Knowledge and Information Systems manuscript No.
KAIS-13-4187R1

Efficiently Spotting the Starting Points of an Epidemic in a Large Graph

B. Aditya Prakash · Jilles Vreeken · Christos Faloutsos

Received: Feb 16, 2013 / Revised: May 04, 2013 / Accepted: May 25, 2013

Abstract Given a snapshot of a large graph, in which an infection has been spreading for some time, can we identify those nodes from which the infection started to spread? In other words, can we reliably tell who the culprits are? In this paper we answer this question affirmatively, and give an efficient method called NetSleuth for the well-known Susceptible-Infected virus propagation model.

Essentially, we are after that set of seed nodes that best explain the given snapshot. We propose to employ the Minimum Description Length (MDL) principle to identify the best set of seed nodes and virus propagation ripple, as the one by which we can most succinctly describe the infected graph.

We give an highly efficient algorithm to identify likely sets of seed nodes given a snapshot. Then, given these seed nodes, we show we can optimize the virus propagation ripple in a principled way by maximizing likelihood. With all three combined, NetSleuth can automatically identify the correct number of seed nodes, as well as which nodes are the culprits.

Experimentation on our method shows high accuracy in the detection of seed nodes, in addition to the correct automatic identification of their number. Moreover, NetSleuth scales linearly in the number of nodes of the graph.

B. Aditya Prakash (✉)
Computer Science Department, Virginia Tech.
2202 Kraft Drive, Blacksburg VA 24060 USA
E-mail: badityap@cs.vt.edu

Jilles Vreeken
Advanced Database Research and Modelling, University of Antwerp
E-mail: jilles.vreeken@ua.ac.be

Christos Faloutsos
Computer Science Department, Carnegie Mellon University
E-mail: christos@cs.cmu.edu

1 Introduction

When considering large graphs, epidemics are everywhere. For social networks, infectious diseases like the flu are prime examples, but hypes/memes are similarly epidemic in nature; whether it is friends discussing that latest gadget or phone, or sharing a funny video, there are nodes 'infecting' each other. Similarly, a computer virus can cause an epidemic in a computer network, as can a contaminant in a water distribution network. In each of these cases, given a single snapshot of a partially infected network, an important and challenging research question is how we can reliably identify those nodes from which the epidemic started; whether for inoculation to prevent future epidemics, or for exploitation for viral marketing.

As such, given a snapshot of a large graph $G(\mathcal{V}, \mathcal{E})$ in which a subset of nodes $\mathcal{V}' \subseteq \mathcal{V}$ is currently infected, and assuming the Susceptible-Infected (SI) propagation model, we consider the problem of how to efficiently and reliably find those seed nodes $\mathcal{S} \subseteq \mathcal{V}'$ from which the epidemic started, without requiring the number of seed nodes in advance. In other words, we address the questions: *How many culprits are there, and who are they?*

We propose to employ the Minimum Description Length (MDL) principle [29; 15] to identify that set of seed nodes and virus propagation ripple that most succinctly describes the given snapshot. We give an highly efficient

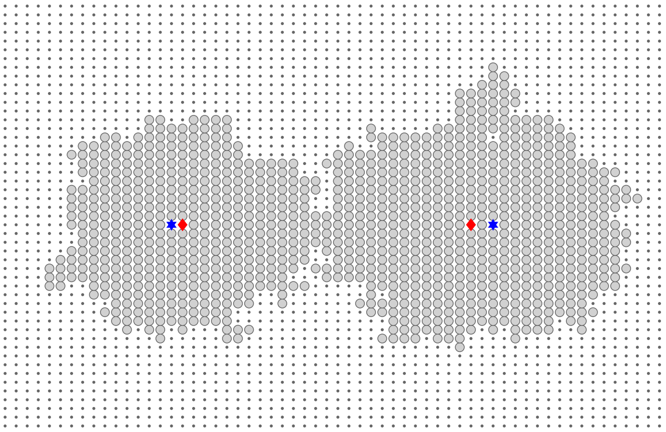


Fig. 1. Example: Culprits, *how many*, and *which ones*? A snapshot of a 2D grid network in which an infection has been stochastically spreading. Grey circles indicate infected nodes, while Grey dots are not infected. The 2 Blue stars denote seeds from which the data was generated. The 2 Red diamonds denote the seeds automatically discovered by NetSleuth—that is, both in *number* (two) and *location*. As the picture shows, the discovered seeds are spatially very close to the true seeds. Moreover, for this random snapshot, the discovered seeds obtain *better* likelihood and MDL scores than the true seeds.

Table 1. Comparison between three culprit-identifying methods: NetSleuth, Rumor-centrality [35], and Effectors [18]

	infection model	$k>1$	automatically determines k	$O(\cdot)^\dagger$
NetSleuth (our method)	SI	✓	✓	Linear
Rumor-centrality [35]	SI	–	–	Quadratic
Effectors [18]	IC	✓	–	Quadratic

† Running time given for arbitrary graphs.

algorithm to identify likely seed nodes, and show we can easily optimize the description length of the virus propagation ripple for a given seed set by greedily maximizing likelihood. As such, we can identify the best set of seed nodes in a principled manner, without having to require the user to choose k, the number of seed nodes, in advance.

As an example, consider Figure 1. It depicts an example grid-structured graph, in which a subgraph has been infected by a stochastic process starting from two seed nodes. The plot shows the true seed nodes, as well as the seed nodes automatically identified by NetSleuth; it finds the correct number of seed nodes, and places these where a human would; in fact, the discovered seeds have a higher likelihood for generating this infected subgraph than the true seed nodes.

We develop a two step approach by first finding high-quality seeds given the number of seeds, and then using our carefully designed MDL score to pinpoint the true number of seeds. For the first part, we use the notion of 'exoneration' from the un-infected frontier—e.g., in Figure 1 the nodes on the edge of the infected snapshot are unlikely to be the culprits due to the large number of un-infected nodes surrounding them. Based on this idea, we develop a novel 'submatrix-laplacian' method to find out the best seed sets given a number of seeds (see Section 5 for more details). Given these seed-sets, we also give an efficient algorithm to compute the MDL scores, thus finding the number of seeds in a parameter-free way.

Although network infection models have been researched extensively, identifying the seed nodes of an epidemic is surprisingly under studied. We are, however, not the first to research this problem. Recently, Shah and Zaman [34, 35] developed rumor-centrality for identifying the *single* source node of an epidemic. In contrast, we allow for *multiple* seed nodes, and automatically determine their number. Lappas et al [18] studied the 'Effectors' problem of identifying k seed nodes in a steady-state network snapshot, under the Independent Cascade (IC) model. In contrast, we study the Susceptible–Infected (SI) model, can consider snapshots from any time point during the epidemic, and our approach is parameter-free as by MDL we can automatically identify the best value for k. Furthermore, and very importantly for large graphs, in comparison our method is computationally very efficient. Table 1 gives a comparison of NetSleuth to these methods. We discuss related work in more detail in Section 2.

Experimentation shows that NetSleuth detects seed nodes and automatically identifies their number, both with high-accuracy. With synthetic data we show it can handle difficult fringe cases, and is in agreement with human intuition. We show we reliably identify the correct number of seed nodes on real data, and also that our detected seeds are of very high quality (measured by multiple metrics). Finally, we show our method scales linearly with the number of edges of the graph.

The remainder of the article is organized straightforwardly. We discuss related work next in Section 2. Then in Section 3, we will discuss notation and give quick introductions to the SI model and the Minimum Description Length principle. Next, in Section 4, we formalize the problem of identifying culprits using MDL. We develop our proposed method for mining good seed sets, NetSleuth, in Section 5. We experimentally evaluate NetSleuth in Section 6. We round up with discussion and conclusions resp. in Section 7 and 8. For readability, we postpone the proofs to the Appendix.

2 Related Work

As mentioned in the introduction, although diffusion processes have been widely studied, the problem of 'reverse engineering' the epidemic has not received much attention, except papers by Shah and Zaman [34, 35] and Lappas et al [18].

Shah and Zaman [34, 35] formalized the notion of *rumor-centrality* for identifying the single source node of an epidemic under the SI model, and showed an optimal algorithm for *d-regular trees.*

Lappas et al [18] study the problem of identifying k seed nodes, or *effectors* of a partially activated network, which is assumed to be in *steady-state* under the IC (Independent-Cascade) model. In contrast, we allow for (a) multiple seed nodes, (b) a snapshot from any time during the infection, and (c) find the number of seeds automatically, even for general graphs. Finally we are also more efficient with linear time on edges of the infected graph. Also we are, to the best of our knowledge, the first to employ MDL with the goal of identifying culprits.

Work related to identifying sources of infection can be categorized into three main parts: epidemic thresholds, information diffusion and ecology. Most of these works either consider only single virus models or typically use only simulation or analyze on very restricted underlying networks. There is a lot of research interest in studying different types of information dissemination processes on large graphs in general, including (a) information cascades [2; 12; 42], (b) blog propagation [21; 14], and (c) viral marketing and product penetration [19].

Epidemic Thresholds The canonical text-book for epidemiological models like SI is Anderson and May [1]. Much research in virus propagation studied the so-called epidemic threshold, that is, to determine the condition under which an epidemic will not break out [17; 24; 5; 11]. Prakash et al [26, 27] in addition discuss that the leading eigenvalue and a model-dependent constant are the only parameters that determine the epidemic threshold for almost all virus propagation models.

Influence Maximization An important problem under the viral marketing setting is the influence maximization problem [28; 16; 13; 6; 32]. Another remotely related work is outbreak detection [20] in the sense that we aim to select a subset of '*important*' nodes on graphs.

Immunization Another related problem for such propagation processes is immunization - the problem of finding the best nodes for removal to stop an epidemic, with effective immunization strategies for static and dynamic graphs [38; 3; 25].

Minimum Description Length We are not the first to use the Minimum Description Length principle [29; 15] for a data mining purpose. Faloutsos and Megalooikonomou [10] argue many data mining problems are related to Kolmogorov Complexity, which means they can be practically solved through compression. Examples of MDL based solutions include clustering [7], pattern set mining [40], outlier detection [36], and community detection [4]. We are, to the best of our knowledge, however, the first to employ MDL with the goal of identifying culprits.

3 Preliminaries

In this section we introduce notation we will use throughout the paper, we discuss preliminaries regarding the Minimum Description Length (MDL) principle as well as for the Susceptible-Infected (SI) model—the virus infection spreading model we use.

3.1 Notation

In Table 2 we give an overview of the most important notation and symbols we will use in the paper. We consider undirected, unweighted graphs $G = (\mathcal{V}, \mathcal{E})$ of $N = |\mathcal{V}|$ nodes. The degree of a node i is denoted by $d(i)$. We indicate by $G_I = (\mathcal{V}_I, \mathcal{E}_I)$ the infected subgraph of G. By $\mathcal{E}_F \subseteq \mathcal{E}$ we denote the all edges in G connecting with nodes in $\mathcal{V}_I$.

All logarithms are to base 2, and we adopt the standard convention that $0 \log 0 = 0$. We denote the transpose of any matrix or vector V as V^T. Finally note that L_A is a *submatrix* of $L(G)$, *not* the laplacian matrix of G_I.

Table 2. Terms and Symbols

Symbol	**Definition and Description**
SI model	Susceptible-Infected model
β	attack probability of the virus in the SI model
$G = (\mathcal{V}, \mathcal{E})$	graph under consideration
$G_I = (\mathcal{V}_I, \mathcal{E}_I)$	given infected subgraph of G
R	ripple, an ordered list of, per iteration, a set of nodes how the virus propagates
N	$\lvert\mathcal{V}\rvert$, number of nodes in graph G
N_I	$\lvert\mathcal{V}_I\rvert$, number of nodes in graph G_I
$d(i)$	degree of node i
$\mathcal{F}$	set of un-infected nodes having at least one infected neighbor (in $\mathcal{V}_I$)
$\mathcal{F}_i^t$	set of un-infected nodes having i infected neighbors (in $\mathcal{V}_I$) at time step t
$\mathcal{E}_F$	set of edges connecting nodes in $\mathcal{F}$ to $\mathcal{V}_I$
$A(G)$	adjacency matrix of graph G (size $N \times N$)
A	adjacency matrix of G_I (size $N_I \times N_I$)
$D(G)$	diagonal degree matrix of graph G
$L(G)$	laplacian matrix of G i.e. $L(G) = D(G) - A(G)$
L_A	*submatrix* (size $N_I \times N_I$) of $L(G)$ corresponding to the infected graph G_I
Q_{MDL}	MDL-based culprits quality measure (see Section 6)
Q_{JD}	set-Jaccard-distance-based culprits quality measure (see § 6)

3.2 The Susceptible-Infected Model

One of the most widely studied epidemic models is the so-called 'Susceptible-Infected' (SI) model [1]. In this model, each object/node in the underlying graph is in one of two states: Susceptible (S) or Infected (I). Once infected, a node stays infected forever. At every time-step, each of the infected nodes tries to infect each of its uninfected neighbors independently with probability β, the model parameter that reflects the strength of the virus.

It is important to note that $1/\beta$ defines a natural time-scale, as intuitively it is the expected number of time-steps for a successful attack over an edge. As an example, if we assume that the underlying network is a clique of N nodes, under continuous time, the model can be written as:

$$\frac{dI(t)}{dt} = \beta(N - I(t))I(t) \quad , \tag{1}$$

where $I(t)$ is the number of infected nodes at time t—the solution is the logistic function and it is invariant to $\beta \times t$. Finally note that given *any* starting point, the whole network will eventually get infected. Hence, closer is the infection to completion (infecting everyone), harder it is to detect the culprits. Below, we will consider a discrete time set-up in which the duration of one time-slice is defined by $1/\beta$. That is, for low β we consider a lower time resolution than for high β. This corresponds with intuition, as by the higher infection probability, the latter case will be much more eventful if we would consider a fixed time resolution.

3.3 Minimum Description Length Principle

The Minimum Description Length principle (MDL) [29; 15], like its close cousin Minimum Message Length (MML) [41], is a practical version of Kolmogorov Complexity [22]. All three embrace the slogan *Induction by Compression*. For MDL, this can be roughly described as follows.

Given a set of models $\mathcal{M}$, the best model $M \in \mathcal{M}$ is the one that minimizes

$$\mathcal{L}(M) + \mathcal{L}(\mathcal{D} \mid M) \quad ,$$

in which

$\mathcal{L}(M)$ is the length in bits of the description of M, and
$\mathcal{L}(\mathcal{D} \mid M)$ is the length of the description of the data encoded with M.

This is called two-part MDL, or *crude* MDL—as opposed to *refined* MDL, where model and data are encoded together [15]. We use two-part MDL because we are specifically interested in the model: the seed nodes and ripple that give the best description. Further, although refined MDL has stronger theoretical foundations, it cannot be computed except for some special cases.

To allow for fair comparison between different $M \in \mathcal{M}$, MDL requires us to define a lossless encoding. However, as our goal is to measure complexity of a description, we are only concerned with code *lengths*, not actual code words—compression is not our goal, but a means to identify good models.

In order to use MDL, we have to define what our models $\mathcal{M}$ are, how a $M \in \mathcal{M}$ describes the data at hand, and how we encode this all in bits.

4 Our Problem Formulation

Loosely speaking, given a snapshot of an infected graph, our goal is to obtain the most succinct explanation of 'what happened'. To do so, we require two ingredients: the first is a formal objective—a cost function—which we discuss in this section. The second is then an algorithm to find good solutions, which we give in Section 5. In this Section we will formalize the problem in terms of the MDL principle.

Our cost function will consist of two parts, 1) scoring the seed set (Model cost) and 2) scoring the successive infected nodes starting from the seed (Data cost).

We assume that both sender and receiver already know the layout of $G = (\mathcal{V}, \mathcal{E})$, yet the receiver does not know yet which nodes are in $G_I = (\mathcal{V}_I, \mathcal{E}_I)$. That is, in the general terms of MDL we gave in Section 3.3, G_I the data $\mathcal{D}$ we want to describe using a set of seed nodes $M \in \mathcal{M}$. As such, informally, our goal is to identify those nodes, and an infection propagation ripple starting from those nodes, by which G_I can most easily be described.

4.1 Cost of the Model

As our models we consider *seed sets*. A seed set $\mathcal{S} \subseteq \mathcal{V}_I$ is a subset of $|\mathcal{S}|$ nodes of G_I from which the infection starts spreading—the 'patients zero', so to speak. We denote by $\mathcal{L}(\mathcal{S})$ the encoded length, in bits, of a seed set $\mathcal{S}$.

To describe a seed set $\mathcal{S}$, we first have to encode how many nodes $\mathcal{S}$ contains. This number, $|\mathcal{S}|$, is upper-bounded by the number of nodes in G. Hence, we can encode $|\mathcal{S}|$ in as many as $\log N$ bits, by which we spend equally many bits to encode either a small or a large number—essentially a uniform prior. In general, however, we favor small seeds sets: simple explanations. Hence, the MDL optimal Universal code for integers [30] is a better choice, as it already rewards smaller seed sets by requiring fewer bits to encode their size. With this encoding, $\mathcal{L}_{\mathbb{N}}$, the number of bits to encode an integer $n \geq 1$ is defined as

$$\mathcal{L}_{\mathbb{N}}(n) = \log^*(n) + \log(c_0) \quad ,$$

where $\log^*$ is defined as $\log^*(n) = \log(n) + \log\log(n) + \cdots$, where only the positive terms are included. By choosing c_0 as

$$c_0 = \sum_{J \geq 1} 2^{-\mathcal{L}_{\mathbb{N}}(j)} \approx 2.865064 \quad ,$$

we ensure the Kraft inequality is satisfied, and hence that $\mathcal{L}_{\mathbb{N}}$ a valid encoding. That is, all probabilities sum to ≤ 1.0.

To identify which nodes in G are seed nodes, we use the very efficient class of data-to-model codes [39]. A data-to-model code is essentially an index into a canonically ordered enumeration of all possible data (values) given the model (the provided information). Here, we know $|\mathcal{S}|$ unique nodes have to be selected out of N, for which there are $\binom{N}{|\mathcal{S}|}$ possibilities. Assuming a canonical order, $\log \binom{N}{|\mathcal{S}|}$ gives us the length in bits of an index to the correct set of node ids.

Combining the above, we now have $\mathcal{L}(\mathcal{S})$ for the number of bits to identify a seed set $\mathcal{S} \subseteq \mathcal{V}_I$ as

$$\mathcal{L}(\mathcal{S}) = \mathcal{L}_{\mathbb{N}}(|\mathcal{S}|) + \log \binom{N}{|\mathcal{S}|} \tag{2}$$

4.2 Cost of the Data given the Model

Next, we need to describe the infected subgraph G_I given a seed set $\mathcal{S}$. We do this by encoding the infection *propagation ripple*, or the description of 'what happened'. Starting from the seed nodes, per time step we identify that set of nodes that gets infected at this time step, iterating until we have identified all the infected nodes.[1]

[1] When not interested in the actual ripple R, one could encode G_I by its overall probability starting from $\mathcal{S}$. Obtaining this probability, however, is very expensive, even by MCMC sampling. As we will see in Sections 5 and 6 computing a good ripple is both cheap and gives good results.

4.2.1 Propagation ripples

More formally, a propagation ripple R is a list of node ids per time-step t, which represents the order in which nodes of G_I became infected, starting from $\mathcal{S}$ at time $t = 0$. Let us write $\mathcal{V}_I^t(\mathcal{S}, R)$ to indicate the set of infected nodes at time t starting from seed set $\mathcal{S}$ and following ripple R, with $\mathcal{V}_I^0 = \mathcal{S}$. For readability, we do not write $\mathcal{S}$ and R wherever clear from context. As such, a valid propagation ripple R is a partitioning of node ids $\mathcal{V}_I \setminus \mathcal{S}$ of G_I, where every node in a part is required to have an edge from a node $j \in \mathcal{V}_I^{t-1}$.

Clearly, however, not every ripple from the seed set to the final infected subgraph is equally simple to describe. For instance, the more infected neighbors an uninfected node has, the more it is under constant attack, and hence the more likely it is that it will get infected. As such, it should be more succinct to describe that a node under more heavy attack gets infected, than it would cost to describe the infection of a node only under single attack.

4.2.2 Frontier sets

To encode a ripple R, at each time t we consider the collection of nodes currently under attack given the SI model (i.e. non-infected nodes with currently at least one infected neighbor, or if $t = 0$, neighbors to a seed-node $\in \mathcal{S}$). We refer to this set as $\mathcal{F}^t$, for the frontier-set at time t. Define *attack degree* $a(n)$ of a non-infected node n as the number of infected neighbor nodes it has at the current iteration, i.e. $a(n) = |\{j \in \mathcal{V} \mid e_{jn} \in \mathcal{E} \wedge X_j(t)\}|$, in which $X_j(t)$ is an indicator function for whether node j is infected at time t.

We divide $\mathcal{F}^t$ into disjoint subsets $\mathcal{F}_i^t$ per attack degree i, that is, into sets of nodes having the same attack degree. As such, we have $\mathcal{F}^t = \mathcal{F}_1^t \cup \mathcal{F}_2^t \cup \ldots$, and correspondingly $f_1^t, f_2^t, \ldots$ for the sizes of these subsets (we will drop using the t superscript, when clear from context).

Starting from the seed set, for every time step t the receiver can easily construct the corresponding frontier set $\mathcal{F}^t$—which leaves us to transmit which of the nodes, if any, in the frontier set got infected in the current iteration. As, however, the infection probabilities per attack degree differ, we transmit this information per $\mathcal{F}_d^t$.

4.2.3 Probability of Infection

The SI model assumes an attack probability parameter β—so, the independent probability p_d of a node in $\mathcal{F}_d$ being infected is: $p_d = 1 - (1 - \beta)^d$. Given p_d we can write down the probability distribution of a total of m_d nodes being infected for each subset $\mathcal{F}_d$. This is simply a Binomial with parameter p_d i.e.

$$p(m_d \mid f_d, d) = \binom{f_d}{k} p_d^{m_d} (1 - p_d)^{f_d - m_d}$$

Hence, as such, a value for β determines p.

4.2.4 Encoding a Wave of Attack

Given p, a probability distribution for seeing m_d nodes out of f_d infected given an attack degree d, we need $-\log p(m_d \mid f_d, d)$ bits to optimally transmit the value of m_d. That is, we encode m_d using an optimal prefix code—for which we can calculate the optimal code lengths by Shannon entropy [8]. Then, once we know both f_d and m_d, we can use code words of resp. $-\log \frac{m_d}{f_d}$ and $-\log 1 - \frac{m_d}{f_d}$ bits long to transmit whether a node in $\mathcal{F}_d$ got infected or not. This gives us

$$\mathcal{L}(\mathcal{F}^t) = -\sum_{\mathcal{F}_d^t \in \mathcal{F}^t} \left(\log\left(p\left(m_d \mid f_d, d\right)\right) + m_d \log\left(\frac{m_d}{f_d}\right) + (f_d - m_d)\log\left(1 - \frac{m_d}{f_d}\right)\right) \tag{3}$$

for encoding the infectees in the frontier set at time t.

Then, for the recipient to know when to stop reading, we have to transmit how many time steps until we have reached G_I. The number of iterations T will be transmitted just like $|\mathcal{S}|$, using $\mathcal{L}_{\mathbb{N}}$. For the ripple R, starting from the frontier-set defined by the seed nodes $\mathcal{S}$, we iteratively transmit which nodes got infected at $t + 1$—which in turn allows the recipient to construct $\mathcal{F}^{t+1}$. Note that, by $\mathcal{L}(\mathcal{F}^t)$ we assume ripple R to be in time scale of $1/\beta$. That is, for low β we consider a lower time resolution than for high β. This is because the SI model displays a natural invariance of time-scale (see Section 3.2). So we have ripple R that gives the infections at every $1/\beta$ time-steps.

With the above, we have $\mathcal{L}(R \mid \mathcal{S})$ for the encoded length of a ripple R starting at a seed set $\mathcal{S}$ as

$$\mathcal{L}(R \mid \mathcal{S}) = \mathcal{L}_{\mathbb{N}}(T) + \sum_{t}^{T} \mathcal{L}(\mathcal{F}^t)$$

4.3 The Problem

With $\mathcal{L}(\mathcal{S})$ and $\mathcal{L}(R \mid \mathcal{S})$, we have as the total description length $\mathcal{L}(G_I, \mathcal{S}, R)$ of an infected subgraph G_I of G following a valid infection propagation ripple R starting from a set of seed nodes $\mathcal{S}$ by

$$\mathcal{L}(G_I, \mathcal{S}, R) = \mathcal{L}(\mathcal{S}) + \mathcal{L}(R \mid \mathcal{S})$$

Note that as G is constant over all seed sets $\mathcal{S}$ and ripples R, we can safely ignore it in the computation of the total encoded size, for its encoded length would be constant term and hence not influence the selection of the best model.

By which we can now formally state our problem.

Minimal Infection Description Problem *Given a snapshot of a graph $G(\mathcal{V}, \mathcal{E})$ of N nodes, of which the subgraph $G_I(\mathcal{V}_I, \mathcal{E}_I)$ of N_I nodes are infected,* *and an infection probability β, by the Minimum Description Length principle we are after that seed set $\mathcal{S}$ and that valid propagation ripple R for which*

$$\mathcal{L}(G_I, \mathcal{S}, R)$$

is minimal for the Susceptible-Infected propagation strategy.

Clearly, this problem entails a large search space.We can break the problem into two sub-problems. First of all, the identification of a set of seed nodes. It is easy to see that the set of possible seed sets consists of all possible subsets of $\mathcal{V}_I$, i.e. there are $2^{|\mathcal{V}_I|}$ such sets. The second problem is to find the optimal propagation path given a seed set. This requires minimizing $\mathcal{L}(R \mid \mathcal{S})$, which comes down to finding the most likely virus propagation path given $\mathcal{S}$.

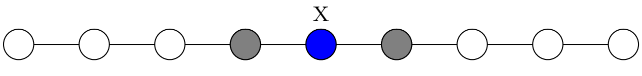
In fact, Shah and Zaman [35] show that for a given infected snapshot in an arbitrary graph the problem of just finding a *single* Maximum Likelihood Estimate (MLE) seed is already quite hard, #P-Complete, equivalent to counting the number of linear extensions of a poset. Further, the provable algorithm they give works for one seed on d-regular trees only. Generalizing this approach towards a fast solution for multiple seeds on general graphs is non-trivial. We hence resort to heuristics.

5 Proposed Method

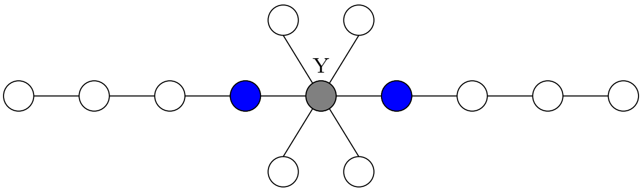
The outline of our approach is as follows: given a fixed number of seeds k, we identify a high-quality k-seed set. Given these seed nodes, we optimize the propagation ripple. With these two combined, we can use our MDL score to identify the best k.

5.1 Best seed-set given number of seeds—‘Exoneration’

A central idea to our approach is that intuitively, un-infected nodes provide some degree of ‘exoneration’ from ‘blame’ of a neighboring infected node being a seed node. See Figure 2—it shows two illustrative examples of an infected chain (a) and a chain with a star in the middle (b) (colored nodes are infected and blue denote the true seeds). Note that while the node X is the most central among the infected nodes and is rightly the most likely seed, the node Y is not a likely seed because of the many un-infected nodes surrounding it. In fact, in this case the most likely starting points would be the two Blue nodes. Hence any method to identify the seed-sets should take into account the centrality of the infected nodes among the infected graph, but also penalize nodes for being too close to the un-infected frontier (the ‘exoneration’). As we explain next, our method is able to do this in a principled manner.



(a) a chain



(b) a chain-star

Fig. 2. Centrality is not enough — the effect of 'exoneration': Example infection snapshots. Colored nodes are infected, blue nodes are the true seeds. (a) Node X is the most central among the infected nodes; (b) Node Y is the most central among infected nodes, but the high count of non-infected neighbors 'exonerates' it.

5.2 Finding best single seed—Our Main Idea

We first explain how to find the best single seed and then how to extend it to multiple seeds. Jumping ahead, the main idea is as follows.

Main Idea The single best seed s^* is the one with the highest score in $\vec{u}_1$ i.e.

$$s^* = \arg\max_s \vec{u}_1(s)$$

where $\vec{u}_1$ is the *smallest* eigenvector of the laplacian submatrix L_A as defined in Table 2. Next, we give the justification.

5.3 Finding the best single seed—Justification

From Section 4, it is clear that nodes that are not in either the final frontier set $\mathcal{F}$ or $\mathcal{V}_I$ play no role, as they were not infectious nor could have been infected. Hence, WLOG, assume G contains only the infected subgraph G_I and the frontier set $\mathcal{F}$. Also, assume nodes are numbered in such a way that the first $|\mathcal{V} - \mathcal{V}_I|$ nodes are the un-infected nodes and the rest are the infected ones. If the total number of nodes in the graph is N, the number of infected nodes is N_I, then the number of un-infected nodes in G is $N - N_I$. Further notation is given in Table 2.

Let $X_i(t)$ be the indicator (0/1) Random Variable denoting if node i in the graph is infected or not at time t (1 = infected, 0 = un-infected). Let $Y_{ij}(t)$ be the indicator random variable denoting if node j successfully attacks i at time t. Consider the following update equation for any node $i \in \mathcal{V}_I$:

$$\begin{aligned} X_i(t+1) = X_i(t) + & \\ & (1 - X_i(t)) \times \\ & \bigvee_{j \in \mathcal{N}(i)} Y_{ij}(t)(X_j(t) - X_i(t) + X_i(t)) \end{aligned} \tag{4}$$

Following the above equation, if $X_i(t) = 1$ then $X_i(t+1) = 1$, i.e., once a node is infected, it stays infected. Also if $X_i(t) = 0$, then $X_i(t+1) = \bigvee_{j \in \mathcal{N}(i)} Y_{ij}(t) X_j(t)$. Or in other words, an uninfected node may get infected only if an infected neighbor successfully transmits the infection. Additionally for any node $i \in \mathcal{V} - \mathcal{V}_I$, we define $X_i(t) = 0$, as these nodes were not infected at all during the infection process. Hence, the above equations exactly define a discrete-time SI process but with the constraint that the nodes in the given final frontier set *always stay un-infected*, thus enforcing the 'exoneration' discussed before. Hence we want to find the seed node which maximizes spread in this 'constrained' epidemic, which we show how to next.

For any node $i \in \mathcal{V}_I$, taking expectations both sides of Equation 4, and using that for any indicator random variable X, $\mathbb{E}[X] = \Pr(X = 1)$, we get:

$$P_i(t+1) = P_i(t) + U - V$$

where,

$$\begin{aligned} P_i(t) &= \Pr(X_i(t) = 1) \\ U &= \mathbb{E}\left[\bigvee_{j \in \mathcal{N}(i)} Y_{ij}(t)(X_j(t) - X_i(t) + X_i(t))\right] \\ V &= \mathbb{E}\left[X_i(t) \times \bigvee_{j \in \mathcal{N}(i)} Y_{ij}(t)(X_j(t) - X_i(t) + X_i(t))\right] \end{aligned}$$

Clearly, as all the terms inside are positive,

$$V \geq 0, U \geq 0 \tag{5}$$

Also,

$$\begin{aligned} U &\leq \sum_{j \in \mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t) + P_i(t)) \\ &= \sum_{j \in \mathcal{N}(i)} A(G)_{ij} P_i(t) + \sum_{j \in \mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t)) \end{aligned}$$

as an infected node j attacks any of its neighbors i independently with probability $A(G)_{ij}$ (i.e. $\mathbb{E}[Y_{ij}(t)] = A(G)_{ij}$) and because by linearity of expectation, for any two events indicator random variables $\mathbb{1}_A$ and $\mathbb{1}_B$, we have $\mathbb{1}_A \vee \mathbb{1}_B = \mathbb{1}_A + \mathbb{1}_B - \mathbb{1}_A \mathbb{1}_B \Rightarrow \mathbb{E}[\mathbb{1}_A \vee \mathbb{1}_B] \leq \mathbb{E}[\mathbb{1}_A] + \mathbb{E}[\mathbb{1}_B]$. Also note that:

$$\sum_{j \in \mathcal{N}(i)} A(G)_{ij} P_i(t) \leq d_{max} \times P_i(t)$$

where d_{max} is the largest degree in graph G. Thus,

$$U \leq d_{max}P_i(t) + \sum_{j \in \mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t)) \tag{6}$$

From Equations 5 and 6, we can conclude that, for each node $i \in \mathcal{V}_I$:

$$\begin{aligned} P_i(t+1) \leq P_i(t) &+ d_{max}P_i(t) \\ &+ \sum_{j \in \mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t)) \end{aligned}$$

Let $\sigma = 1 + d_{max}$. Recall that $\forall t,\ P_i(t) = 0$ for any *eventual* un-infected node $i \in \mathcal{V} - \mathcal{V}_I$. Let $\vec{P}(t) = [P_1(t), P_2(t), \ldots, P_N(t)]^T$ (over all the nodes in $\mathcal{V}$). Then we can write:

$$\vec{P}(t+1) \leq \sigma(I - \frac{1}{\sigma}M)\vec{P}(t) \tag{7}$$

where, the matrix M (size $N \times N$) is:

$$M = \begin{vmatrix} 0_{N-N_I,N-N_I} & 0_{N-N_I,N_I} \\ 0_{N_I,N-N_I} & L_A \end{vmatrix}$$

where we write $0_{N,M}$ for an all-zeros matrix of size $N \times M$. Let the subvector of $\vec{P}(t+1)$ corresponding to the infected nodes be written as $\vec{P}_I(t+1)$. Then continuing from above and using the upper bound as an approximation, we get:

$$\begin{aligned} \vec{P}_I(t+1) &\approx \sigma(I - \frac{1}{\sigma}L_A)\vec{P}_I(t) \\ &= \sigma(I - \frac{1}{\sigma}L_A)^t\vec{P}_I(0) \\ &= \sigma\sum_i \lambda_i^t \vec{u}_i \vec{u}_i^T \vec{P}_I(0) \end{aligned} \tag{8}$$

where, λ_i and $\vec{u}_i$ are the eigenvalues and eigenvectors of the matrix $I - \frac{1}{\sigma}L_A$. We have the following two lemmas:

Lemma 1 *The largest eigenvalue λ_1 and eigenvector $\vec{u}_1$ of the matrix $I - \frac{1}{\sigma}L_A$ are all positive and real.*

Proof (Details omitted for brevity) The matrix $I - \frac{1}{\sigma}L_A$ is non-negative, and imagining $I - \frac{1}{\sigma}L_A$ as an adjacency matrix, the corresponding graph is irreducible, as graph G_I (adjacency matrix A) is connected. We then get the lemma due to the Perron-Frobenius theorem [23]. For the formal proof, see the Appendix. □

Lemma 2 *The largest eigenvalue of matrix $I - \frac{1}{\sigma}L_A$ and the smallest eigenvalue of L_A are related as $\lambda_1(I - \frac{1}{\sigma}L_A) = 1 - \frac{1}{\sigma}\lambda_N(L_A)$.*

Proof (Details omitted for brevity) It is easy to see that any eigenvalue $eig(I - \frac{1}{\sigma}L_A) = 1 - eig(\frac{1}{\sigma}L_A)$. As the matrices are symmetric, all the eigenvalues involved are real. By the Cauchy eigenvalue interlacing theorem [37] applied to $L(G)$, all the eigenvalues of any co-factor C_{LG} of $L(G)$ are positive. By the famous Kirchhoff's matrix theorem [9], the determinant of any co-factor C_{LG} is also non-zero as it counts the number of spanning trees of G. Also, it is well-known that the determinant of any matrix is just the product of its eigenvalues [37]. Hence, all eigenvalues of any co-factor matrix C_{LG} of $L(G)$ are strictly positive. We can similarly apply eigenvalue interlacing successively to a suitable C_{LG} and so on till we get to L_A (a principal submatrix of $L(G)$), and get that all eigenvalues of L_A are strictly positive. The lemma follows then. For the formal proof, see the Appendix. □

Hence, the eigenvector $\vec{u}_1$ is also the eigenvector corresponding to the smallest eigenvalue of L_A.

Now, from Equation 8 and Lemma 1, we have:

$$\begin{aligned}\vec{P}_I(t+1) &= \sigma\lambda_1^t \sum_i \frac{\lambda_i^t}{\lambda_1^t} \vec{u}_i \vec{u}_i^T \vec{P}_I(0) \\ &\approx \sigma\lambda_1^t \vec{u}_1 \vec{u}_1^T \vec{P}_I(0)\end{aligned}$$

assuming a substantial eigen-gap or 'big-enough' t. Now assuming that $\vec{P}_I(0)$ is all zero *except* for a single seed s for which it is 1, we can conclude that ultimately in our 'constrained' epidemic,

$$\forall i \in \mathcal{V}_I,\ Pr(X_i = 1|s) \underset{\sim}{\propto} \vec{u}_1(i)\vec{u}_1(s) \tag{9}$$

$$\forall i \in \mathcal{V} - \mathcal{V}_I,\ Pr(X_i = 1|s) = 0 \tag{10}$$

Clearly the most likely single seed s^* would be:

$$s^* = \arg\max_s \left[\sum_{i \in \mathcal{V}_I} Pr(X_i = 1|s) + \sum_{i \in \mathcal{V} - \mathcal{V}_I} (1 - Pr(X_i = 1|s)) \right]$$

Using Equations 9 and 10,

$$\begin{aligned} s^* &\approx \arg\max_s \vec{u}_1(s) \sum_{i \in \mathcal{V}_I} \vec{u}_1(i) \\ &= \arg\max_s \vec{u}_1(s) \end{aligned} \tag{11}$$

Hence, for a single seed, we just need to find the node with the largest score in $\vec{u}_1$ (which is also the smallest eigenvector of the laplacian submatrix L_A from Lemma 2).

5.4 Finding best k-seed set

Note that simply taking the top-k in the above eigenvector will not give good k-seed-sets due to lack of diversity. This is because the error in the upper-bound approximation used in Equation 8 will become larger due to increase in the norm of $\vec{P}_I(0)$. Hence, we treat the newly chosen seed, say s^*, as *un-infected*, effectively exonerating its neighbors and boosting diversity. We redo our computation on the resulting *smaller* infected graph, but a potentially larger frontier set—hence, we take the next best seed *given* the s^* that has already been chosen. So for any given k, we successively find the best next seed, given the previous choices, by removing the previously chosen seeds from the infected set and solving Equation 11. For example, in Figure 1, the top suspect (Red on the right) will have a lot of suspicious neighbors as well. Thus, using our exoneration technique, the algorithm will be forced away from them towards the remaining Red seed.

5.5 Finding a good ripple

As discussed before, once we find the best seed-set $\mathcal{S}_k$ for a given k, we optimize the propagation ripple of $\mathcal{S}_k$ to G_I to minimize the total encoded size. Recall from Section 4 that this involves minimizing $\mathcal{L}(R \mid \mathcal{S})$, which consists of two terms. First, we have the cost of encoding the length of the ripple, the number of time-steps. While $\mathcal{L}_{\mathbb{N}}$ does grow for higher values of T, in practice this term will be dwarfed by the actual encoding of the subsequent frontier sets. As such, minimizing $\mathcal{L}(R \mid \mathcal{S})$ essentially comes down to minimizing $-\sum_t^T \mathcal{L}(\mathcal{F}^t)$, or, in other words, maximizing the likelihood of the ripple R. Further recall that the SI model has a natural scaling invariance, $1/\beta$. As our score takes this into account, the ripple with the smallest description length should too.

Hence, we design the following procedure. For each attack-degree set $\mathcal{F}_d$, at any iteration we scale the number of attacks by $1/\beta$ i.e. a set of size f_d is equivalent to a set of size f_d/β. Then, to get the overall Maximum Likelihood Estimate (MLE) ripple, we adopt the following greedy heuristic. We assume that the overall MLE ripple always performs a locally optimal next step. Hence this boils down to choosing the most-likely nodes to get infected at any given step, for a given frontier set $\mathcal{F}$.

It is well-known that a Binomial distribution $B(n, p)$ has its mode at $\lfloor (n+1)p \rfloor$. Using this fact, at any iteration t, taking into account the scaling, we can see that the most likely number of nodes infected in an attack-degree set $\mathcal{F}_d$ would be $n_d = \lfloor (f_d/\beta + 1) \times p_d \rfloor$—where p_d as defined before in Section 4 is the attack probability in the set $\mathcal{F}_d$. As such, we can simply uniformly choose this number of nodes from the $\mathcal{F}_d$, as each node in $\mathcal{F}_d$ is equally likely to be infected. We do this for every non-empty attack-degree set, for every iteration, until we have infected exactly the observed snapshot. This way, we obtain an approximation of the most likely propagation ripple for a given seed-set $\mathcal{S}_k$

Algorithm 1 NetSleuth

```
Input: $G(\mathcal{V},\mathcal{E}) \equiv G_I^* \cup \mathcal{F}^*$, $G_I^*(\mathcal{V}_I,\mathcal{E}_I)$ (the infected graph) and $\mathcal{F}^*$ (the frontier set).
Output: $\mathcal{S}$ = the set of seeds (culprits).
$L(G) = D(G) - A(G)$, the Laplacian matrix corresponding to graph $G$.
$\mathcal{S} = \{\}$
$G_I = G_I^*$
while $\mathcal{L}(G_I,\mathcal{S},R)$ decreases do
    $L_A$ = the submatrix of $L(G)$ corresponding to $G_I$.
    $v$ = eigenvector of $L_A$ corresponding to the smallest eigenvalue.
    $next = \arg\max_i v(i)$
    $\mathcal{S} = \mathcal{S} \cup \{next\}$
    $R$ = ripple maximizing likelihood of $G_I$ from $\mathcal{S}$
    $G_I = G_I \backslash \{next\}$ (Graph $G_I$ with node $next$ removed)
end while
return $\mathcal{S}$
```

and can subsequently score the quality of the seed set and ripple using our MDL score.

Another possible way of getting a high quality ripple is through Markov Chain Monte Carlo simulations. This essentially entails that we perform multiple full epidemic simulations from the given seed-set to reach the infected set. Each such simulation can give us a ripple which we can score using our MDL formulation. Hence this method is very expensive (though arguably more accurate as we compare multiple ripples), and we do not use it.

5.6 The NetSleuth Algorithm

A naive approach would be to consider all possible k, and select that $\mathcal{S}_k$ that minimizes $\mathcal{L}(G_I,\mathcal{S}_k,R)$. Clearly, this raises computational issues. Assuming convexity of the score over k, we stop the search as soon as increasing k increases the MDL score. Though it is hard to prove the convexity of the MDL score, we will see in the experimental evaluation it is a valid strategy. Algorithm 1 gives the pseudo-code and Lemma 3 shows the running time for our algorithm NetSleuth.

Lemma 3 (Running Time of NetSleuth) *The time complexity of* NetSleuth *is* $O(k^*(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I))$.

Proof (Details omitted for brevity) The main thing to note is that running time depends on the true number of seeds k^*, as we repeat steps for every value of k from 1 to k^*. For any one iteration, the running time is $O(\mathcal{E}_I + T_{\text{RIPPLE}} + T_{\text{MDL}})$, where we compute the smallest eigenvector of L_A using the Lanczos method, which is approximately $O(E)$ (# edges) for sparse graphs. T_{RIPPLE} and T_{MDL} are the running time to compute the best ripple and the MDL score given seeds. These tasks are intuitively linear on the number of edges and nodes in the frontier and infected sets, as we need to traverse each of the infected and frontier sets only once each for each task. Hence we get the given overall linear complexity. For the formal proof, please see the Appendix. □

Hence NetSleuth is *linear* in the number of edges and vertices of the infected sub-graph and the frontier set, which makes our method scalable for large graphs (as compared to the methods in [35; 18] which, even for detecting a *single seed*, are $O(N^2)$).

6 Experiments

Here we experimentally evaluate NetSleuth, in particular its effectiveness in finding culprits—whether it correctly identifies (a) *how many* as well as (b) *which ones*—and (c) its scalability.

6.1 Experimental Setup

We implemented NetSleuth in Matlab, and in addition implemented the SI model as a discrete event simulation in C++. All reported results are averaged over 10 independent runs (so we generate 10 graphs for each seed set).

In our study we use both synthetic and real networks—we chose synthetic networks exemplifying different types of situations. We consider the following networks:

1. GRID is a 60×60 2D grid as shown in Figure 1.
2. CHAIN-STAR It is a graph of total 107 nodes. The first 7 nodes form a linear chain and the middle node has 100 additional neighbors.
3. AS-OREGON The Oregon AS router graph which is a network graph collected from the Oregon router views. It contains 15 420 links among 3 995 AS peers.[2]

For the experiments on AS-OREGON, we ran the experiments for true-seed count $k^* = 1, 2, 3, 5$. So for each seed-set, we run a simulation till at least 30% of the graph is infected, and give the resulting footprint as input to NetSleuth. Note that, the larger the number of infections, the tougher it is to find the true seeds, as in the SI model any seed will eventually infect the whole graph with certainty. Finally, we make sure that the infected sub-graph was connected—otherwise, we just have separate problem instances.

As discussed in the introduction and Section 2, the existing proposals for identifying culprits consider significantly different problems settings than we do (see Table 1); the Rumor Centrality of Shah and Zaman [34, 35] can only discover *one* seed node, while Effectors of Lappas et al [18] even consider a completely different infection model. As such we can not meaningfully compare performances and hence here only consider NetSleuth.

[2] For more information see `http://topology.eecs.umich.edu/data.html`.

6.2 Evaluation—a subtle issue

How to evaluate the goodness of a seed set? That is, in Figure 1, how close are the red seeds (recovered) from the blue seeds (true)? Notice that the recovered seeds may actually have *better* score than the actual ones, for the same reason that the sample mean of a group of 1D Gaussian instances gives lower sum-squared-distances than the theoretical mean of the distribution. Moreover, even for evaluation, it is intractable to compute the exact probability of observing the footprint from a given seed-set.

Thus we propose two quality measures. The first, Q_{MDL}, is based on our MDL formulation: we report the ratio of the MDL score of our seeds, vs. the MDL score of the actual seeds i.e.

$$Q_{\text{MDL}} = \frac{\mathcal{L}(G_I, \mathcal{S}, R)}{\mathcal{L}(G_I, \mathcal{S}^*, R^*)} \tag{12}$$

Clearly, the closer to 1, the better.

The second Q_{JD} intuitively measures the *overlap* of the footprint produced by a seed-set $\mathcal{S}$ and the input footprint $G_I(\mathcal{V}_I, \mathcal{E}_I)$. Clearly, the candidate seed-set $\mathcal{S}$ can produce n footprints, when we run n simulations; so we compute $\mathbb{E}[JD_{\mathcal{S}}(\mathcal{V}_I)]$, the average Jaccard distance[3] of all these n footprints, w.r.t. the true input footprint $\mathcal{V}_I$. As with Q_{MDL}, we normalize it with the corresponding score $\mathbb{E}[JD_{\mathcal{S}^*}(\mathcal{V}_I)]$ for the true seed-set, and thus report the ratio,

$$Q_{\text{JD}} = \frac{\mathbb{E}[JD_{\mathcal{S}}(\mathcal{V}_I)]}{\mathbb{E}[JD_{\mathcal{S}^*}(\mathcal{V}_I)]} \tag{13}$$

Again, the closer to 1, the better.

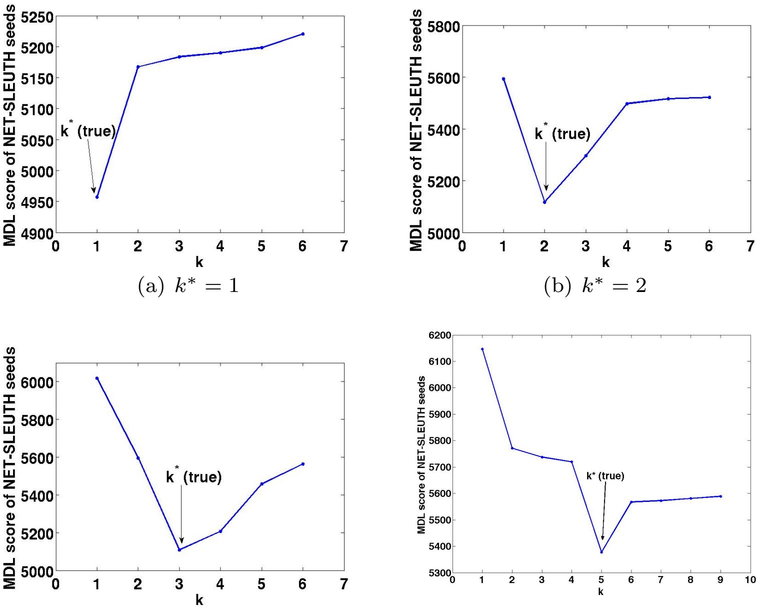
6.3 Effectiveness of NetSleuth in identifying How Many

In short, NetSleuth was able to find the exact number of seeds for all the cases. Figures 3 show the MDL score as a function of $k = 1, 2, \ldots, 9$ seeds found by NetSleuth before stopping, for true seed-sets with (a) $k^* = 1$, (b) $k^* = 2$, (c) $k^* = 3$ and (d) $k^* = 5$ respectively on the AS-OREGON network. Note that the plots show near-convexity, with the minimum at the true k^*, justifying our choice of stopping after $j = 6$ iterations of increasing scores. It also shows the power of our approach, as we can easily recover the true number of seeds using a principled approach.

6.4 Effectiveness of NetSleuth in identifying Which Ones

In short, in addition to finding the correct number of seeds, NetSleuth is able to identify good-quality seeds with high accuracy. For our synthetic graphs,

[3] We use the standard definition of Jaccard Distance between two sets $\mathcal{A}$ and $\mathcal{B} = 1 - \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$.



(a) $k^* = 1$

(b) $k^* = 2$

(c) $k^* = 3$

(d) $k^* = 5$

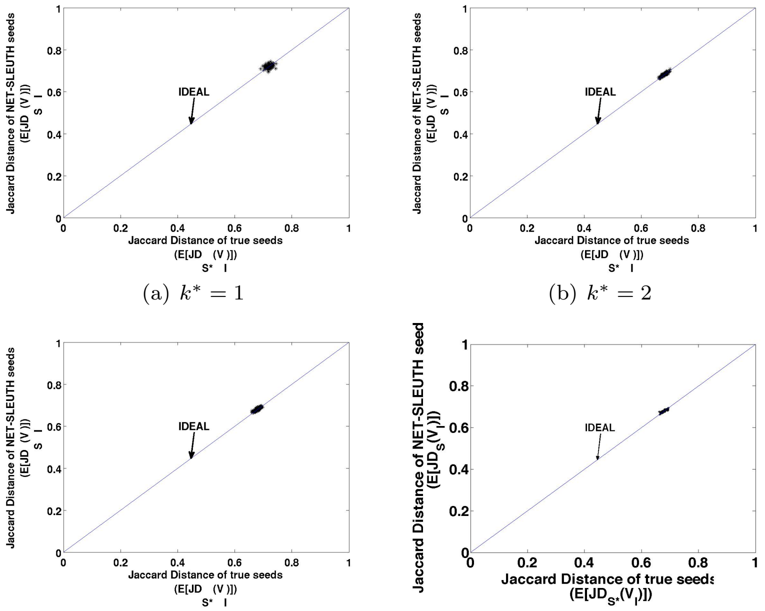
Fig. 3. MDL finds the correct k: MDL scores of seeds by NetSleuth when (a) $k^* = 1$, (b) $k^* = 2$, (c) $k^* = 3$ and (d) $k^* = 5$ on the AS-OREGON network. Each point average of 10 runs. Note the near-convexity of the score, with the minimum at exactly k^*, enabling us to stop correctly.

NetSleuth is able to point out that there must have been exactly 2 seeds for both the GRID and CHAIN-STAR examples—respectively identified as the Red circles in Figure 1, and the Blue nodes in Figure 2(b)), agreeing with the ground-truth and intuition.

Figures 4 show the results of our experiments for different number $k^* = 1, 2, 3, 5$ of true seeds on the AS-OREGON graph. We randomly selected 90 seed-sets of each size. We made sure that the seed-sets contained both well-connected and weakly connected nodes. Each point is an average of 10 runs.

Firstly, although not shown in the figures, NetSleuth was able to perfectly recover the true number of seeds in almost all cases. For each seed-set, we calculate $JD_{\mathcal{S}}(\mathcal{V}_I)$ for the seeds returned by NetSleuth and the true seeds and give the scatter plot in Figures 4 for true-seed count $k^* = 1, 2, 3, 5$ respectively. Hence points on or below the 45-degree line (solid blue) are better. Clearly, almost all points are concentrated near the diagonal, showing high quality. In fact, many points are exactly on the line, meaning we are able to recover the true seeds *perfectly* for many cases. We do not show similar plots with our MDL score due to lack of space.

Next, we calculate Q_{MDL} and Q_{JD} averaged over all the different seed-sets (of the same size for $k^* = 1, 2, 3, 5$). Results are shown in the bar plots of



(a) $k^* = 1$

(b) $k^* = 2$

(c) $k^* = 3$

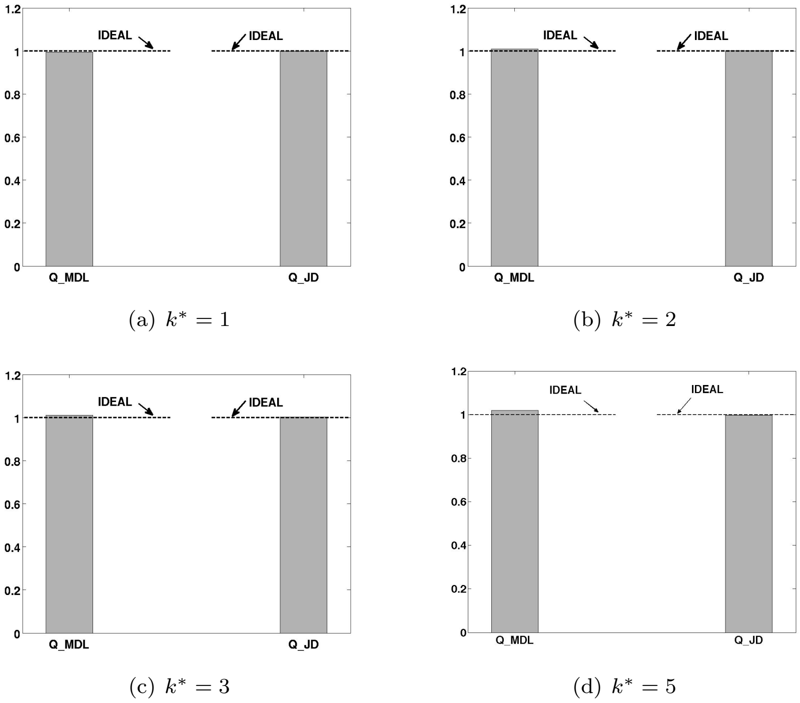
(d) $k^* = 5$

Fig. 4. Scatter Plot of the Jaccard scores of seeds returned by NetSleuth (y-axis) and the corresponding true seeds (x-axis) for (a) $k^* = 1$, (b) $k^* = 2$, (c) $k^* = 3$ and (d) $k^* = 5$. On or below the 45-degree line is better. Each point is an average over 10 runs. For all cases, the correct number of seeds was *automatically* determined. Note that for many runs the seeds identified by NetSleuth score exactly or even better than the true seeds.

Figure 5. The true-seed scores are represented by the dotted line at 1, for both Q_{MDL} and Q_{JD}. Clearly, all of bars are close to 1, demonstrating that NetSleuth consistently finds very good culprits. Moreover, both Q_{MDL} and Q_{JD} quality metrics are similar in magnitude for all k^*'s — increasing our confidence in our results.

6.5 Scalability

Figure 6 demonstrates the scalability of NetSleuth after running it on increasingly larger infected snapshots of AS-OREGON (as the complexity just depends on the size of the snapshot), for (a) $k^* = 1$ and (b) $k^* = 2$ respectively. As expected from our Lemma 3, the running-time is linear on the number of edges of the infected graph, for *both* values of k^*.



(a) $k^* = 1$

(b) $k^* = 2$

(c) $k^* = 3$

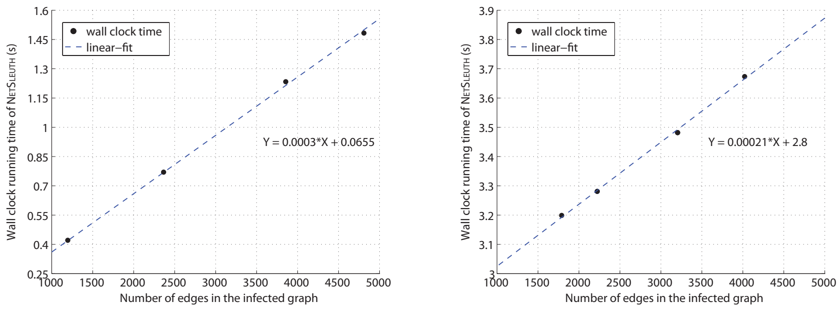
(d) $k^* = 5$

Fig. 5. Average Q_{MDL} and Q_{JD} scores for the seeds returned by NetSleuth for (a) $k^* = 1$, (b) $k^* = 2$, (c) $k^* = 3$ and (d) $k^* = 5$. Each bar represents the average over 90 different seed-sets. Note that all the bars are close to 1, indicating that we consistently find high-quality seed sets both with the Jaccard measure, and with the MDL measure.

7 Discussion

The experiments clearly show NetSleuth reliably correctly detects the correct number of seeds of an epidemic, as well as identifies their location in the graph with high precision. Moreover, with synthetic data, such as depicted in Fig. 1 and Fig. 2, we show its estimates agree with human intuition—while experiments on real data show we can handle realistic network distributions. As real graphs are very large, scalability is of utmost importance, and we validated that NetSleuth scales linearly in the size of the infected footprint of the graph. As such, NetSleuth constitutes an answer to the question: who are the culprits?

We employ the MDL principle to identify the best seed set. As we have seen in the experiments, the encoding we propose works very well in practice. It is important to note, however, that MDL is not a magic wand. Any encoding involves choices, and so does ours. One important choice we here make is to



(a) $k^* = 1$ (b) $k^* = 2$

Fig. 6. NetSleuth Scalability: Wall-clock running time (in seconds) for increasingly larger infected snapshots of AS-OREGON (as the complexity just depends on the size of the snapshot) for (a) $k^* = 1$ and (b) $k^* = 2$. Each point average of 10 runs. Note that, as expected, the running time scales linearly, for *both* values of k^*.

describe data using a *single* ripple. While this offers very high efficiency, and good results, by considering only a single path from the seeds to the footprint, we may end up in a local minimum.

The obvious alternative is to calculate the probability of reaching the footprint over *all* possible paths. For general graphs, calculating this probability is not trivial, however. Though arbitrarily good estimates can be obtained using MCMC sampling, this has very high computational cost and would harm scalability of the method. From a robustness point of view, however, it is interesting to investigate whether it is possible to define efficient encodings that consider more than a single path. One appealing approach from modern MDL theory would be to employ sequential Normalized Maximum Likelihood [31] to this end. Further, alternate model selection techniques, such as BIC [33] could be applicable.

NetSleuth is the first approach to automatically identify both the number and the location of the seed nodes. As lunch is never free, the current proposal has a number of limitations that make for engaging future research.

First of all, NetSleuth is defined for unweighted and undirected graphs. Both our current MDL score and ripple MLE are applicable to directed graphs without adjustment; for weighted graphs, however, calculating the infection probabilities becomes more involved. An overarching challenge is to extend our submatrix-Laplacian based seed set search to directed and weighted graphs.

Based on the notion of exoneration, NetSleuth relies on un-infected nodes in the graph to be true negatives; if they are (unknowingly) infected, but reported as un-infected, search for good seeds may erroneously be pushed away from the true seed. Extending our method to allow for noise in the input graph G_I will allow for more robust identification of seed nodes.

Here, we consider the SI model, which is both often used and relatively simple epidemic model. There exist many more realistic models, such as the Susceptible-Infected-Recovered (SIR) model, where infected nodes can recover and stop infecting neighbors. While it would add to the applicability of our approach, it is not immediately apparent how to generalize our encodings and submatrix-Laplacian method to models such as SIR and SIRS.

The SI model has one important parameter, β, which identifies the potency of the virus, as well as its natural time scale. In this paper we assume β to be known, as it is a parameter normally estimated by epidemiologists. As long as all considered values are assumed equi-probable, our current MDL score, can, however, be employed to identify that value of β that minimizes the description length. The key issue is efficient search, as simply running NetSleuth for all possible values of β raises computational issues.

8 Conclusions

In this paper we discussed finding culprits, the challenging problem of identifying the nodes from which an infection in a graph started to spread. We proposed to employ the Minimum Description Length principle for identifying that set of seed nodes from which the given snapshot can be described most succinctly. We introduced NetSleuth (based on a novel 'submatrix-laplacian' method), a highly efficient algorithm for both identifying the set of seed nodes that best describes the given situation, and *automatically* selecting the best number of seed nodes—in contrast to the state of the art.

Experiments showed NetSleuth attains high accuracy in detecting the seed nodes, as well as correctly identifying their number. Importantly, NetSleuth scales *linearly* with the number of edges of the infected graph, $O(\mathcal{E}_F + \mathcal{E}_I + \mathcal{V}_I)$, making it applicable on large graphs.

Future work includes extending NetSleuth to directed and weighted graphs, which entails formalizing an appropriate MDL score, and finding good and efficient path optimization strategies—while for the former this should be fairly straightforward, the latter is not quite trivial. Another promising line of research is to extend NetSleuth to the popular SIR epidemic model.

APPENDIX

Proofs

We formally prove various Lemmas used in the paper in this section.

Proof (of Lemma 1) The graph G_I is connected (as we assume the set of infected nodes are connected, otherwise we are just dealing with separate problems, one for each connected component). Now the laplacian matrix $L(G)$ has all entries except the diagonal elements as non-positive and all the diagonal elements as positive. In addition $L(G)$ is a symmetric matrix (as $A(G)$ is symmetric). The matrix L_A is a principal submatrix (i.e. it has been formed by removing matching rows and columns) of size $N_I \times N_I$ of $L(G)$. As a result, L_A is symmetric.

Consider matrix $M = (I - \frac{L_A}{\sigma})$. Clearly it is symmetric (due to the above). Consider some diagonal element M_{ii} (for some index i):

$$\begin{aligned} M_{ii} &= 1 - \frac{d_i}{1+\sigma} \\ &= 1 - \frac{d_i}{1+d_{\max}} \qquad (14) \\ &> 0 \qquad (15) \end{aligned}$$

Any off-diagonal element M_{ij} is:

$$M_{ij} = \begin{cases} 0, & \text{if } \{i,j\} \notin E_I \\ \frac{1}{1+d_{\max}}, & \text{if } \{i,j\} \in E_I \end{cases}$$

Hence, firstly M is a non-negative matrix. Further the structure of the matrix $M = (I - \frac{L_A}{\sigma})$ represents the adjacency matrix of a weighted connected graph G_M (with self loops). This is because its off-diagonal elements are non-zero only when the corresponding edge is present in G_I. Hence as G_I is connected, so is G_M. Now as G_M is connected, we have that M is irreducible.

Finally applying the well-known Perron-Frobenius theorem [23] on the non-negative irreducible matrix $M = (I - \frac{L_A}{\sigma})$, we get that the first (largest) eigenvalue λ_1 and the corresponding eigenvector $\vec{u}_1$ are all positive and real.

□

Proof (of Lemma 2) Firstly note that both matrices $M = (I - \frac{L_A}{\sigma})$ and L_A are symmetric (see proof of Lemma 1 above). Hence it follows that all their eigenvalues are real [37].

Now consider matrix the laplacian matrix $L(G)$ of graph G. It is well-known that its smallest eigenvalue is 0 [9]. Hence all its eigenvalues are non-negative. Let its eigenvalues be

$$0 \leq \mu_2 \leq \mu_3 \leq \ldots \leq \mu_N$$

Consider any co-factor matrix C_{LG} of $L(G)$. Recall that a co-factor matrix is a principal sub-matrix resulting after the removal of one matching row and column. Clearly C_{LG} is also symmetric and has all real eigenvalues. Let them be:

$$\nu_1 \leq \nu_2 \leq \nu_3 \ldots \leq \nu_{N-1}$$

We can now apply the Cauchy eigenvalue interlacing theorem [37] to $L(G)$ and C_{LG}. Applying it we get that:

$$0 \leq \nu_1 \leq \mu_2 \leq \nu_2 \ldots \leq \nu_{N-1} \leq \mu_N$$

Hence all the eigenvalues of C_{LG} are non-negative.

Now, recall that according to the famous Kirchoff matrix tree theorem [9], the determinant of *any* co-factor of the laplacian matrix $L(G)$ of graph G is equal to the number of spanning trees of G. As the number can not be zero, the determinant of C_{LG} is also non-zero i.e.

$$\det(C_{LG}) > 0$$

Further it is well known that the determinant of any matrix is equal to the product of its eigenvalues [37]. So:

$$\det(C_{LG}) = \Pi_{i=1}^{N-1} \nu_i > 0 \Rightarrow \forall_i \;\; \nu_i > 0 \tag{16}$$

i.e. *none* of the eigenvalues of C_{LG} are zero.

Recall that L_A is a principal sub-matrix of $L(G)$—hence it is a co-factor of *some* other larger principal submatrix, which is a co-factor of some other still larger principal submatrix and so on till we reach C_{LG}. Hence there is a sequence of submatrices which can lead us from C_{LG} to L_A, in which each submatrix is a co-factor of the one before it. Hence, applying Equation 16 above succesively to this sequence, we get that all the eigenvalues of L_A are strictly positve (non-zero).

Finally, note that *any* eigenvalue of $(I - \frac{L_A}{\sigma})$, $eig((I - \frac{L_A}{\sigma})) = 1 - eig(\frac{L_A}{\sigma})$. Hence as all the eigenvalues of L_A are non-zero, we get

$$\lambda_1(1 - \frac{L_A}{\sigma}) = 1 - \frac{\lambda_{N_I}(L_A)}{\sigma}$$

where $\lambda_{N_I}(L_A)$ is the smallest eigevalue of L_A. □

Proof (of Lemma 3) We keep finding $\mathcal{S}_k$ for each seed-set size until MDL tells us to stop. Hence the running time is $O(k^*(\mathcal{E}_I + T_{\text{RIPPLE}} + T_{\text{MDL}}))$, if k^* is the optimal seed-set size and T_{MDL} is the running time of computing the MDL score given the seed set size is k^*. Here we used the fact that calculating the eigenvector using the Lanczos method is approximately $O(E)$ (# edges) for sparse graphs.

The worst-case complexity T_{MDL} of calculating $\mathcal{L}(G_I, \mathcal{S}, R)$ for a given G_I, $\mathcal{S}$, and R, is $O(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I)$. The $\mathcal{L}(\mathcal{S})$ term is $O(1)$. For the $\mathcal{L}(R \mid \mathcal{S})$ term, we need to iterate over the ripple, which is at most $\mathcal{V}_I$ steps long. We only have to update the frontier set $\mathcal{F}$ when one or more nodes got infected, for which we then have to update the attack degrees of the nodes connected to the nodes infected at time t. Hence we traverse every edge in $\mathcal{E}_I + \mathcal{E}_F$, and every node in $\mathcal{V}_I$, which gives it the complexity of $O(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I)$.

Finally, the running time T_{RIPPLE} of computation of the MLE ripple for a given $\mathcal{S}_k$ is also $O(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I)$.

So the overall complexity of NetSleuth is $O(k^*(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I))$. □

Acknowledgments

This material is based upon work supported by the Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053, and the National Science Foundation under Grant No. IIS-1017415. Jilles Vreeken is supported by a Post-Doctoral Fellowship of the Research Foundation – Flanders (FWO).

References

1. Anderson RM, May RM (1991) Infectious Diseases of Humans: Dynamics and Control. Oxford University Press
2. Bikhchandani S, Hirshleifer D, Welch I (1992) A theory of fads, fashion, custom, and cultural change in informational cascades. Polit Econ 100(5):992–1026
3. Briesemeister L, Lincoln P, Porras P (2003) Epidemic profiles and defense of scale-free networks. In: WORM 2003, Washington, DC
4. Chakrabarti D, Papadimitriou S, Modha DS, Faloutsos C (2004) Fully automatic cross-associations. In: Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Seattle, WA, pp 79–88
5. Chakrabarti D, Wang Y, Wang C, Leskovec J, Faloutsos C (2008) Epidemic thresholds in real networks. TISSEC 10(4)
6. Chen W, Wang C, Wang Y (2010) Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC
7. Cilibrasi R, Vitányi P (2005) Clustering by compression. IEEE Transactions on Information Technology 51(4):1523–1545
8. Cover TM, Thomas JA (2006) Elements of Information Theory. Wiley-Interscience New York
9. Cvetković DM, Doob M, Sachs H (1998) Spectra of Graphs: Theory and Applications, 3rd Ed.
10. Faloutsos C, Megalooikonomou V (2007) On data mining, compression and Kolmogorov complexity. In: Data Mining and Knowledge Discovery, vol 15, Springer-Verlag, pp 3–20
11. Ganesh A, Massoulié L, Towsley D (2005) The effect of network topology on the spread of epidemics. In: INFOCOM
12. Goldenberg J, Libai B, Muller E (2001) Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters
13. Goyal A, Lu W, Lakshmanan LVS (2011) Simpath: An efficient algorithm for influence maximization under the linear threshold model. In: Proceedings of the 11th IEEE International Conference on Data Mining (ICDM), Vancouver, Canada
14. Gruhl D, Guha R, Liben-Nowell D, Tomkins A (2004) Information diffusion through blogspace. In: Proceedings of the 13th International Conference on World Wide Web (WWW)
15. Grünwald P (2007) The Minimum Description Length Principle. MIT Press
16. Kempe D, Kleinberg J, Tardos E (2003) Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC
17. Kephart JO, White SR (1993) Measuring and modeling computer virus prevalence. In: SP
18. Lappas T, Terzi E, Gunopulos D, Mannila H (2010) Finding effectors in social networks. In: Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC, pp 1059–1068
19. Leskovec J, Adamic LA, Huberman BA (2006) The dynamics of viral marketing. In: EC
20. Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, Glance NS (2007) Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM International

Conference on Knowledge Discovery and Data Mining (SIGKDD), San Jose, CA, pp 420–429

21. Leskovec J, McGlohon M, Faloutsos C, Glance N, Hurst M (2007) Cascading behavior in large blog graphs: Patterns and a model. In: Proceedings of the 7th SIAM International Conference on Data Mining (SDM), Minneapolis, MN
22. Li M, Vitányi P (1993) An Introduction to Kolmogorov Complexity and its Applications. Springer
23. McCuler CR (2000) The many proofs and applications of Perron's theorem. SIAM Review 42
24. Pastor-Santorras R, Vespignani A (2001) Epidemic spreading in scale-free networks. Phys Rev Let 86 14
25. Prakash BA, Tong H, Valler N, Faloutsos M, Faloutsos C (2010) Virus propagation on time-varying networks: Theory and immunization algorithms. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Barcelona, Spain
26. Prakash BA, Chakrabarti D, Faloutsos M, Valler N, Faloutsos C (2011) Threshold conditions for arbitrary cascade models on arbitrary networks. In: Proceedings of the 11th IEEE International Conference on Data Mining (ICDM), Vancouver, Canada
27. Prakash BA, Chakrabarti D, Valler N, Faloutsos M, Faloutsos C (2012) Threshold conditions for arbitrary cascade models on arbitrary networks. Knowledge and Information Systems 33(3):549–575
28. Richardson M, Domingos P (2002) Mining knowledge-sharing sites for viral marketing. In: Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Edmonton, Alberta
29. Rissanen J (1978) Modeling by shortest data description. Automatica 14(1):465–471
30. Rissanen J (1983) Modeling by shortest data description. The Annals of Statistics 11(2):416–431
31. Roos T, Rissanen J (2008) On sequentially normalized maximum likelihood models. In: Proc. Workshop on Information Theoretic Methods in Science and Engineering (WITMSE)
32. Saito K, Kimura M, Ohara K, Motoda H (2012) Efficient discovery of influential nodes for sis models in social networks. Knowledge and Information Systems 30(3):613–635
33. Schwarz G (1978) Estimating the dimension of a model. The Annals of Statistics 6(2):461–464
34. Shah D, Zaman T (2010) Detecting sources of computer viruses in networks: theory and experiment. In: SIGMETRICS, pp 203–214
35. Shah D, Zaman T (2011) Rumors in a network: Who's the culprit? IEEE Transactions on Information Technology 57(8):5163–5181
36. Smets K, Vreeken J (2011) The odd one out: Identifying and characterising anomalies. In: Proceedings of the 11th SIAM International Conference on Data Mining (SDM), Mesa, AZ, Society for Industrial and Applied Mathematics (SIAM), pp 804–815
37. Strang G (1988) Linear Algebra and its Applications, 3rd edn. Harcourt Brace Jonanovich, San Diego
38. Tong H, Prakash BA, Tsourakakis CE, Eliassi-Rad T, Faloutsos C, Chau DH (2010) On the vulnerability of large graphs. In: Proceedings of the 10th IEEE International Conference on Data Mining (ICDM), Sydney, Australia
39. Vereshchagin N, Vitanyi P (2004) Kolmogorov's structure functions and model selection. IEEE Transactions on Information Technology 50(12):3265– 3290
40. Vreeken J, van Leeuwen M, Siebes A (2011) Krimp: Mining itemsets that compress. Data Mining and Knowledge Discovery 23(1):169–214
41. Wallace C (2005) Statistical and inductive inference by minimum message length. Springer-Verlag
42. Zhao J, Wu J, Feng X, Xiong H, Xu K (2011) Information propagation in online social networks: a tie-strength perspective. Knowledge and Information Systems pp 1–20, URL `http://dx.doi.org/10.1007/s10115-011-0445-x`

Author Biographies

B. Aditya Prakash is an Assistant Professor in the Computer Science Department at Virginia Tech. He graduated with a Ph.D. from the Computer Science Department at Carnegie Mellon University in 2012, and got his B.Tech (in CS) from the Indian Institute of Technology (IIT) – Bombay in 2007. He has published more than 25 refereed papers in major venues, holds two U.S. patents and has given two tutorials (VLDB 2012 and ECML/PKDD 2012). His work has received one best paper award and two best-of-conference selections (CIKM 2012, ICDM 2012, ICDM 2011). His interests include Data Mining, Applied Machine Learning and Databases, with emphasis on large real-world networks and time-series.

Jilles Vreeken is a Post-Doctoral Researcher in the Advanced Database Research and Modelling (ADReM) group of the University of Antwerp (Belgium). His general research interests include data mining and machine learning, exploratory data analysis and pattern mining. He is particularly interested in developing well-founded theory and efficient methods for extracting informative models and patterns from very large data, as well as putting these to good use. He has published over 35 conference and journal papers, and won two best student research paper awards. In 2009, he defended his Ph.D. thesis 'Making Pattern Mining Useful' at the Universiteit Utrecht (Netherlands) under supervision of prof. Arno Siebes, for which he was awarded the 2010 ACM SIGKDD Best Dissertation Runner-Up award.

Christos Faloutsos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), the Research Contributions Award in ICDM 2006, the SIGKDD Innovations Award (2010), seventeen best paper awards, (including two 'test of time') and four teaching awards. He has served as a member of the executive committee of SIGKDD; he is an ACM Fellow; he has published over 200 refereed articles, 11 book chapters and one monograph. He holds five patents and he has given over 30 tutorials and over 10 invited distinguished lectures. His research interests include data mining for graphs and streams, fractals, database performance, and indexing for multimedia and bio-informatics data.