degree 3, comprised of a clique on 4 nodes, which we call nodes 1 to 4, and $N - 4$ other nodes with 3 directed edges to nodes 1, 2 and 3. All edges have weight p in E_1, and $p + \Delta$ in E_2.

We define a *valid sample* for edge (i, j) as a cascade during which i became infected when j was not infected. Indeed, in this case, an infection could happen along edge (i, j), and we can therefore gain information about the weight of this edge. We first state a general claim:

Claim 10. *We need at least $\Omega(\frac{1}{\Delta^2})$ valid samples for edge (i, j) to determine the weights of this edge in the mixture.*

Proof. Using Sanov's theorem [70], and writing the Kullback–Leibler divergence between p and q as $\mathcal{D}(p||q)$, we know we need at least $\Omega(\mathcal{D}(p||p + \Delta))$ valid samples to determine whether the valid samples came from a random flip of probability p, or a random flip of probability $p + \Delta$, which is an easier task than computing both weights of the mixture.

Then, using standard Kullback–Leibler divergence bounds [21], we obtain $\mathcal{D}(p||p+\Delta) \geq \frac{1}{\Delta^2}$, which gives us the desired result.

We now combine this with Coupon collector's result to obtain our lower bound.

Claim 11. *We need at least* $\Omega\left(N\log(N) + \frac{N\log\log(N)}{\Delta^2}\right)$ *cascades to obtain enough valid samples for all the edges in the graph.*

Proof. We notice that if we want to learn all edges in the graph, it implies that we have to learn all the edges from the $N-4$ nodes to node 1. However, if i is not part of the clique, any valid sample for such an edge (i, 1) has to have i as its source. Having enough valid samples for each of these edges is therefore equivalent to collecting $\Omega(\frac{1}{\Delta^2})$ copies of $N-4$ distinct coupons in the standard Coupon collector problem. Using results from [64, **?**], we need $\Omega((K\log(K) + (d-1)\cdot K\cdot\log\log(K))$ samples to obtain d copies of each coupon when there are K distinct coupons in total, which is here $\Omega\left((N-4)\log(N-4) + (\frac{1}{\Delta^2}-1)\cdot(N-4)\cdot\log\log(N-4)\right)$ cascades. Using standard approximation, we get the desired result.

Combining the results:

Theorem C.3.1. *We need at least* $\Omega\left(N\log(N) + \frac{N\log\log(N)}{\Delta^2}\right)$ *cascades to learn any mixture of directed graphs of minimum out-degree 3.*

C.3.2 Undirected lower bound

We reuse a lot of the techniques in the previous subsection. This time, we consider a simple line graph on N nodes, where for all $1 \leq i \leq N-1$, node i is connected to node $i+1$. Like in the previous example, the weights are all p in G_1, and all $p+\Delta$ in G_2.

Reusing Claim 10, we now prove:

Claim 12. *We need at least $\Omega\left(\frac{N}{\Delta^2}\right)$ cascades to obtain enough valid samples for edge (1,2).*

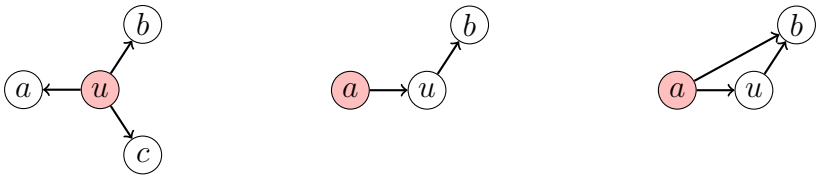
Proof. To provide a valid sample, either:

- Node 1 is the source, which happens with probability $\mathcal{P}_1 = \frac{1}{N}$.
- Node 2 was infected, which happens with probability $\mathcal{P}_2 \leq \sum_{i=2}^{N} \frac{1}{N} p_{max}^{i-2} \leq \frac{1}{N} \frac{1}{1-p_{max}}$.

Therefore, the probability of getting a valid sample is smaller than $\mathcal{P}_1 + \mathcal{P}_2 \leq \frac{1}{N} \cdot \frac{2}{1-p_{max}}$. Hence, we need at least $\Omega(\frac{1-p_{max}}{2} \cdot N \cdot \frac{1}{\Delta^2}) = \Omega\left(\frac{N}{\Delta^2}\right)$ cascades to obtain enough valid samples.

Since we need to learn at least edge $(1,2)$ to learn all the edges of this graph:

Theorem C.3.2. *We need at least $\Omega\left(\frac{N}{\Delta^2}\right)$ cascades to learn any mixture of undirected graphs.*



(a) A star vertex u for a directed graph.

(b) First structure to ensure sign consistency.

(c) Second structure to ensure sign consistency.

Figure C.2: Structures for directed graphs of minimum out-degree three.

C.4 Directed graphs

C.4.1 Structures

Star vertex For directed graph of out-degree at least 3, every vertex is a star vertex. This implies we can reuse the star vertex equations to learn the weights of the whole neighborhood of each node. However, if we learn the neighborhoods of node u in both graphs, which we call $\mathcal{N}_1^u$ and $\mathcal{N}_2^u$, as well as the neighbordhoods of node a, which we call $\mathcal{N}_1^a$ and $\mathcal{N}_2^a$, it is impossible to recover from the star structure alone if $\mathcal{N}_1^u$ and $\mathcal{N}_1^a$ are in the same mixture, or if it is $\mathcal{N}_1^u$ and $\mathcal{N}_2^a$ instead. We therefore use the two other structures in Figure C.2 to ensure mixture consistency.

Mixture consistency Suppose we have learned the weights of all the edges stemming from a, as well as all the weighted edges stemming from u, and suppose there is no edge between a and b. The probability that a infected u, which in turn infected b is:

$$\mathbb{P}(a \to u \to b | a \in I_0) = \frac{p_{au}p_{ub} + q_{au}q_{ub}}{2}.$$

This gives us a way to decide whether $\mathcal{N}_1^u$ and $\mathcal{N}_1^a$ are in the same mixture, or if it is $\mathcal{N}_1^u$ and $\mathcal{N}_2^a$ instead. Indeed, if we know $p_{au} \in \mathcal{N}_1^a, q_{au} \in \mathcal{N}_2^a$, and we also know $w_{ub} \in \mathcal{N}_1^u, w'_{ub} \in \mathcal{N}_2^u$, and we have an estimator $\hat{Y}_{au,ub}$ for $\mathbb{P}(a \to u \to b | a \in I_0)$, then we can check whether $\hat{Y}_{au,ub} \approx \frac{p_{au}w_{ub}+q_{au}w'_{ub}}{2}$, in which case $\mathcal{N}_1^u$ belongs with $\mathcal{N}_1^a$, or whether $\hat{Y}_{au,ub} \approx \frac{p_{au}w'_{ub}+q_{au}w_{ub}}{2}$, in which case $\mathcal{N}_2^u$ belongs in the with $\mathcal{N}_1^a$. We call this procedure CheckPath.

Similarly, if there is an edge between a and b, then:

$$\mathbb{P}(a \to u \to b | a \in I_0) = \frac{p_{au}(1-p_{ab})p_{ub} + q_{au}(1-q_{ab})q_{ub}}{2}.$$

This also allows us to ensure mixture consistency. We call this procedure CheckTriangle.

Here is the final algorithm:

Algorithm 8 Learn the weights of directed edges

Input Vertex set V

Output Edge weights for the two epidemics graphs

$E \leftarrow$ LearnEdges(V)
Select any first node v
$W \leftarrow$ LearnStar(v, E, W)
$S = \{v\}$
while $S \neq V$ **do**
    Select $a \in S, v \in V \backslash S$ such that $(a, u) \in E$ ▷ v **has out-degree at least 3**
    $\mathcal{N}_1, \mathcal{N}_2 \leftarrow$ LearnStar(u, E, W)
    Select $b \neq a$ neighbor of u ▷ b **exists because u os of degree at least 3.**
    if $(a, b) \notin E$ **then** ▷ **Use first structure.**
        if CheckPath($v, u, b, W, \mathcal{N}_1, \mathcal{N}_2$) **then**
            $W = \{W_1 \cup \mathcal{N}_1, W_2 \cup \mathcal{N}_2\}$
        else
            $W = \{W_1 \cup \mathcal{N}_2, W_2 \cup \mathcal{N}_1\}$
    else ▷ **Use second structure.**
        if CheckTriangle($v, u, b, W, \mathcal{N}_1, \mathcal{N}_2$) **then**
            $W = \{W_1 \cup \mathcal{N}_1, W_2 \cup \mathcal{N}_2\}$
        else
            $W = \{W_1 \cup \mathcal{N}_2, W_2 \cup \mathcal{N}_1\}$
    $S \leftarrow S \cup \{u\}$
return W

C.5 Unbalanced/Unknown Mixtures

In this section we provide the primitives required for LearnStar and LearnLine, when the first mixture occurs with probability α and the second mixture with probability $(1 - \alpha)$.

Notations: In this section, to avoid clutter in notation we use i, j

and k to be all *distinct* unless mentioned otherwise. Also, let $\sigma(\{a,b,c\}) = \{(a,b,c),(b,c,a),(c,a,b)\}$ denote all the permutations of a, b, and c.

Claim 13. *If a and b are two distinct nodes of $V_1 \cap V_2$ such that $(a,b) \in E_1 \cap E_2$ then under general mixture model $X_{ab} = \alpha p_{ab} + (1-\alpha) q_{ab}$.*

Fuither, when the four nodes u, a, b and c forms a star graph (Fig. 5.3) with u in the center under general mixture model

$$1)\ \forall i,j \in \{a,b,c\}, i,j \neq u, Y_{ui,uj} = \alpha p_{ui} p_{uj} + (1-\alpha) q_{ui} q_{uj},$$

$$2)\ Z_{ua,ub,uc} = \alpha p_{ua} p_{ub} p_{uc} + (1-\alpha) q_{ua} q_{ub} q_{uc}.$$

Finally, when the four nodes u, a, b and c forms a line graph (Fig. 5.4) under general mixture model

$$1)\ Y^{|}_{ua,ub} = \alpha p_{ua} p_{ub} + (1-\alpha) q_{ua} q_{ub}, 2)\ Y^{|}_{ub,bc} = \alpha p_{ub} p_{bc} + (1-\alpha) q_{ub} q_{bc},$$

$$3)\ Z^{|}_{ua,ub,bc} = \alpha p_{ua} p_{ub} p_{bc} + (1-\alpha) q_{ua} q_{ub} q_{bc}.$$

The proof of the above claim is omitted as it follows closely the proofs of Claim 3, 4, and 5.

C.5.1 Star Graph

We now present the following two lemmas which recover the weights p_{ui},and q_{ui} for all $i \in \{a,b,c\}$ in the star graph (Fig. 5.3), and the general mixture parameter α, respectively.

Lemma C.5.1 (Weights of General Star Graph)**.** *Under Conditions 1 and 2, in the setting of infinite samples, for the star structure (u, a, b, c) with u as the central vertex the weight of any edge (u, a) is given by:*

$$p_{ua} = X_{ua} + s_{ua}\sqrt{\tfrac{1-\alpha}{\alpha}}\sqrt{\frac{(Y_{ua,ub} - X_{ua}X_{ub})(Y_{ua,uc} - X_{ua}X_{uc})}{Y_{ub,uc} - X_{ub}X_{uc}}}$$

$$q_{ua} = X_{ua} - s_{ua}\sqrt{\tfrac{\alpha}{1-\alpha}}\sqrt{\frac{(Y_{ua,ub} - X_{ua}X_{ub})(Y_{ua,uc} - X_{ua}X_{uc})}{Y_{ub,uc} - X_{ub}X_{uc}}}$$

where $s_{ua} \in \{-1, 1\}$ and $b, c \in N_1(u) \cap N_2(u)$ such that $b, c \neq a$, $b \neq c$.

Proof. We notice that for $r \neq j \in \{a, b, c\}$

$$\begin{aligned}(Y_{ui,uj} - X_{ui}X_{uj}) &= (\alpha p_{ui}p_{uj} + (1-\alpha)q_{ui}q_{uj}) - (\alpha p_{ui} + (1-\alpha)q_{ui})\,(\alpha p_{uj} + (1-\alpha)q_{uj}) \\ &= \alpha(1-\alpha)(p_{ui} - q_{ui})(p_{uj} - q_{uj}).\end{aligned}$$

The rest of the proof follows the same steps as given in the proof of Lemma 5.4.1 with the above modification. □

Lemma C.5.2 (Sign Ambiguity Star Graph)**.** *Under Conditions 1 and 2, in the setting of infinite samples, for edges $(u, a), (u, b)$ for the star structure (u, a, b, c) with u as the central vertex, the sign pattern s_{ua}, s_{ub} satisfy the following relation.*

$$s_{ub}s_{ua} = \mathsf{sgn}(Y_{ua,ub} - X_{ua}X_{ub}).$$

Proof. The proof of the first statement follows the same logic as the proof of Lemma C.2.2, after noting that $\mathsf{sgn}(\alpha(1-\alpha)) = 1$ for $\alpha \in (0, 1)$. □

C.5.2 Line Graph

We now present the recovery of parameters in the case of a line graph with knowledge of α

Lemma C.5.3 (Weights of General Line Graph)**.** *Under Conditions 1 and 2, in the setting of infinite samples, the weights of the edges* (u, a)*, and* (u, b) *for a line graph* $a - u - b - c$ *can be learned in closed form (as given in the proof), as a function of*

(1) the mixture parameter α*,*

(2) estimators X_{ua}*,* X_{ub}*,* X_{bc}*,* $Y^{|}_{ua,ub}$*,* $Y^{|}_{ub,bc}$*, and* $Z^{|}_{ua,ub,bc}$*,*

(3) one variable $s_{ub} \in \{-1, +1\}$*.*

Proof. We first note that we have access to the following three relations

$$
\begin{aligned}
&1) \quad (Y^{|}_{ua,ub} - X_{ua}X_{ub}) = \alpha(1-\alpha)(p_{ua} - q_{ua})(p_{ub} - q_{ub}) \\
&2) \quad (Y^{|}_{ub,bc} - X_{ub}X_{bc}) = \alpha(1-\alpha)(p_{ub} - q_{ub})(p_{bc} - q_{bc}) \\
&3) \quad (Z^{|}_{ua,ub,bc} + X_{ua}X_{ub}X_{bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}) \\
&\qquad = \alpha(1-\alpha)((1-\alpha)p_{ub} + \alpha q_{ub})(p_{ua} - q_{ua})(p_{bc} - q_{bc}).
\end{aligned}
$$

The first two inequalities follow similar to Lemma 5.4.2. We derive the final

equality below.

$$
\begin{aligned}
&Z^{|}_{ua,ub,bc} + X_{ua}X_{ub}X_{bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub} \\
&= \alpha p_{ua}p_{ub}p_{bc} + (1-\alpha)q_{ua}q_{ub}q_{bc} \\
&- (\alpha p_{ua} + (1-\alpha)q_{ua})((\alpha p_{ub}p_{bc} + (1-\alpha)q_{ub}q_{bc}) - (\alpha p_{bc} + (1-\alpha)q_{bc})((\alpha p_{ua}p_{ub} + (1-\alpha)q_{ua}q_{ub}) \\
&+ (\alpha p_{ua} + (1-\alpha)q_{ua})(\alpha p_{ub} + (1-\alpha)q_{ub})(\alpha p_{bc} + (1-\alpha)q_{bc}) \\
&= \alpha(1-\alpha)^2 p_{ua}p_{ub}p_{bc} + \alpha^2(1-\alpha)q_{ua}q_{ub}q_{bc} \\
&- \alpha(1-\alpha)^2 p_{ua}p_{ub}q_{bc} + \alpha^2(1-\alpha)p_{ua}q_{ub}p_{bc} - \alpha(1-\alpha)^2 q_{ua}p_{ub}p_{bc} \\
&- \alpha^2(1-\alpha)q_{ua}q_{ub}p_{bc} + \alpha(1-\alpha)^2 q_{ua}p_{ub}q_{bc} - \alpha^2(1-\alpha)p_{ua}q_{ub}q_{bc} \\
&= \alpha(1-\alpha)((1-\alpha)p_{ub} + \alpha q_{ub})(p_{ua} - q_{ua})(p_{bc} - q_{bc})
\end{aligned}
$$

Therefore, we obtain the following quadratic equation in p_{ub} and q_{ub} (unlike the $\alpha = 1/2$ case it cannot be easily reduced to a linear equation),

$$\frac{\alpha(1-\alpha)(p_{ub} - q_{ub})^2}{((1-\alpha)p_{ub} + \alpha q_{ub})} = \frac{(Y^{|}_{ua,ub} - X_{ua}X_{ub})(Y^{|}_{ub,bc} - X_{ub}X_{bc})}{(Z^{|}_{ua,ub,bc} + X_{ua}X_{ub}X_{bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub})} := C^{|}_{ub}$$

Note that $X_{ub} = \alpha p_{ub} + (1-\alpha)q_{ub}$, thus the above can be reduced to

$$
\begin{aligned}
&\frac{\alpha(1-\alpha)(p_{ub} - X_{ub})^2/(1-\alpha)^2}{(p_{ub}(1-2\alpha) + \alpha X_{ub})/(1-\alpha)} = C^{|}_{ub} \\
&p^2_{ub} - 2\left(X_{ub} + \tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ub}\right)p_{ub} = C^{|}_{ub}X_{ub} - X^2_{ub} \\
&p_{ub} = X_{ub} + \tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ub} + s_{ub}\sqrt{\left(\tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ub}\right)^2 + \tfrac{1-\alpha}{\alpha}C^{|}_{ub}X_{ub}} \\
&q_{ub} = X_{ub} - \tfrac{(1-2\alpha)}{2(1-\alpha)}C^{|}_{ub} - s_{ub}\sqrt{\left(\tfrac{(1-2\alpha)}{2(1-\alpha)}C^{|}_{ub}\right)^2 + \tfrac{\alpha}{1-\alpha}C^{|}_{ub}X_{ub}}
\end{aligned}
$$

We substitute in the above two equations θ and s_α as defined below

$$\alpha = \tfrac{1}{2}(1 - s_\alpha\sqrt{\theta}), \qquad (1-\alpha) = \tfrac{1}{2}(1 + s_\alpha\sqrt{\theta}), \qquad (1-2\alpha) = s_\alpha\sqrt{\theta}.$$

From the substitution we obtain,

$$p_{ub} = X_{ub} + \frac{s_\alpha\sqrt{\theta}(1+s_\alpha\sqrt{\theta})C^{|}_{ub}}{(1-\theta)}\left(1 + s_\alpha s_{ub}\sqrt{1+\frac{(1-\theta)X_{ub}}{\theta C^{|}_{ub}}}\right)$$
$$q_{ub} = X_{ub} - \frac{s_\alpha\sqrt{\theta}(1-s_\alpha\sqrt{\theta})C^{|}_{ub}}{(1-\theta)}\left(1 + s_\alpha s_{ub}\sqrt{1+\frac{(1-\theta)X_{ub}}{\theta C^{|}_{ub}}}\right)$$

Next we use p_{ub}, and q_{ub} to obtain p_{ua}, and q_{ua}. Specifically, we have

$$\alpha(1-\alpha)(p_{ub}-q_{ub})(p_{ua}-q_{ua}) = (Y^{|}_{ua,ub} - X_{ua}X_{ub})$$
$$(p_{ua}-q_{ua}) = \frac{4(Y^{|}_{ua,ub} - X_{ua}X_{ub})}{s_\alpha\sqrt{\theta}\left(1 + s_\alpha s_{ub}\sqrt{1+\frac{(1-\theta)X_{ub}}{\theta C^{|}_{ub}}}\right)}.$$

Finally, we use the above relation to arrive at the required result.

$$p_{ua} = X_{ua} + \frac{2(1+s_\alpha\sqrt{\theta})(Y^{|}_{ua,ub} - X_{ua}X_{ub})}{s_\alpha\sqrt{\theta}\left(1 + s_\alpha s_{ub}\sqrt{1+\frac{(1-\theta)X_{ub}}{\theta C^{|}_{ub}}}\right)}$$
$$q_{ua} = X_{ua} - \frac{2(1-s_\alpha\sqrt{\theta})(Y^{|}_{ua,ub} - X_{ua}X_{ub})}{s_\alpha\sqrt{\theta}\left(1 + s_\alpha s_{ub}\sqrt{1+\frac{(1-\theta)X_{ub}}{\theta C^{|}_{ub}}}\right)}$$

□

Lemma C.5.4 (Sign Ambiguity Line graph on 5 nodes)**.** *Under Conditions 1 and 2, in the setting of infinite samples, for a line structure $a-u-b-c-d$ the sign patterns s_{ub} and s_{bc} satisfy the relation, $s_{ub}s_{bc} = \mathsf{sgn}(Y^{|}_{ub,bc} - X_{ub}X_{bc})$.*

Proof. The proof is almost identical to the other sign ambiguity proofs. □

C.5.3 Finite Sample Complexity

We start by observing that the Claim 7 still holds in the general case.

Claim 14. *If we can estimate $X_{ua}, Y^{*}_{ua,ub}, Y^{|}_{ua,ab}$ and $Z^{|}_{ua,ub,bc}$ within ϵ_1, we can estimate p_{ua} and q_{ua} within precision $\epsilon = \mathcal{O}\left(\epsilon_1 / \min(p_{min}, \Delta)^5 \min(\alpha, 1-\alpha)^4\right)$.*

Proof. The proof proceeds in a very similar manner as Claim9. Following the derivations for $\hat{p}_{ua}$ and $\hat{q}_{ua}$ in the proof of Claim9, we can see that for the star primitive all the computation carry over with a scaling of $\frac{4}{\alpha(1-\alpha)}$ as we have $Y^{*}_{ur,us} - X_{ur}X_{us} \geq \Delta^2\alpha(1-\alpha)$ instead of $\Delta^2/4$.

The line primitive presents with increased difficulty as the estimator is more complex. We first observe that $\alpha(1-\alpha)\Delta^2 \leq C^{|}_{ub} \leq \max(\alpha, (1-\alpha))$. We recall that

$$
\begin{aligned}
&(Z^{|}_{ua,ub,bc} + X_{ua}X_{ub}X_{bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub}) \\
&= \alpha(1-\alpha)((1-\alpha)p_{ub} + \alpha q_{ub})(p_{ua} - q_{ua})(p_{bc} - q_{bc}) \\
&\geq \min(\alpha, 1-\alpha)^2 p_{min} \min(p_{min}, \Delta)^2/2, \\
&(Y^{|}_{ua,ub} - X_{ua}X_{ub}) = \alpha(1-\alpha)(p_{ua} - q_{ua})(p_{ub} - q_{ub}) \geq \min(\alpha, 1-\alpha)\min(p_{min}, \Delta)^2/2.
\end{aligned}
$$

Let us assume the error in $(Z^{|}_{ua,ub,bc} + X_{ua}X_{ub}X_{bc} - X_{ua}Y^{|}_{ub,bc} - X_{bc}Y^{|}_{ua,ub})$ is bounded as ϵ_d and the error in $(Y^{|}_{ua,ub} - X_{ua}X_{ub})(Y^{|}_{ub,bc} - X_{ub}X_{bc})$ is bounded as ϵ_n. We have $\epsilon_n \leq 4\epsilon_1$ and $\epsilon_d \leq 3\epsilon_1$ as all the estimators are assumed to have error bounded by ϵ_1.

Therefore, using $|x/y - \hat{x}/\hat{y}| \leq x/y(\delta_x/x + \delta_y/y) + \mathcal{O}(\delta_x\delta_y)$,

$$|\hat{C}^{|}_{ub} - C^{|}_{ub}| \leq \epsilon_c := \mathcal{O}\left(\frac{\epsilon_n}{\min(\alpha,1-\alpha)^2\min(p_{min},\Delta)^4} + \frac{\epsilon_d}{\min(\alpha,1-\alpha)^2 p_{min}\min(p_{min},\Delta)^2}\right)$$
$$= \mathcal{O}(\epsilon_1/\min(\alpha,1-\alpha)^2\min(p_{min},\Delta)^4).$$

Using the above bound in the expression of p_{ua} we can obtain,

$$\begin{aligned}
&|\hat{p}_{ua} - p_{ua}| \leq |\hat{X}_{ua} - X_{ua}| + \tfrac{(1-2\alpha)}{2\alpha}|\hat{C}^{|}_{ua} - C^{|}_{ua}| + \ldots \\
&+ \Big|\sqrt{\Big(\tfrac{(1-2\alpha)}{2\alpha}\hat{C}^{|}_{ua}\Big)^2 + \tfrac{(1-\alpha)}{\alpha}\hat{C}^{|}_{ua}\hat{X}_{ua}} - \sqrt{\Big(\tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ua}\Big)^2 + \tfrac{(1-\alpha)}{\alpha}C^{|}_{ua}X_{ua}}\Big| \\
&\leq |\hat{X}_{ua} - X_{ua}| + \tfrac{(1-2\alpha)}{2\alpha}|\hat{C}^{|}_{ua} - C^{|}_{ua}| + \ldots \\
&+ \frac{\Big(\tfrac{(1-2\alpha)}{2\alpha}\Big)^2|\hat{C}^{|}_{ua} - C^{|}_{ua}|(\hat{C}^{|}_{ua} + C^{|}_{ua}) + \tfrac{(1-\alpha)}{\alpha}|\hat{C}^{|}_{ua}\hat{X}_{ua} - C^{|}_{ua}X_{ua}|}{\sqrt{\Big(\tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ua}\Big)^2 + \tfrac{(1-\alpha)}{\alpha}C^{|}_{ua}X_{ua}}} \\
&\leq \epsilon_1 + \tfrac{(1-2\alpha)}{2\alpha}\epsilon_c + \tfrac{2}{(1-\alpha)\min(p_{min},\Delta)}\left(2\left(\tfrac{(1-2\alpha)}{2\alpha}\right)^2\epsilon_c + \tfrac{(1-\alpha)}{\alpha}(\epsilon_1+\epsilon_c)\right) + o(\epsilon_1) + o(\epsilon_c) \\
&\leq \mathcal{O}(\epsilon_1/\min(p_{min},\Delta)\alpha(1-\alpha)) + \mathcal{O}(\epsilon_c/\min(p_{min},\Delta)\alpha^2(1-\alpha)) + o(\epsilon_1) + o(\epsilon_c)
\end{aligned}$$

Therefore, using the estimate of ϵ_c we obtain,

$$|\hat{p}_{ua} - p_{ua}| \leq \mathcal{O}\left(\epsilon_1/\min(p_{min},\Delta)^5\alpha\min(\alpha,1-\alpha)^3\right).$$

Switching α and $(1-\alpha)$ gives us the same bounds for $|\hat{q}_{ua} - q_{ua}|$. □

In the above derivation we have used $\sqrt{\Big(\tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ua}\Big)^2 + \tfrac{(1-\alpha)}{\alpha}C^{|}_{ua}X_{ua}} \geq$

$(1-\alpha)\min(p_{min},\Delta)/2$. We now derive the above inequality.

$$
\begin{aligned}
&|\sqrt{\left(\tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ua}\right)^2+\tfrac{(1-\alpha)}{\alpha}C^{|}_{ua}X_{ua}}| = |p_{ua}-X_{ua}-\tfrac{(1-2\alpha)}{2\alpha}C^{|}_{ua}| \\
&= |(1-\alpha)(p_{ua}-q_{ua})-\tfrac{(1-2\alpha)(1-\alpha)(p_{ua}-q_{ua})^2}{2((1-\alpha)p_{ua}+\alpha q_{ua})}| \\
&\geq \begin{cases} (1-\alpha)\min(p_{min},\Delta), (\alpha\geq 1/2\wedge p_{ua}\geq q_{ua})\vee(\alpha<1/2\wedge p_{ua}<q_{ua}) \\ (1-\alpha)\min(p_{min},\Delta)|1-\tfrac{(1-2\alpha)}{2(1-\alpha)}|, (\alpha<1/2\wedge p_{ua}\geq q_{ua}) \\ (1-\alpha)\min(p_{min},\Delta)|1-\tfrac{(2\alpha-1)}{2\alpha}|, (\alpha\geq 1/2\wedge p_{ua}<q_{ua}), \end{cases}
\end{aligned}
$$

Finally, using union bound on all the estimators involved accross all possible edges, we can obtain the error bound in the following Theorem C.5.5.

Theorem C.5.5. *Suppose Condition 1 and 2 are true, there exists an algorithm that runs on epidemic cascades over a mixture of two undirected, weighted graphs $G_1=(V,E_1)$ and $G_2=(V,E_2)$, and recovers the edge weights corresponding to each graph up to precision ϵ in time $O(N^2)$ and sample complexity $O\left(\frac{N\log N}{\epsilon^2\Delta^4}\right)$ for $\alpha=1/2$ and $O\left(\frac{N\log N}{\epsilon^2\Delta^{10}\min(\alpha,1-\alpha)^8}\right)$ for general $\alpha\in(0,1),\alpha\neq 1/2$, where $N=|V|$.*

Bibliography

[1] Imported from Combinatorial Pure Exploration with Continuous and Separable Reward Functions and Its Applications (Extended Version) http://arxiv.org/abs/1805.01685v1.

[2] Bruno Abrahao, Flavio Chierichetti, Robert Kleinberg, and Alessandro Panconesi. Trace complexity of network inference. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 491, 2013.

[3] Edouard Amouroux, Stéphanie Desvaux, and Alexis Drogoul. Towards virtual epidemiology: an agent-based approach to the modeling of h5n1 propagation and persistence in north-vietnam. In *Pacific Rim International Conference on Multi-Agents*, pages 26–33. Springer, 2008.

[4] Ery Arias-castro, Emmanuel J Candès, and Arnaud Durand. Detection of an anomalous cluster in a network. *The Annals of Statistics*, 39(1):278–304, 2011.

[5] Ery Arias-Castro, Emmanuel J. Candès, Hannes Helgason, and Ofer Zeitouni. Searching for a trail of evidence in a maze. *Annals of Statistics*, 36(4):1726–1757, 2008.

[6] Ery Arias-castro and S T Nov. Detecting a Path of Correlations in a Network. pages 1–12.

[7] Sivaraman Balakrishnan, Martin J Wainwright, Bin Yu, et al. Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77–120, 2017.

[8] Daniel Bernoulli and Sally Blower. An attempt at a new analysis of the mortality caused by smallpox and of the advantages of inoculation to prevent it. *Reviews in medical virology*, 14:275–288, 2004.

[9] Abhijit Bose, Xin Hu, Kang G Shin, and Taejoon Park. Behavioral Detection of Malware on Mobile Handsets. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 225–238, New York, NY, USA, 2008. ACM.

[10] Daniel G Brown. How I wasted too long finding a concentration inequality for sums of geometric variables. *Found at https://cs. uwaterloo. ca/˜ browndg/negbin. pdf*, 6.

[11] A. Cayley. A theorem on trees. In *Collected Mathematical Papers Vol. 13*, pages 26–28. Cambridge University Press, 1897.

[12] Yudong Chen, Xinyang Yi, and Constantine Caramanis. Convex and nonconvex formulations for mixed regression with two components: Minimax optimal rates. *IEEE Transactions on Information Theory*, 64(3):1738–1766, 2017.

[13] Justin Cheng, Lada A. Adamic, P. Alex Dow, Jon Kleinberg, and Jure Leskovec. Can Cascades be Predicted? In *Proceedings of the 23rd international conference on World wide web (WWW' 14)*, 2014.

[14] Edward Choi, Nan Du, Robert Chen, Le Song, and Jimeng Sun. Constructing disease network and temporal progression model via context-sensitive hawkes process. In *2015 IEEE International Conference on Data Mining*, pages 721–726. IEEE, 2015.

[15] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[16] Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. Ten steps of em suffice for mixtures of two gaussians. In *30th Annual Conference on Learning Theory*, 2017.

[17] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, page 201517441, 2016.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal ofthe Royal Statistical Society*, 39(1):1–38, 1977.

[19] Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. List-decodable robust mean estimation and learning mixtures of spherical gaussians. In

Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pages 1047–1060. ACM, 2018.

[20] Ming Dong, Bolong Zheng, Nguyen Quoc Viet Hung, Han Su, and Guohui Li. Multiple rumor source detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 569–578, 2019.

[21] Sever S Dragomir and V Gluscevic. Some inequalities for the kullback-leibler and x^2- distances in information theory and applications. *RGMIA research report collection*, 3(2):199–210, 2000.

[22] Kimon Drakopoulos, Asuman Ozdaglar, and John N. Tsitsiklis. An efficient curing policy for epidemics on graphs. *arXiv preprint arXiv:1407.2241*, (December):1–10, 2014.

[23] Kimon Drakopoulos, Asuman Ozdaglar, and John N. Tsitsiklis. A lower bound on the performance of dynamic curing policies for epidemics on graphs. (978):3560–3567, 2015.

[24] Kimon Drakopoulos, Asuman Ozdaglar, and John N. Tsitsiklis. When is a network epidemic hard to eliminate? pages 1–17, 2015.

[25] Jalal Etesami, Negar Kiyavash, Kun Zhang, and Kushagra Singhal. Learning network of multivariate hawkes processes: A time series approach. *arXiv preprint arXiv:1603.04319*, 2016.

[26] Giulia Fanti, Peter Kairouz, Sewoong Oh, Kannan Ramchandran, and Pramod Viswanath. Rumor source obfuscation on irregular trees. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science (SIGMETRICS' 16)*, pages 153–164. ACM, 2016.

[27] Giulia Fanti, Peter Kairouz, Sewoong Oh, Kannan Ramchandran, and Pramod Viswanath. Hiding the Rumor Source. *IEEE Transactions on Information Theory*, 63(10):6679–6713, 2017.

[28] Giulia Fanti, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Spy vs. Spy: Rumor Source Obfuscation. *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS' 14)*, pages 271–284, 2015.

[29] Giulia Fanti, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Spy vs. spy: Rumor source obfuscation. In *ACM SIGMETRICS Performance Evaluation Review*, volume 43, pages 271–284. ACM, 2015.

[30] Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. Fake News Mitigation via Point Process Based Intervention. In *Proceedings of the 34th International Conference on Machine Learning (ICML' 17)*, 2017.

[31] Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. Fake news mitiga-

tion via point process based intervention. *arXiv preprint arXiv:1703.07823*, 2017.

[32] Robert Gallager. Stochastic Processes: 9 - Random Walks, Large Deviations, and Martingales. *Stochastic Processes*, 2013.

[33] Ayalvadi Ganesh, Laurent Massoulié, and Don Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1455–1466. IEEE, 2005.

[34] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[35] Manuel Gomez-rodriguez, Jure Leskovec, and Andreas Krause. Inferring Networks of Diffusion and Influence. In *ACM Transactions on Knowledge Discovery from Data (TKDD' 12)*, volume 5, 2012.

[36] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Schölkopf. Structure and Dynamics of Information Pathways in Online Media. In *6th International Conference on Web Search and Data Mining (WSDM 2013)*, 2013.

[37] David Grimmett, Geoffrey Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[38] Ilkka Hanski. Metapopulation dynamics. *Nature*, 396(6706):41–49, 1998.

[39] Jessica Hoffmann, Soumya Basu, Surbhi Goel, and Constantine Caramanis. Learning mixture of graphs from epidemic cascades. *Proceedings of the 29th International Conference of Machine Learning - ICML*, 2020.

[40] Jessica Hoffmann and Constantine Caramanis. The Cost of Uncertainty in Curing Epidemics. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS' 18)*, 2(2):11–13, 2018.

[41] Jessica Hoffmann and Constantine Caramanis. Learning graphs from noisy epidemic cascades. *arXiv preprint arXiv:1903.02650*, 2019.

[42] Lars Hufnagel, Dirk Brockmann, and Theo Geisel. Forecast and control of epidemics in a globalized world. *Proceedings of the National Academy of Sciences*, 101(42):15124–15129, 2004.

[43] Tomoharu Iwata, Amar Shah, and Zoubin Ghahramani. Discovering Latent Influence in Online Social Activities via Shared Cascade Poisson Processes. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD' 13)*, 2013.

[44] Grégoire Jacob, Hervé Debar, and Eric Filiol. Behavioral detection of malware: from a survey towards an established taxonomy. *Journal in Computer Virology*, 4(3):251–266, 2008.

[45] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM*

SIGKDD international conference on Knowledge discovery and data mining - KDD '03, 2003.

[46] Justin Khim and Po-Ling Loh. Permutation Tests for Infection Graphs. pages 1–28, 2017.

[47] Justin Khim and Po-Ling Loh. A theory of maximum likelihood for weighted infection graphs. pages 1–47, 2018.

[48] Ryota Kobayashi and Renaud Lambiotte. Tideh: Time-dependent hawkes process for predicting retweet dynamics. In *Tenth International AAAI Conference on Web and Social Media*, 2016.

[49] Naimisha Kolli and Balakrishnan Narayanaswamy. Influence maximization from cascade information traces in complex networks in the absence of network structure. *IEEE Transactions on Computational Social Systems*, 6(6):1147–1155, 2019.

[50] Jeongyeol Kwon and Constantine Caramanis. Em converges for a mixture of many linear regressions. *arXiv preprint arXiv:1905.12106*, 2019.

[51] Jeongyeol Kwon, Wei Qian, Constantine Caramanis, Yudong Chen, and Damek Davis. Global Convergence of the EM Algorithm for Mixtures of Two Component Linear Regression. XX:1–57, 2019.

[52] Jeongyeol Kwon, Wei Qian, Constantine Caramanis, Yudong Chen, and Damek Davis. Global convergence of the em algorithm for mixtures of

two component linear regression. In *32nd Annual Conference on Learning Theory*, pages 2055–2110. PMLR, 2019.

[53] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective Outbreak Detection in Networks. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07)*, page 420, 2007.

[54] Xiang Li, Jian-Bo Wang, and Cong Li. Towards identifying and predicting spatial epidemics on complex meta-population networks. In *Temporal Network Epidemiology*, pages 129–160. Springer, 2017.

[55] Shenghua Liu, Huawei Shen, Houdong Zheng, Xueqi Cheng, and Xiangwen Liao. Ct lis: Learning influences and susceptibilities through temporal behaviors. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6):1–21, 2019.

[56] Yishay Mansour. Lecture 5 : Lower Bounds using Information Theory Tools Distance between Distributions KL-Divergence. 2011.

[57] Benjamin Mark, Garvesh Raskutti, and Rebecca Willett. Network estimation from point process data. *IEEE Transactions on Information Theory*, 65(5):2953–2975, 2018.

[58] Eli A. Meirom, Chris Milling, Constantine Caramanis, Shie Mannor, Ariel Orda, and Sanjay Shakkottai. Localized epidemic detection in networks

with overwhelming noise. pages 1–27, 2014.

[59] Eli A Meirom, Chris Milling, Constantine Caramanis, Shie Mannor, Sanjay Shakkottai, and Ariel Orda. Localized epidemic detection in networks with overwhelming noise. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS' 15)*, volume 43, pages 441–442. ACM, 2015.

[60] Chris Milling, Constantine Caramanis, Shie Mannor, and Sanjay Shakkottai. Network Forensics : Random Infection vs Spreading Epidemic. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems (SIGMETRICS' 12)*, 2012.

[61] Chris Milling, Constantine Caramanis, Shie Mannor, and Sanjay Shakkottai. Distinguishing Infections on Different Graph Topologies. *IEEE Transactions on Information Theory*, 61(6):3100–3120, 2015.

[62] Chris Milling, Constantine Caramanis, Shie Mannor, and Sanjay Shakkottai. Local detection of infections in heterogeneous networks. *Proceedings - IEEE INFOCOM*, 26:1517–1525, 2015.

[63] Praneeth Netrapalli and Sujay Sanghavi. Learning the Graph of Epidemic Cascades. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems (SIGMETRICS' 12)*, pages 211–222, 2012.

[64] Donald J Newman. The double dixie cup problem. *The American Mathematical Monthly*, 67(1):58–61, 1960.

[65] Mark E. J. Newman. Spread of epidemic disease on networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 66(1), 2002.

[66] Richard S Ostfeld, Gregory E Glass, and Felicia Keesing. Spatial epidemiology: an emerging (or re-emerging) discipline. *Trends in ecology & evolution*, 20(6):328–336, 2005.

[67] Han-Ching Ou, Arunesh Sinha, Sze-Chuan Suen, Andrew Perrault, and Milind Tambe. Who and when to screen: Multi-round active screening for recurrent infectious diseases under uncertainty, 2019.

[68] Padmavathi Patlolla, Vandana Gunupudi, Armin R Mikler, and Roy T Jacob. Agent-based simulation tools in computational epidemiology. In *International Workshop on Innovative Internet Community Systems*, pages 212–223. Springer, 2004.

[69] Liudmila Prokhorenkova, Alexey Tikhonov, and Nelly Litvak. Learning clusters through information diffusion. In *The World Wide Web Conference*, pages 3151–3157, 2019.

[70] Ivan N. Sanov. On the Probability of Large Deviations of Random Variables, 1961.

[71] Devavrat Shah and Tauhid Zaman. Detecting sources of computer viruses in networks: theory and experiment. In *ACM SIGMETRICS Performance Evaluation Review*, volume 38, pages 203–214. ACM, 2010.

[72] Devavrat Shah and Tauhid Zaman. Rumors in a Network : Who ' s the Culprit ? *IEEE Transactions on information theory*, 57(8):1–43, 2010.

[73] Devavrat Shah and Tauhid Zaman. Rumors in a network: Who's the culprit? *IEEE Transactions on information theory*, 57(8):5163–5181, 2011.

[74] Devavrat Shah and Tauhid Zaman. Rumor centrality: a universal source detector. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 199–210. ACM, 2012.

[75] James Sharpnack, Alessandro Rinaldo, and Aarti Singh. Changepoint detection over graphs with the spectral scan statistic. *arXiv preprint*, 31:1–14, 2012.

[76] Sam Spencer and R Srikant. On the impossibility of localizing multiple rumor sources in a line graph. *ACM SIGMETRICS Performance Evaluation Review*, 43(2):66–68, 2015.

[77] Anirudh Sridhar and H Vincent Poor. Sequential estimation of network cascades. *arXiv preprint arXiv:1912.03800*, 2019.

[78] Lin Wang and Xiang Li. Spatial epidemiology of networked metapopulation: An overview. *Chinese Science Bulletin*, 59(28):3511–3522, 2014.

[79] Shengling Wang, Shasha Chen, Xiuzhen Cheng, Weifeng Lv, and Jiguo Yu. Analysis of antagonistic dynamics for rumor propagation. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1253–1263. IEEE, 2019.

[80] Zhaoxu Wang, Wenxiang Dong, Wenyi Zhang, and Chee Wei Tan. Rumor source detection with multiple observations: Fundamental limits and algorithms. In *ACM SIGMETRICS Performance Evaluation Review*, volume 42, pages 1–13. ACM, 2014.

[81] Liang Wu and Huan Liu. Tracing Fake-News Footprints: Characterizing Social Media Messages by How They Propagate. In *(WSDM 2018) The 11th ACM International Conference on Web Search and Data Mining*, 2018.

[82] Yujia Xie, Haoming Jiang, Feng Liu, Tuo Zhao, and Hongyuan Zha. Meta learning with relational information for short sequences. In *Advances in Neural Information Processing Systems*, pages 9901–9912, 2019.

[83] Ji Xu, Daniel J Hsu, and Arian Maleki. Global analysis of expectation maximization for mixtures of two gaussians. In *Advances in Neural Information Processing Systems*, pages 2676–2684, 2016.

[84] Wen Yan, Po-Ling Loh, Chunguo Li, Yongming Huang, and Luxi Yang. Conquering the worst case of infections in networks. *IEEE Access*, 2019.

[85] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating minimization for mixed linear regression. In *International Conference on Machine Learning*, pages 613–621, 2014.

[86] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Solving a mixture of many random linear equations by tensor decomposition and alternating minimization. *arXiv preprint arXiv:1608.05749*, 2016.

[87] Ali Zarezade, Ali Khodadadi, Mehrdad Farajtabar, Hamid R Rabiee, and Hongyuan Zha. Correlated Cascades : Compete or Cooperate. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 238–244, 2017.

[88] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. SEISMIC: A Self-Exciting Point Process Model for Predicting Tweet Popularity. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*, 2015.

[89] Lili Zheng, Garvesh Raskutti, Rebecca Willett, and Benjamin Mark. Context-dependent self-exciting point processes: models, methods, and risk bounds in high dimensions. *arXiv preprint arXiv:2003.07429*, 2020.

Vita

Jessica Hélène Hoffmann was born in Philadelphia, Pennsylvania on December 30^{th}, 1991, the daughter of Prof. Isabelle Berrebi-Hoffmann and Claude Hoffmann. She then grew up in Paris, France, where she received two Bachelor of Science degrees, one in Physics and one in Computer Science, from École Normale Supérieure de Paris, followed by a Master of Science in Applied Mathematics (Computer Vision and Machine Learning) from École Normale Supérieure de Cachan, while on a full scholarship. After a year of travel, she joined the Ph.D. program at the University of Texas at Austin in August, 2015. Both during and prior to her doctoral work, she was fortunate to complete internships at the Institut Pierre et Marie Curie in Paris with Prof. Jean-Baptiste Manneville, the Univerisity of California at Berkeley with Prof. Pieter Abbeel, Tel Aviv University with Prof. Ron Shamir, Quantcast, and Google.

Permanent address: 4510 W. Guadalupe St., #C202
Austin, Texas 78751

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.