Trace Complexity of Network Inference

Bruno Abrahao[*]
Department of Computer Science
Cornell University
Ithaca, NY, 14850, USA
abrahao@cs.cornell.edu

Flavio Chierichetti[†]
Dipartimento di Informatica
Sapienza University
Rome, Italy
flavio@di.uniroma1.it

Robert Kleinberg[‡]
Department of Computer Science
Cornell University
Ithaca, NY, 14850, USA
rdk@cs.cornell.edu

Alessandro Panconesi[†]
Dipartimento di Informatica
Sapienza University
Rome, Italy
ale@di.uniroma1.it

ABSTRACT

The network inference problem consists of reconstructing the edge set of a network given traces representing the chronology of infection times as epidemics spread through the network. This problem is a paradigmatic representative of prediction tasks in machine learning that require deducing a latent structure from observed patterns of activity in a network, which often require an unrealistically large number of resources (e.g., amount of available data, or computational time). A fundamental question is to understand which properties we can predict with a reasonable degree of accuracy with the available resources, and which we cannot. We define the *trace complexity* as the number of distinct traces required to achieve high fidelity in reconstructing the topology of the unobserved network or, more generally, some of its properties. We give algorithms that are competitive with, while being simpler and more efficient than, existing network inference approaches. Moreover, we prove that our algorithms are nearly optimal, by proving an information-theoretic lower bound on the number of traces that an optimal inference algorithm requires for performing this task in the general case. Given these strong lower bounds, we turn our attention to special cases, such as trees and bounded-degree graphs, and to property recovery tasks, such as reconstructing the degree distribution without inferring the network. We show that these problems require a much smaller (and more realistic) number of traces, making them potentially solvable in practice.

[*]Supported by AFOSR grant FA9550-09-1-0100, by a Cornell University Graduate School Travel Fellowship, and by a Google Award granted to Alessandro Panconesi.

[†]Supported by two Google Faculty Research Awards and by the MULTIPLEX project (EU-FET-317532).

[‡]Supported by AFOSR grant FA9550-09-1-0100.

Categories and Subject Descriptors

I.5.1 [**Computing Methodology**]: Pattern Recognition — *Design Methodology*

Keywords

Network inference, Independent Cascade Model, Network Epidemics, Sampling Complexity

1. INTRODUCTION

Many technological, social, and biological phenomena are naturally modeled as the propagation of a contagion through a network. For instance, in the blogosphere, "memes" spread through an underlying social network of bloggers [1], and, in biology, a virus spreads over a population through a network of contacts [2]. In many such cases, an observer may not directly probe the underlying network structure, but may have access to the sequence of times at which the nodes are infected. Given one or more such records, or *traces*, and a probabilistic model of the epidemic process, we can hope to deduce the underlying graph structure or at least estimate some of its properties. This is the *network inference* problem, which researchers have studied extensively in recent years [1, 7, 12, 13, 20].

In this paper we focus on the number of traces that network inference tasks require, which we define as the *trace complexity* of the problem. Our work provides inference algorithms with rigorous upper bounds on their trace complexity, along with information-theoretic lower bounds. We consider network inference tasks under a diffusion model presented in [13], whose suitability for representing real-world cascading phenomena in networks is supported by empirical evidence. In short, the model consists of a random cascade process that starts at a single node of a network, and each edge $\{u, v\}$ independently propagates the epidemic, once u is infected, with probability p after a random *incubation time*.

Overview of results. In the first part of this paper, we focus on determining the number of traces that are necessary and/or sufficient to perfectly recover the edge set of the whole graph with high probability. We present algorithms and (almost) matching lower bounds for exact inference by showing that in the worst case, $\Omega(n\Delta^{1-\epsilon})$ traces are necessary and $O(n\Delta \log n)$ traces are sufficient, where n is the number of nodes in the network and Δ is its maximum degree. In the second part, we consider a natural line of investigation, given the preceding strong lower bounds, where we ask whether exact inference is possible using a smaller number of traces for special classes of networks that frequently arise in the analysis of social and information networks. Accordingly, we present improved algorithms and trace complexity bounds for two such cases. We give a very simple and natural algorithm for exact inferences of trees that uses only $O(\log n)$ traces.[1] To further pursue this point, we give an algorithm that exactly reconstructs graphs of degree bounded by Δ using only $O(\mathrm{poly}(\Delta) \log n)$ traces, under the assumption that epidemics always spread throughout the whole graph. Finally, given that recovering the topology of a hidden network in the worst case requires an impractical number of traces, a natural question is whether some non-trivial property of the network can be accurately determined using a moderate number of traces. Accordingly, we present a highly efficient algorithm that, using vastly fewer traces than are necessary for reconstructing the entire edge set, reconstructs the degree distribution of the network with high fidelity by using $O(n)$ traces.

The information contained in a trace. Our asymptotic results also provide some insight into the usefulness of information contained in a trace. Notice that the first two nodes of a trace unambiguously reveal one edge — the one that connects them. As we keep scanning a trace the signal becomes more and more blurred: the third node could be a neighbor of the first or of the second node, or both. The fourth node could be the neighbor of any nonempty subset of the first three nodes, and so on. The main technical challenge in our context is whether we can extract any useful information from the *tail* of a trace, i.e., the suffix consisting of all nodes from the second to the last. As it turns out, our lower bounds show that, for perfect inference on general connected graphs, the answer is "no": we show that the *First-Edge algorithm*, which just returns the edges corresponding to the first two nodes in each trace and ignores the rest, is essentially optimal. This limitation precludes optimal algorithms with practical trace complexity[2]. This result motivates further exploration of trace complexity for special-case graphs. Accordingly, for trees and bounded degree graphs, we illustrate how the tail of traces can be extremely useful for network inference tasks.

Our aforementioned algorithms for special-case graphs make use of maximum likelihood estimation (MLE) but in different ways. Previous approaches, with which we compare our results, have also employed MLE for network inference. For instance, NETINF [13] is an algorithm that attempts to reconstruct the network from a set of independent traces by exploring a submodular property of its MLE formulation. Another example, and closest to ours, is the work by Netrapalli and Sangahvi [20], whose results include qualitatively similar bounds on trace complexity in a quite different epidemic model.

Turning our attention back to our algorithms, our tree reconstruction algorithm performs global likelihood maximization over the entire graph, like the NETINF algorithm [13], whereas our bounded-degree reconstruction algorithm, like the algorithm in [20], performs MLE at each individual vertex. Our algorithms and analysis techniques, however, differ markedly from those of [13] and [20], and may be of independent interest.

In the literature on this rapidly expanding topic, researchers have validated their findings using small or stylized graphs and a relatively large number of traces. In this work, we aim to provide, in the same spirit as [20], a formal and rigorous understanding of the potentialities and limitations of algorithms that aim to solve the network inference problem.

This paper is organized as follows. Section 2 presents an overview of previous approaches to network learning. Section 3 presents the cascade model we consider throughout the paper. Section 4 deals with the *head of the trace*: it presents the First-Edge algorithm for network inference, shows that it is essentially optimal in the worst case, and shows how the first edges' timestamps can be used to guess the degree distribution of the network. Section 5, instead, deals with the *tail of the trace*: it presents efficient algorithms for perfect reconstruction of the topology of trees and of bounded degree networks. Section 6 presents an experimental analysis that compares ours and existing results through the lens of trace complexity. Finally, Section 7 offers our conclusions.

2. RELATED WORK

Network inference has been a highly active area of investigation in data mining and machine learning [1, 7, 12, 13, 20]. It is usually assumed that an event initially activates one or more nodes in a network, triggering a cascading process, e.g., bloggers acquire a piece of information that interests other bloggers [15], a group of people are the first infected by a contagious virus [2], or a small group of consumers are the early adopters of a new piece of technology that subsequently becomes popular [22]. In general, the process spreads like an epidemic over a network (i.e., the network formed by blog readers, the friendship network, the coworkers network). Researchers derive observations from each cascade in the form of *traces* — the identities of the people that are activated in the process and the timestamps of their activation. However, while we do see traces, we do not directly observe the network over which the cascade spreads. The network inference problem consists of recovering the underlying network using the epidemic data.

In this paper we study the cascade model that Gomez-Rodrigues et al. [13] introduced, which consists of a variation of the independent cascade model [16]. Gomez-Rodrigues et al. propose NETINF, a maximum likelihood algorithm, for network reconstruction. Their method is evaluated under the exponential and power-law distributed incubation times. In our work, we restrict our analysis to the case where the incubation times are exponentially distributed as this makes for a rich arena of study.

Gomez-Rodrigues et al. have further generalized the model to include different transmission rates for different edges and a broader collection of waiting times distributions [12, 19]. Later on, Du et al. [7] proposed a kernel-based method that is able to recover the network without prior assumptions on the waiting time distributions. These methods have significantly higher computational costs than NETINF, and, therefore, than ours. Nevertheless, experiments on real and synthetic data show a marked improvement in accuracy, in addition to gains in flexibility. Using a more combinatorial approach, Gripon and Rabbat [14] consider the problem of reconstructing a graph from traces defined as sets of unordered nodes, in which the nodes that appear in the same trace are connected by a path containing exactly the nodes in the trace. In this work, traces of size three are considered, and the authors identify necessary and sufficient conditions to reconstruct graphs in this setting.

The performance of network inference algorithms is dependent on the amount of information available for the reconstruction, i.e., the number and length of traces. The dependency on the number of traces have been illustrated in [7], [12], and [13] by plotting the performance of the algorithms against the number of available traces. Nevertheless, we find little research on a rigorous analysis

[1]All inference results in this paper hold with high probability.

[2]On the other hand, the use of short traces may not be only a theoretical limitation, given the real world traces that we observe in modern social networks. For example, Bakshy et al. [3] report that most cascades in Twitter (`twitter.com`) are short, involving one or two hops.

of this dependency, with the exception of one paper [20] that we now discuss.

Similarly to our work, Netrapalli and Sangahvi [20] present quantitative bounds on trace complexity in a quite different epidemic model. The model studied in [20] is another variation of the independent cascade model. It differs from the model we study in a number of key aspects, which make that model a simplification of the model we consider here. For instance, (i) [20] assumes a cascading process over discrete time steps, while we assume continuous time (which has been shown to be a realistic model of several real-world processes [13]), (ii) the complexity analyzed in [20] applies to a model where nodes are active for a single time step — once a node is infected, it has a single time step to infect its neighbors, after which it becomes permanently inactive. The model we consider does not bound the time that a node can wait before infecting a neighbor. Finally, (iii) [20] rely crucially on the "correlation decay" assumption, which implies – for instance — that each node can be infected during the course of the epidemics by *less* than 1 neighbor in expectation. The simplifications in the model presented by [20] make it less realistic — and, also, make the inference task significantly easier than the one we consider here.

We believe that our analysis introduces a rigorous foundation to assess the performance of existing and new algorithms for network inference. In addition, to the best of our knowledge, our paper is the first to study how different parts of the trace can be useful for different network inference tasks. Also, it is the first to study the trace complexity of special case graphs, such as bounded degree graphs, and for reconstructing non-trivial properties of the network (without reconstructing the network itself), such as the node degree distribution.

3. CASCADE MODEL

The cascade model we consider is defined as follows. It starts with one activated node, henceforth called the *source* of the epidemic, which is considered to be activated, without loss of generality, at time $t = 0$.

As soon as a node u gets activated, for each neighbor v_i, u flips an independent coin: with probability p it will start a countdown on the edge $\{u, v_i\}$. The length of the countdown will be a random variable distributed according to $\mathrm{Exp}(\lambda)$ (exponential[3] with parameter λ). When the countdown reaches 0, that edge is *traversed* — that is, that epidemic reaches v_i via u.

The "trace" produced by the model will be a sequence of tuples (node v, $t(v)$) where $t(v)$ is the first time at which the epidemics reaches v.

In [13], the source of the epidemics is chosen uniformly at random from the nodes of the network. In general, though, the source can be chosen arbitrarily[4].

The cascade process considered here admits a number of equivalent descriptions. The following happens to be quite handy: independently for each edge of G, remove the edge with probability $1 - p$ and otherwise assign a random edge length sampled from $\mathrm{Exp}(\lambda)$. Run Dijkstra's single-source shortest path algorithm on the subgraph formed by the edges that remain, using source s and the sampled edge lengths. Output vertices in the order they are discovered, accompanied by a timestamp representing the shortest path length.

[3] [7, 12, 13] consider other random timer distributions; we will mainly be interested in exponential variables as this setting is already rich enough to make for an interesting and extensive analysis.

[4] Choosing sources in a realistic way is an open problem — the data that could offer a solution to this problem seems to be extremely scarce at this time.

4. THE HEAD OF A TRACE

In this section we will deal with the head of a trace — that is, with the edge connecting the first and the second nodes of a trace. We show that, for general graphs, that edge is the only useful information that can be extracted from traces. Moreover, and perhaps surprisingly, this information is enough to achieve close-to-optimal trace complexity, i.e., no network inference algorithm can achieve better performance than a simple algorithm that only extracts the head of the trace and ignores the rest. We analyze this algorithm in the next section.

4.1 The First-Edge Algorithm

The First-Edge algorithm is simple to state. For each trace in the input, it extracts the edge connecting the first two nodes, and adds this edge the guessed edge set, ignoring the rest of the trace. This procedure is not only optimal in trace complexity, but, as it turns out, it is also computationally efficient.

We start by showing that First-Edge is able to reconstruct the full graph with maximum degree Δ using $\Theta(n\Delta \log n)$ traces, under the cascade model we consider.

THEOREM 4.1. *Suppose that the source $s \in V$ is chosen uniformly at random. Let $G = (V, E)$ be a graph with maximum degree $\Delta \le n - 1$. With $\Theta\left(\frac{n\Delta}{p}\log n\right)$ traces, First-Edge correctly returns the graph G with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$.*

PROOF. Let $e = \{u, v\}$ be any edge in E. The probability that a trace starts with u, and continues with v can be lower bounded by $\frac{p}{n\Delta}$, that is, by the product of the probabilities that u is selected as the source, that the edge $\{u, v\}$ is not removed from the graph, and that v is the first neighbor of u that gets infected. Therefore, if we run $c\frac{n\Delta}{p}\ln n$ traces, the probability that none of them starts with the ordered couple of neighboring nodes u, v is at most:

$$\left(1 - \frac{p}{n\Delta}\right)^{\frac{n\Delta}{p}c\ln n} \le \exp(-c\ln n) = n^{-c}.$$

Therefore, the assertion is proved for any constant $c > 2$. □

We notice that a more careful analysis leads to a proof that

$$\Theta\left(\left(\Delta + p^{-1}\right) n \log n\right)$$

traces are enough to reconstruct the whole graph with high probability. To prove this stronger assertion, it is sufficient to show the probability that a specific edge will be the first one to be traversed is at least $\frac{2}{n} \cdot \left(1 - e^{-1}\right) \cdot \min\left(\Delta^{-1}, p\right)$. In fact one can even show that, for each $d \le \Delta$, if the First-Edge algorithm has access to $O\left(\left(d + p^{-1}\right) n \log n\right)$ traces, then it will recover all the edges having at least one endpoint of degree $O(d)$. As we will see in our experimental section, this allows us to reconstruct a large fraction of the edges using a number of traces that is significantly smaller than the maximum degree times the number of nodes.

Finally, we note that the above proof also entails that First-Edge performs as stated for any waiting time distribution (that is, not just for the exponential one). In fact, the only property that we need for the above bounds to hold, is that the first node, and the first neighbor of the first node, are chosen independently and uniformly at random by the process.

4.2 Lower Bounds

Here, we discuss a number of lower bounds for network inference. Due to limited space, we omit proofs in this section[5]. Here we discuss the main ideas underlying these results.

We start by observing that if the source node is chosen adversarially — and, say if the graph is disconnected —no algorithm can reconstruct the graph (traces are trapped in one connected component and, therefore, do not contain any information about the rest of the graph.) Moreover, even if the graph is forced to be connected, by choosing $p = \frac{1}{2}$ (that is, edges are traversed with probability $\frac{1}{2}$) an algorithm will require at least $2^{\Omega(n)}$ traces even if the graph is known to be a path. Indeed, if we select one endpoint as the source, it will take $2^{\Omega(n)}$ trials for a trace to reach the other end of the path, since at each node, the trace flips an unbiased coin and dies out with probability $\frac{1}{2}$.

This is the reason why we need the assumption that the epidemic selects $s \in V$ uniformly at random — we recall that this is also an assumption in [13]. Whenever possible, we will consider more realistic assumptions, and determine how this changes the trace complexity of the reconstruction problem.

We now turn our attention to our main lower bound result. Namely, for $p = 1$, and assuming that the source is chosen uniformly at random, we need $\omega(n\Delta^{1-\epsilon})$ traces to reconstruct the graph.

First, let G_0 be the clique on the node set $V = \{1, \ldots, n\}$, and let G_1 be the clique on V minus the edge $\{1, 2\}$.

Suppose that Nature selects the unknown graph uniformly at random in the set $\{G_0, G_1\}$. We will show that with $O(n^{2-\epsilon})$, the probability that we are able to guess the unknown graph is $\frac{1}{2} + o(1)$ — that is, flipping a coin is close to being the best one can do for guessing the existence of the edge $\{1, 2\}$.

Before embarking on this task, though, we show that this result directly entails that $O(n\Delta^{1-\epsilon})$ traces are not enough for reconstruction even if the graph has maximum degree Δ, for each $1 \leq \Delta \leq n - 1$. Indeed, let the graph G'_0 be composed of a clique on $\Delta + 1$ nodes, and of $n - \Delta - 1$ disconnected nodes. Let G'_1 be composed of a clique on $\Delta + 1$ nodes, minus an edge, and of $n - \Delta - 1$ disconnected nodes. Then, due to our yet-unproven lower bound, we need at least $\omega(\Delta^{2-\epsilon})$ traces to start in the large connected component for the reconstruction to succeed. The probability that a trace starts in the large connected component is $O\left(\frac{\Delta}{n}\right)$. Hence, we need at least $\omega(n\Delta^{1-\epsilon})$ traces.

We now highlight some of the ideas we used to prove the $\Omega(n^{2-\epsilon})$ lower bound. First, we show a technical lemma that proves that the random ordering of nodes produced by a trace in G_0 is uniform at random, and that the random ordering produced by a trace in G_1 is "close" to being uniform at random.

LEMMA 4.2. *Let π_0 be the random ordering of nodes produced by the random process on G_0, and π_1 be the random ordering of nodes produced by the random process on G_1. Then, (i) π_0 is a uniform at random permutation over $[n]$; (ii) for each $1 \leq a < b \leq n$, the permutation π_1 conditioned on the vertices $1, 2$ appearing (in an arbitrary order) in the positions a, b, has its other elements chosen uniformly at random; (iii) the probability $p_{a,b}$ that π_1 has the vertices $1, 2$ appearing (in an arbitrary order) in the positions $a < b$ is equal to $p_{a,b} = \frac{1+d_{a,b}}{\binom{n}{2}}$, with $d_{a,b} = -1$ if $a = 1, b = 2$, and $|d_{a,b}| \leq O\left(\frac{\ln n}{n} + \frac{1}{b}\right)$ otherwise; moreover, $\sum_{a=1}^{n-1}\sum_{b=a+1}^{n} d_{a,b} = 0$.*

[5]The proofs we omitted here will appear in an extended version of this paper (in preparation).

The preceding Lemma can be used to prove the following result:

LEMMA 4.3. *If we disregard timestamps, we cannot distinguish G_0 and G_1 with probability more than $\frac{1}{2} + o(1)$ using only $m = o\left(\frac{n^2}{\log^3 n}\right)$ traces.*

To prove this Lemma, we bound the KL-divergences of the two distributions generated by G_0 and G_1 from all their weighted geometric means, thus obtaining a bound on the Chernoff information [6]. Then we can use the latter bound to show that $\Omega\left(\frac{n^2}{\log^3 n}\right)$ traces are necessary.

The KL-divergence bounds are obtained by leveraging on the strong approximation bounds for the likelihoods of (most) traces given by Lemma 4.2.

We now complement the above lower bound (that holds only if we disregard the timestamps), with a full-fledged lower bound. To do so, we show that under a conditioning having high probability, the probability that a set of traces has higher likelihood in G_0 than in G_1 is $\frac{1}{2} \pm o(1)$.

LEMMA 4.4. *Let a set of $m = n^{2-\epsilon}$ traces be given. Let $\mathcal{W}$ be the waiting times in the traces. There exists an event E such that, (i) for both the unknown graph G_0 and G_1, the probability of E is $1 - o(1)$, moreover, (ii) conditioning on E, the probability that $L_0(\mathcal{W}) > L_1(\mathcal{W})$ is equal to $\frac{1}{2} \pm o(1)$.*

Finally, Lemma 4.3 and Lemma 4.4 together allow us to obtain the following Theorem:

THEOREM 4.5. *Suppose that at most $m = n^{2-\epsilon}$ traces are given. Then, no algorithm can correctly guess whether the unknown graph is G_0 or G_1 with probability more than $\frac{1}{2} + o(1)$.*

As already noted, the lower bound in the above Theorem can be easily transformed in a $\Omega(n\Delta^{1-\epsilon})$ lower bound, for any $1 \leq \Delta \leq n - 1$.

4.3 Reconstructing the Degree Distribution

In this section we study the problem of recovering the degree distribution of a hidden network and show that this can be done with $\Omega(n)$ traces while achieving high accuracy, using, again, only the first edge of a trace.

The degree distribution of a network is a characteristic structural property of networks, which influences their dynamics, function, and evolution [21]. Accordingly, many networks, including the Internet and the world wide web exhibit distinct degree distributions [10]. Thus, recovering this property allows us to make inferences about the behavior of processes that take place in these networks, without knowledge of their actual link structure.

Let ℓ traces starting from the same node v be given. For trace i, let t_i be the differences between the time of exposure of v, and the the time of exposure of the second node in the trace.

Recall that in the cascade model, the waiting times are distributed according to an exponential random variable with a known parameter λ. If we have ℓ traces starting at a node v, we aim to estimate the degree of v the time gaps $t_1, \ldots, t_\ell$ between the first node and the second node of each trace.

If v has degree d in the graph, then t_i ($1 \leq i \leq \ell$) will be distributed as an exponential random variable with parameter $d\lambda$ [8]. Furthermore, the sum T of the t_i's, $T = \sum_{i=1}^{\ell} t_i$, is distributed as an Erlang random variable with parameters $(\ell, d\lambda)$ [8].

In general, if X is an Erlang variable with parameters (n, λ), and Y is a Poisson variable with parameter $z \cdot \lambda$, we have that $\Pr[X < z] = \Pr[Y \geq n]$. Then, by using the tail bound for the Poisson distribution [5, 17], we have that the probability that T is at most $(1+\epsilon) \cdot \frac{\ell}{d\lambda}$ is

$$\Pr\left[\text{Pois}\left((1+\epsilon) \cdot \ell\right) \geq \ell\right] \geq 1 - e^{-\Theta\left(\epsilon^2 \ell\right)}.$$

Similarly, the probability that T is at least $(1-\epsilon) \cdot \frac{\ell}{d\lambda}$ is

$$1 - \Pr\left[\text{Pois}((1-\epsilon) \cdot \ell) \geq \ell\right] \geq 1 - e^{-\Theta\left(\epsilon^2 \ell\right)}.$$

We then have:

$$\Pr\left[\left|T - \frac{\ell}{d\lambda}\right| \leq \epsilon \cdot \frac{\ell}{d\lambda}\right] \geq 1 - e^{-\Theta\left(\epsilon^2 \ell\right)}.$$

Let our degree inference algorithm return $\hat{d} = \frac{\ell}{T\lambda}$ as the degree of v. Also, let d be the actual degree of v. We have:

$$\Pr\left[\left|\hat{d} - d\right| \leq \epsilon d\right] \geq 1 - e^{-\Theta\left(\epsilon^2 \ell\right)}.$$

We have then proved the following theorem:

THEOREM 4.6. *Provided that $\Omega\left(\frac{\ln \delta^{-1}}{\epsilon^2}\right)$ traces start from v, the degree algorithm returns a $1 \pm \epsilon$ multiplicative approximation to the degree of v with probability at least $1 - \delta$.*

5. THE TAIL OF THE TRACE

A naïve interpretation of the lower bound for perfect reconstruction, Theorem 4.5, would conclude that the information in the "tail" of the trace — the list of nodes infected after the first two nodes, and their timestamps — is of negligible use in achieving the task of perfect reconstruction. In this section we will see that the opposite conclusion holds for important classes of graphs. We specialize to two such classes, trees and bounded-degree graphs, in both cases designing algorithms that rely heavily on information in the tails of traces to achieve perfect reconstruction with trace complexity $O(\log n)$, an exponential improvement from the worst-case lower bound in Theorem 4.5. The algorithms are quite different: for trees we essentially perform maximum likelihood estimation (MLE) of the entire edge set all at once, while for bounded-degree graphs we run MLE separately for each vertex to attempt to find its set of neighbors, then we combine those sets while resolving inconsistencies.

In Section 6 we provide one more example of an algorithm, which we denote by First-Edge+, that makes use of information in the tail of the trace. Unlike the algorithms in this section, we do not know of a theoretical performance guarantee for First-Edge+ so we have instead analyzed it experimentally.

It is natural to compare the algorithms in this section with the NETINF algorithm [13], since both are based on MLE. While NETINF is a general-purpose algorithm, and the algorithms developed here are limited to special classes of graphs, we believe our approach offers several advantages. First, and most importantly, we offer provable trace complexity guarantees: $\Omega(\log n)$ complete traces suffice for perfect reconstruction of a tree with high probability, and $\Omega(\text{poly}(\Delta) \log n)$ traces suffice for perfect reconstruction of a graph with maximum degree Δ. Previous work has not provided rigorous guarantees on the number of traces required to ensure that algorithms achieve specified reconstruction tasks. Second, our tree reconstruction algorithm is simple (an easy preprocessing step followed by computing a minimum spanning tree) and has worst-case running time $O(n^2 \ell)$, where n is the number of nodes and $\ell = \Omega(\log n)$ is the number of traces, which compares favorably with the running time of NETINF.

5.1 Reconstructing Trees

In this section we consider the special case in which the underlying graph G is a tree, and we provide a simple algorithm that requires $\Omega(\log n)$ complete traces and succeeds in perfect reconstruction with high probability. Intuitively, reconstructing trees is much simpler than reconstructing general graphs for the following reason. As noted in [13], the probability that an arbitrary graph G generates trace T is a sum, over all spanning trees F of G, of the probability that T was generated by an epidemic propagating along the edges of F. When G itself is a tree, this sum degenerates to a single term and this greatly simplifies the process of doing maximum likelihood estimation. In practical applications of the network inference problem, it is unlikely that the latent network will be a tree; nevertheless we believe the results in this section are of theoretical interest and that they may provide a roadmap for analyzing the trace complexity of other algorithms based on maximum likelihood estimation.

Algorithm 1 The tree reconstruction algorithm.

```
Input: A collection $T_1, \ldots, T_\ell$ of complete traces generated by
    repeatedly running the infection process with $p = 1$ on a fixed
    tree.
    Let $t_i(v)$ denote the infection time of node $v$ in trace $T_i$.
Output: An estimate, $\hat{G}$, of the tree.
1: for all pairs of nodes $u, v$ do
2:     Let $c(u, v)$ be the median of the set $\{|t_i(u) - t_i(v)|\}_{i=1}^{\ell}$.
3:     if $\exists$ a node $p$ and a pair of traces $T_i, T_j$ such that $t_i(p) <$
       $t_i(u) < t_i(v)$ and $t_j(p) < t_j(v) < t_j(u)$ then
4:         Set $c(u, v) = \infty$.
5: Output $\hat{G}$ = minimum spanning tree with respect to cost ma-
   trix $c(u, v)$.
```

The tree reconstruction algorithm is very simple. It defines a cost for each edge $\{u, v\}$ as shown in Figure 1, and then it outputs the minimum spanning tree with respect to those edge costs. The most time-consuming step is the test in step 3, which checks whether there is a node p whose infection time precedes the infection times of both u and v in two distinct traces T_i, T_j such that the infection times of u and v are oppositely ordered in T_i and T_j. (If so, then G contains a path from p to u that does not include v, and a path from p to v that does not include u, and consequently $\{u, v\}$ cannot be an edge of the tree G. This justifies setting $c(u, v) = \infty$ in step 4.) To save time, one can use lazy evaluation and perform this test only for pairs u, v that are about to be inserted into the tree.

The analysis of the algorithm is based on the following outline: first, we show that if $\{u, v\}$ is any edge of G, then $c(u, v) < \lambda^{-1}$ with high probability (Lemma 5.1). Second, we show that if $\{u, v\}$ is any edge not in G, then $c(u, v) > \lambda^{-1}$ with high probability (Lemma 5.2). The edge pruning in steps 3 and 4 of the algorithm is vital for attaining the latter high-probability guarantee. When both of these high-probability events occur, it is trivial to see that the minimum spanning tree coincides with G.

LEMMA 5.1. *If $\{u, v\}$ is an edge of the tree G, then Algorithm 1 sets $c(u, v) < \lambda^{-1}$ with probability at least $1 - {c_1}^{\lambda}$, for some absolute constant $c_1 < 1$.*

PROOF. First, note that the algorithm never sets $c(u, v) = \infty$. This is because if one were to delete edge $\{u, v\}$ from G, it would disconnect the graph into two connected components G_u, G_v, containing u and v, respectively. The infection process cannot spread from G_u to G_v or vice-versa without traversing edge $\{u, v\}$. Consequently, for every node $p \in G_u$, the infection time $t_i(u)$ occurs strictly between $t_i(p)$ and $t_i(v)$ in all traces. Similarly, if $p \in G_v$ then the infection time $t_i(v)$ occurs strictly between $t_i(p)$ and $t_i(u)$ in all traces.

Therefore, the value of $c(u,v)$ is equal to the median of $|t_i(u) - t_i(v)|$ over all the traces $T_1, \dots, T_\ell$. In any execution of the infection process, if the first endpoint of edge $\{u,v\}$ becomes infected at time t, then the opposite endpoint receives a timestamp $t+X$ where $X \sim \mathrm{Exp}(\lambda)$. Consequently the random variable $|t_i(u) - t_i(v)|$ is an independent sample from $\mathrm{Exp}(\lambda)$ in each trace. The lemma now follows by an application of Chernoff's bound. $\square$

The proof of the following lemma, while similar to that of the preceding one, is somewhat more involved. It is omitted for space reasons.

LEMMA 5.2. *If $\{u,v\}$ is not an edge of G, then Algorithm 1 sets $c(u,v) > \lambda^{-1}$ with probability at least $1 - c_2 \cdot c_3^\ell$ for some absolute constants $c_2 < \infty$ and $c_3 < 1$.*

Combining Lemmas 5.1 and 5.2, and using the union bound, we find that with probability at least $1 - (n-1)c_1^\ell - \binom{n-1}{2} c_2 c_3^\ell$, the set of pairs (u,v) such that $c(u,v) < \lambda^{-1}$ coincides with the set of edges of the tree G. Whenever the $n-1$ cheapest edges in a graph form a spanning tree, it is always the minimum spanning tree of the graph. Thus, we have proven the following theorem.

THEOREM 5.3. *If G is a tree, then Algorithm 1 perfectly reconstructs G with probability at least $1 - (n-1)c_1^\ell - \binom{n-1}{2} c_2 c_3^\ell$, for some absolute constants $c_1, c_3 < 1$ and $c_2 < \infty$. This probability can be made greater than $1 - 1/n^c$, for any specified $c > 0$, by using $\ell \geq c_4 \cdot c \cdot \log n$ traces, where $c_4 < \infty$ is an absolute constant.*

5.2 Bounded-Degree Graphs

In this section, we show that $O(\mathrm{poly}(\Delta) \log n)$ complete traces suffice for perfect reconstruction (with high probability) when the graph G has maximum degree Δ. In fact, our proof shows a somewhat stronger result: it shows that for any pair of nodes u, v, there is an algorithm that predicts whether $\{u,v\}$ is an edge of G with failure probability at most $1 - 1/n^c$, for any specified constant $c > 0$, and the algorithm requires only $\Omega(\mathrm{poly}(\Delta) \log n)$ independent partial traces in which u and v are both infected. However, for simplicity we will assume complete traces throughout this section.

The basic intuition behind our algorithm can be summarized as follows. To determine if $\{u,v\}$ is an edge of G, we try to reconstruct the entire set of neighbors of u and then test if v belongs to this set. We use the following insight to test whether a candidate set S is equal to the set $N(u)$ of all neighbors of u. Any such set defines a "forecasting model" that specifies a probability distribution for the infection time $t(u)$. To test the validity of the forecast we use a strictly proper scoring rule [11], specifically the logarithmic scoring rule, which is defined formally in the paragraph following Equation (1). Let us say that a set S differs significantly from the set of neighbors of u (henceforth denoted $N(u)$) if the symmetric difference $S \oplus N(u)$ contains a vertex that is infected before u with constant probability. We prove that the expected score assigned to $N(u)$ by the logarithmic scoring rule is at least $\Omega(\Delta^{-4})$ greater than the score assigned to any set that differs significantly from $N(u)$. Averaging over $\Omega(\Delta^4 \log \Delta \log n)$ trials is then sufficient to ensure that all sets differing significantly from $N(u)$ receive strictly smaller average scores.

The scoring rule algorithm thus succeeds (with high probability) in reconstructing a set $R(u)$ whose difference from $N(u)$ is insignificant, meaning that the elements of $R(u) \oplus N(u)$ are usually infected after u. To test if edge $\{u,v\}$ belongs to G, we can now use the following procedure: if the event $t(v) < t(u)$ occurs in a constant fraction of the traces containing both u and v, then we predict that edge $\{u,v\}$ is present if $v \in R(u)$; this prediction must be correct with high probability, as otherwise the element $v \in R(u) \oplus N(u)$ would constitute a significant difference. Symmetrically, if $t(u) < t(v)$ occurs in a constant fraction of the traces containing both u and v, then we predict that edge $\{u,v\}$ is present if $u \in R(v)$.

Algorithm 2 Bounded-degree reconstruction algorithm.

```
Input: An infection rate parameter, $\lambda$.
    A set of vertices, $V$.
    An upper bound, $\Delta$, on the degrees of vertices.
    A collection $T_1, \dots, T_\ell$ of complete traces generated by repeatedly running the infection process on a fixed graph $G$ with vertex set $V$ and maximum degree $\Delta$.
    Let $t_i(v)$ denote the infection time of node $v$ in trace $T_i$.
Output: An estimate, $\hat{G}$, of $G$.
 1: for all nodes $u$ do
 2:     for all sets $S \subseteq V \setminus \{u\}$ of at most $\Delta$ vertices do
 3:         for all traces $T_i$ do
 4:             Let $S_i^u = \{v \in S \mid t_i(v) < t_i(u)\}$.
 5:             if $S_i^u = \emptyset$ then
 6:                 Let $\mathrm{score}_i(S,u) = 0$ if $u$ is the source of $T_i$, otherwise $\mathrm{score}_i(S,u) = -\infty$.
 7:             else
 8:                 $\mathrm{score}_i(S,u) = \log|S_i^u| - \lambda \sum_{v \in S_i^u} [t_i(u) - t_i(v)]$.
 9:         Let $\mathrm{score}(S,u) = \ell^{-1} \cdot \sum_i \mathrm{score}_i(S,u)$.
10:     Let $R(u) = \mathrm{argmax}\{\mathrm{score}(S,u)\}$.
11: for all ordered pairs of vertices $u, v$ do
12:     if $t_i(v) < t_i(u)$ in at least $\ell/3$ traces and $v \in R(u)$ then
13:         Insert edge $\{u,v\}$ into $\hat{G}$.
14: Output $\hat{G}$.
```

KL-divergence. For distributions p, q on $\mathbb{R}$ having density functions f and g, respectively, their KL-divergence is defined by

$$D(p \,\|\, q) = \int f(x) \log\left(\frac{f(x)}{g(x)}\right) dx. \quad (1)$$

One interpretation of the KL-divergence is that it is the expected difference between $\log(f(x))$ and $\log(g(x))$ when x is randomly sampled using distribution p. If one thinks of p and q as two forecasts of the distribution of x, and one samples x using p and applies the *logarithmic scoring rule*, which outputs a score equal to the log-density of the forecast distribution at the sampled point, then $D(p \,\|\, q)$ is the difference in the expected scores of the correct and the incorrect forecast. A useful lower bound on this difference is supplied by Pinsker's Inequality:

$$D(p \,\|\, q) \geq 2\, \|p - q\|_{\mathsf{TV}}^2, \quad (2)$$

where $\|\cdot\|_{\mathsf{TV}}$ denotes the total variation distance.

Quasi-timestamps and conditional distributions From now on in this section, we assume $\lambda = 1$. This assumption is without loss of generality, since the algorithm's behavior in unchanged if we modify its input by setting $\lambda = 1$ and multiplying the timestamps in all traces by λ; after modifying the input in this way, the input distribution is the same as if the traces had originally been sampled using the infection process with parameter $\lambda = 1$.

Our analysis of Algorithm 2 hinges on understanding the conditional distribution of the infection time $t(u)$, given the infection times of its neighbors. Directly analyzing this conditional distribution is surprisingly tricky, however. The reason is that u itself may infect some of its neighbors, so conditioning on the event that a neighbor of u was infected at time t_0 influences the probability density of $t(u)$ in a straightforward way at times $t > t_0$ but in a much less straightforward way at times $t < t_0$. We can avoid this "backward conditioning" by applying the following artifice.

Recall the description of the infection process in terms of Dijkstra's algorithm in Section 3: edges sample i.i.d. edge lengths and the timestamps $t(v)$ are equal to the distance labels assigned by Dijkstra's algorithm when computing single-source shortest paths from source s. Now consider the sample space defined by the tuple of independent random edge lengths $y(v, w)$. For any vertices $u \neq v$, define a random variable $\mathring{t}(v)$ to be the distance label assigned to v when we *delete* u and its incident edges from G to obtain a subgraph $G - u$, and then we run Dijkstra's algorithm on this subgraph. One can think of $\mathring{t}(v)$ as the time when v would have been infected if u did not exist. We will call $\mathring{t}(v)$ the *quasi-timestamp of* v (with respect to u). If $N(u) = \{v_1, \ldots, v_k\}$ is the set of neighbors of u, and if we sample a trace originating at a source $s \neq u$, then the executions of Dijkstra's algorithm in G and $G-u$ will coincide until the step in which u is discovered and is assigned the distance label $t(u) = \min_j\{\mathring{t}(v_j)+y(v_j, u)\}$. From this equation, it is easy to deduce a formula for the conditional distribution of $t(u)$ given the k-tuple of quasi-timestamps $\mathring{\mathbf{t}} = (\mathring{t}(v_j))_{j=1}^k$. Using the standard notation z^+ to denote $\max\{z, 0\}$ for any real number z, we have

$$\Pr(t(u) > t \mid \mathring{\mathbf{t}}) = \exp\left(-\sum_{j=1}^{k}(t - \mathring{t}(v_j))^+\right). \tag{3}$$

The conditional probability density is easy to calculate by differentiating the right side of (3) with respect to t. For any vertex set S not containing u, let $S\langle t\rangle$ denote the set of vertices $v \in S$ such that $\mathring{t}(v) < t$, and let $\rho(t, S) = |S\langle t\rangle|$. Then the conditional probability density function of $t(u)$ satisfies

$$f(t) = \rho(t, N(u)) \exp\left(-\sum_{j=1}^{k}(t - \mathring{t}(v_j))^+\right) \tag{4}$$

$$\log f(t) = \log(\rho(t, N(u))) - \sum_{v \in N(u)}(t - \mathring{t}(v))^+. \tag{5}$$

It is worth pausing here to note an important and subtle point. The information contained in a trace T is insufficient to determine the vector of quasi-timestamps $\mathring{\mathbf{t}}$, since quasi-timestamps are defined by running the infection process in the graph $G - u$, whereas the trace represents the outcome of running the same process in G. Consequently, our algorithm does not have sufficient information to evaluate $\log f(t)$ at arbitrary values of t. Luckily, the equation

$$(t(u) - t(v))^+ = (t(u) - \mathring{t}(v))^+$$

holds for all $v \neq u$, since $\mathring{t}(v)$ differs from $t(v)$ only when both quantities are greater than $t(u)$. Thus, our algorithm has sufficient information to evaluate $\log f(t(u))$, and in fact the value $\text{score}_i(S, u)$ defined in Algorithm 2, coincides with the formula for $\log f(t(u))$ on the right side of (5), when $S = N(u)$ and $\lambda = 1$.

Analysis of the reconstruction algorithm. The foregoing discussion prompts the following definitions. Fix a vector of quasi-timestamps $\mathring{\mathbf{t}} = (\mathring{t}(v))_{v \neq u}$, and for any set of vertices S not containing u, let p^S be the probability distribution on $\mathbb{R}$ with density function

$$f^S(t) = \rho(t, S) \exp\left(-\sum_{v \in S}(t - \mathring{t}(v))^+\right). \tag{6}$$

One can think of p^S as the distribution of the infection time $t(u)$ that would be predicted by a forecaster who knows the values $\mathring{t}(v)$ for $v \in S$ and who believes that S is the set of neighbors of u. Letting $N = N(u)$, each timestamp $t_i(u)$ is a random sample from the distribution p^N, and $\text{score}_i(S, u)$ is the result of applying the logarithmic scoring rule to the distribution $f^S(t)$ and the random sample $t(u)$. Therefore

$$\mathbb{E}[\text{score}_i(N, u) - \text{score}_i(S, u)] = D(p^N \parallel p^S) \tag{7}$$

$$\geq 2\|p^N - p^S\|_{\mathsf{TV}}^2. \tag{8}$$

The key to analyzing Algorithm 2 lies in proving a lower bound on the expected total variation distance between p^N and p^S. The following lemma supplies the lower bound. Its proof is omitted for space reasons.

LEMMA 5.4. *Fix a vertex u, let $N = N(u)$ be its neighbor set, and fix some $S \subseteq V \setminus \{u\}$ distinct from N. Letting $\pi(S \oplus N, u)$ denote the probability that at least one element of the set $S \oplus N$ is infected before u, we have*

$$\mathbb{E}\left(\|p^N - p^S\|_{\mathsf{TV}}\right) \geq \tfrac{1}{10}\Delta^{-2}\pi(S \oplus N, u). \tag{9}$$

Combining Pinsker's Inequality with Lemma 5.4 we immediately obtain the following corollary.

COROLLARY 5.5. *If $N = N(u)$ and S is any set such that $\pi(S \oplus N, u) > 1/4$, then for each trace T_i the expected value of $\text{score}_i(N) - \text{score}_i(S)$ is $\Omega(\Delta^{-4})$.*

Using this corollary, we are ready to prove our main theorem.

THEOREM 5.6. *For any constant $c > 0$, the probability that Algorithm 2 fails to perfectly reconstruct G, when given*

$$\ell = \Omega(\Delta^9 \log^2 \Delta \log n)$$

complete traces, is at most $1/n^c$.

PROOF. Let us say that a set S *differs significantly* from $N(u)$ if $\pi(S \oplus N(u), u) > 1/4$. Let us say that an ordered pair of vertices (u, v) violates the *empirical frequency property* if the empirical frequency of the event $t_i(v) < t_i(u)$ among the traces $T_1, \ldots, T_\ell$ differs by more than $\frac{1}{12}$ from the probability that $t(v) < t(u)$ in a random trace. Exponential tail inequalities for sums of i.i.d. random variables establish that when ℓ is as specified in the theorem statement, with probability at least $1 - 1/n^c$, there is no vertex u such that $R(u)$ differs significantly from $N(u)$ and no ordered pair (u, v) that violates the empirical frequency property. The proofs of these high-probability guarantees are omitted for space reasons.

Assuming that no set $R(u)$ differs significantly from $N(u)$ and that no pair (u, v) violates the empirical frequency property, we now prove that the algorithm's output, $\hat{G}$, is equal to G. If $\{u, v\}$ is an edge of G, assume without loss of generality that the event $t(v) < t(u)$ has probability at least 1/2. By the empirical frequency property, at least $\ell/3$ traces satisfy $t_i(v) < t_i(u)$. Furthermore, v must belong to $R(u)$, since if it belonged to $R(u) \oplus N(u)$ it would imply that $\pi(R(u) \oplus N(u), u) \geq \Pr(t(v) < t(u)) \geq 1/2$, violating our assumption that $R(u)$ doesn't differ significantly from $N(u)$. Therefore $v \in R(u)$ and the algorithm adds $\{u, v\}$ to $\hat{G}$. Now suppose $\{u, v\}$ is an edge of $\hat{G}$, and assume without loss of generality that this edge was inserted when processing the ordered pair (u, v). Thus, at least $\ell/3$ traces satisfy $t_i(v) < t_i(u)$, and $v \in R(u)$. By the empirical frequency property, we know that a random trace satisfies $t(v) < t(u)$ with probability at least $1/4$. As before, if v belonged to $R(u) \oplus N(u)$ this would violate our assumption that $R(u)$ does not differ significantly from $N(u)$. Hence $v \in N(u)$, which means that $\{u, v\}$ is an edge of G as well. $\square$

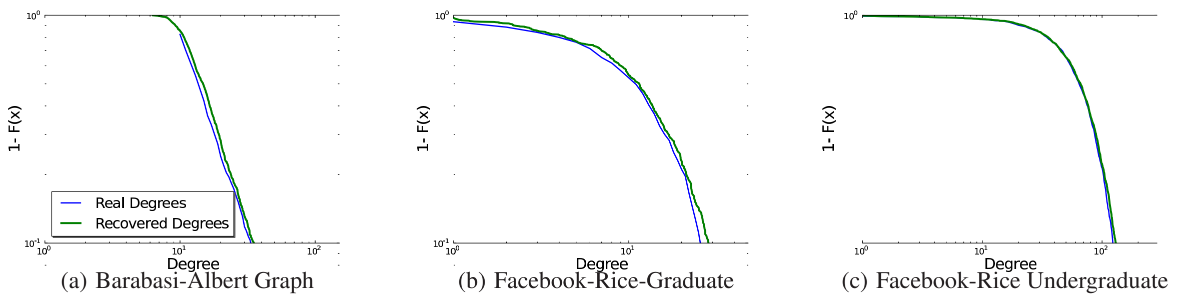

Figure 1: Complementary cumulative density function (CCDF) of degree reconstruction using $\Omega(n)$ traces for (a) a synthetic network with 1,024 nodes generated using the Barabasi-Albert algorithm, and two real social networks: two subsets of the Facebook network comprising 503 graduate students (a) and 1220 undergraduate students (c), respectively, from Rice University.

6. EXPERIMENTAL ANALYSIS

In the preceding sections we have established trace complexity results for various network inference tasks. In this section, our goal is to assess our predictions on real and synthetic social and information networks whose type, number of nodes, and maximum degree (Δ) we now describe.

We use two real social networks, namely two Facebook subnetworks comprising 503 ($\Delta = 48$) graduate and 1220 ($\Delta = 287$) undergraduate students, respectively [18]. We also generate three synthetic networks, each possessing 1024 vertices, whose generative models frequently arise in practice in the analysis of networks. We generated a *Barabasi-Albert Network* [4] ($\Delta = 174$), which is a preferential attachment model, a $G_{(n,p)}$ *Network* [9] ($\Delta = 253$) with $p = 0.2$, and a *Power-Law Tree*, whose node degree distribution follows a power-law distribution with exponent 3 ($\Delta = 94$).

First, we evaluate the performance of the algorithm to reconstruct the degree distribution of networks without inferring the network itself (Section 4.3). Figure 1 shows the reconstruction of the degree distribution using $\Omega(n)$ traces of the Barabasi-Albert Network and the two Facebook subnetworks. We used $10n$ traces, and the plots show that the CCDF curves for the real degrees and for the reconstructed distribution have almost perfect overlap.

Turning our attention back to network inference, the $\Omega(n\Delta^{1-\epsilon})$ lower-bound established in Section 3 tells us that the First-Edge algorithm is nearly optimal for perfect network inference in the general case. Thus, we assess the performance of our algorithms against this limit. The performance of First-Edge is notoriously predictable: if we use ℓ traces where ℓ is less than the total number of edges in the network, then it returns nearly ℓ edges which are all true positives, and it never returns false positives.

If we allow false positives, we can use heuristics to improve the First-Edge's recall. To this end, we propose the following heuristic that uses the degree distribution reconstruction algorithm (Section 4.3) in a pre-processing phase, and places an edge in the inferred network provided the edge has probability at least p of being in the graph. We call this heuristic *First-Edge+*.

In First-Edge+, we use the memoryless property of the exponential distribution to establish the probability p of an edge pertaining to a network G. The algorithm works as follows. Consider a node u that appears as the root of a trace at time $t_0 = 0$. When u spreads the epidemic, some node v is going to be the next infected at time t_1, which was sampled from an exponential distribution with parameter λ. At time t_1, notice that there are exactly $d_u - 1$ nodes waiting to be infected by u, and exactly $d_v - 1$ waiting to be infected by v, where d_u and d_v are the degrees of u and v respectively. At time t_1 any of these nodes is equally likely to be infected, due to the memoryless property. Moreover, the next node w that appears in a trace after time t_1 is going to be infected by u with probability $p_{(u,w)} = \frac{d_u-1}{du+dv-2}$ and by v with probability $p_{(v,w)} = \frac{d_v-1}{du+dv-2}$. We can approximate[6]this reasoning for larger prefixes of the trace: given a sequence $u_1, \cdots, u_k$ of infected nodes starting at the source of the epidemic, the probability that u_{k+1} is a neighbor of u_i is roughly $p_{(u_i,u_{k+1})} \simeq \frac{d_{u_i}}{\sum_j d_{u_j}}$. Therefore, for every segment of a trace that starts at the source, we infer an edge (u, v) if $p_{(u,v)} > p$, computed using the reconstructed degrees, where p is a tunable parameter. In our experiments we arbitrarily chose $p = 0.5$.

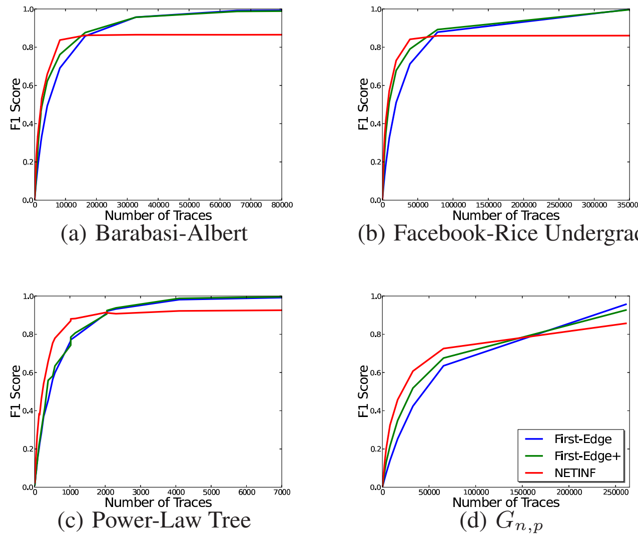

Figure 2: F1 score of the First-Edge, First-Edge+, and NetInf algorithms applied to different real and synthetic networks against a varying number of traces. (best viewed in color)

Note that First-Edge+ may not terminate as soon as we have inferred enough edges, even in the event that all true positives have been found, an effect that degrades its precision performance. To prevent this, we keep a variable T, which can be thought of as the *temperature* of the inference process. Let M be a counter of the edges inferred at any given time during the inference process, and

[6]The exact probability depends on the number of edges between each of the nodes $u_1, \ldots, u_k$ and the rest of the graph.

$\hat{E}$ be an estimate of the total number of edges, computed using the degree reconstruction algorithm in the pre-processing phase. We define $T = \frac{M}{\hat{E}}$ and run the algorithm as long as $T < 1.0$. In addition, whenever we infer a new edge, we flip a coin and remove, with probability T, a previously inferred edge with the lowest estimated probability of existence. Thus, while the network is "cold", i.e., many undiscovered edges, edges are rapidly added and a few are removed, which boosts the recall. When the network is "warm", i.e., the number of inferred edges approaches $|E|$, we carefully select edges by exchanging previously inferred ones with better choices, thereby contributing to the precision.

Figure 2 contrasts the performance of First-Edge, First-Edge+ and an existing network algorithm, NETINF [13], with respect to the F1 measure. NETINF requires the number of edges in the network as input, and thus we give it an advantage, by setting the number of edges to the true cardinality of edges for each network.

In Figures 2(a) and 2(b), we observe that, as First-Edge+ and NETINF are less conservative, their F1 performances have an advantage over First-Edge for small numbers of traces, with First-Edge+ approaching the performance to NETINF. Interestingly, in Figure 2(c), we see that First-Edge and First-Edge+ achieve perfect tree inference with roughly $5,000$ traces, which reflects a trace complexity in $\Omega(n)$ rather than in $\Omega(\log n)$, which is the trace complexity of Algorithm 1.[7] This result illustrates the relevance of the algorithms for special cases we developed in Section 5. Last, in proving lower-bounds for trace complexity, we frequently use random graphs as the worst-case examples. This is shown in Figure 2(d), where neither our algorithms nor NETINF can achieve high inference performance, even for large numbers of traces.

In accordance with our discussion in Section 4.1, we confirm that, in practice, we need significantly fewer than $n*\Delta$ traces for inferring most of the edges. It is perhaps surprising that First-Edge+, which is extremely simple, achieves comparable performance to the more elaborate counterpart, NETINF. In addition, while NETINF reaches a plateau that limits its performance, First-Edge+ approaches perfect inference as the number of traces goes to $\Omega(n\Delta)$. In the cases in which NETINF achieves higher performance than First-Edge+, the latter is never much worse than the former. This presents a practitioner with a trade-off between the two algorithms. For large networks, while First-Edge+ is extremely easy to implement and makes network inferences (in a preemptive fashion) in a matter of seconds, NETINF takes a couple of hours to run to completion and requires the implementation of an elaborate algorithm.

7. CONCLUSION

Our goal is to provide the building blocks for a rigorous foundation to the rapidly-expanding network inference topic. Previous works have validated claims through experiments on relatively small graphs as compared to the large number of traces utilized, whereas the relation that binds these two quantities remains insufficiently understood. Accordingly, we believe that a solid foundation for the network inference problem remains a fundamental open question, and that works like [20], as well as ours, provide the initial contributions toward that goal.

Our results have direct applicability in the design of network inference algorithms. More specifically, we rigorously study how much useful information can be extracted from a trace for network inference, or more generally, the inference of network properties without reconstructing the network, such as the node degree distribution. We first show that, to perfectly reconstruct general graphs, nothing better than looking at the first pair of infected nodes in a trace can really be done. We additionally show that the remainder of a trace contains rich information that can reduce the trace complexity of the task for special case graphs. Finally, we build on the previous results to develop extremely simple and efficient reconstruction algorithms that exhibit competitive inference performance with the more elaborate and computationally costly ones.

[7]In our experiments Algorithm 1 consistently returned the true edge set without false positives with $O(\log n)$ traces for various networks of various sizes. Therefore, in the interest of space we omit the data from these experiments.

8. REFERENCES

[1] E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Proc. of the 2005 IEEE/WIC/ACM Int'l Conf. on Web Intelligence*, 2005.

[2] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, London, 1975.

[3] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proc. of the 4th ACM Int'l Conf. on Web search and Data Mining*, 2011.

[4] A.-L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, Oct. 1999.

[5] S. G. Bobkov and M. Ledoux. On modified logarithmic sobolev inequalities for bernoulli and poisson measures. *Journal of Functional Analysis*, 156(2):347 – 365, 1998.

[6] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[7] N. DU, L. Song, A. Smola, and M. Yuan. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems 25*, pages 2789–2797. 2012.

[8] R. Durrett. *Probability: Theory and examples*. Cambridge Series in Statistical and Probabilistic Mathematics, 2011.

[9] P. Erdös and A. Rényi. On the evolution of random graphs. In *Pub. of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[10] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM Comput. Commun. Rev.*, 29(4):251–262, Aug. 1999.

[11] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *J. Amer. Stat. Assoc.*, 102:359–378, 2007.

[12] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proc. of the 28th Int'l Conf. on Machine Learning*, 2011.

[13] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2010.

[14] V. Gripon and M. Rabbat. Reconstructing a graph from path traces. *CoRR*, abs/1301.6916, 2013.

[15] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proc. of the 13th Int'l Conf. on World Wide Web*, 2004.

[16] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2003.

[17] I. Kontoyiannis and M. Madiman. Measure concentration for compound poisson distributions. *Electron. Commun. Probab.*, 11:no. 5, 45–57, 2006.

[18] A. Mislove, B. Viswanath, K. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *Proc. 3rd ACM Int'l. Conf. on Web Search and Data Mining*, 2010.

[19] S. Myers and J. Leskovec. On the convexity of latent social network inference. In *Advances in Neural Information Processing Systems 23*, pages 1741–1749. 2010.

[20] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. In *SIGMETRICS*, pages 211–222, 2012.

[21] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167–256, 2003.

[22] E. M. Rogers and E. Rogers. *Diffusion of Innovations*. Free Press, 5th edition, Aug. 2003.