Towards Large-Scale Social Networks with Online Diffusion Provenance Detection

Haishuai Wang[a], Jia Wu[a,*], Shirui Pan, Peng Zhang[a], Ling Chen[a]

[a] *Quantum Computation and Intelligent Systems (QCIS) Centre,*
Faculty of Engineering & Information Technology, University of Technology Sydney, NSW 2007, Australia.

Abstract

In this paper we study a new problem of online discovering diffusion provenances in large networks. Existing work on network diffusion provenance identification focuses on offline learning where data collected from network detectors are static and a snapshot of network is available before learning. However, an offline learning model does not meet the needs of early warning, real-time awareness, and real-time response of malicious information spreading in networks. To this end, we propose an online regression model for real-time diffusion provenance identification. Specifically, we first use offline collected network cascades to infer the edge transmission weights, and then use an online l_1 non-convex regression model as the identification model. The proposed methods are empirically evaluated on both synthetic and real-world networks. Experimental results demonstrate the effectiveness of the proposed model.

Keywords: Provenance, Social Network, Online Identification, L_1 Regression

1. Introduction

In recent years, information diffusion in large networks has attracted much attention. The spread of malicious information such as viruses, spams and rumors has made various networks vulnerable to privacy attacks, viral advertising, etc. To stop the propagation of malicious information, researchers recently proposed several models to identify the diffusion provenances in large networks. Online diffusion provenance discovery is also a significant presence on social networks in business. No matter how small, medium or large a business is, a brands health and reputation is often defined by the information diffused in social media. While fake reviews or deceptive messages, spread by malevolent people, in social media are inevitable, they may cause damage to brands or corporate reputation. A recent study (*i.e.,* Spam Trends in Today's Business World [1]) reported that the productivity cost of malicious information to European companies was an estimated US$2.8 billion, while US-based companies

*Corresponding author. Tel.: +61 416387666, Fax.: +61 2 9514 4535
Email addresses: haishuai.wang@student.uts.edu.au (Haishuai Wang), jia.wu@uts.edu.au (Jia Wu), shirui.pan@uts.edu.au (Shirui Pan), peng.zhang@uts.edu.au (Peng Zhang), ling.chen@uts.edu.au (Ling Chen)

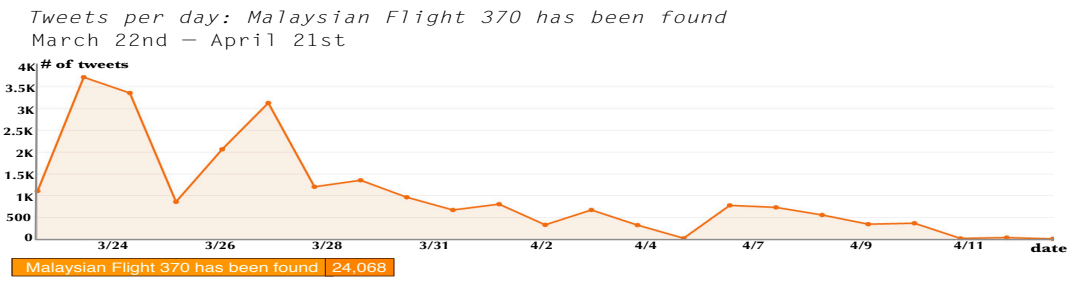


Figure 1: The rumor "*Malaysian Flight 370 has been found*" propagated on Twitter from March 22 to April 21, 2014. The x axis is the time and the y axis is the total number of tweets including the rumor.

reported a loss of US$20 billion. According to The Washington Post [1], on Tuesday, April 24, 2013, a single hoax message sent via Twitter erased $200 billion from the US stock market in 2 minutes. In cases like these, locating and severing the provenances of the diffusion in a timely manner is critical.

A false rumor (*i.e.,* libelous statement) about the financial performance of a firm may be spread by market manipulators to influence the price of the firms stock, resulting in fines from regulators or data protection enforcement agencies. While legal recourse exists for victims of libel, law enforcement agencies still need to identify the original source of the rumor and those who spread it. If the perpetrators are anonymous, tracing the IP address and identities of the individual profiles carrying or linking to the opinion piece becomes significantly more difficult and time-consuming.

Existing diffusion provenance identification models can be roughly categorized into two classes: the snapshot-based provenance identification [2, 3] and the detector-based identification [4, 5, 6]. The snapshot-based methods are under the assumption that a snapshot of the entire network can be obtained and the provenances can be estimated under stochastic propagation models such as the SI [7] and SIR [8] models. Although these methods have shown promising results in experiments, fetching a snapshot of the entire network is very expensive, if not impossible. The detector-based methods assume that only a small subset of nodes in a network can be monitored, and the provenances can be inferred from the observations (samples) from these detectors. This group of methods has recently attracted increasing attention due to its potential usage in real-world applications.

However, to our best knowledge, most existing work on the diffusion provenance locating problem falls into the category of offline identification, where the data are assumed to be static and available all the time. In fact, for time-critical security monitoring applications, it is necessary to unveil the diffusion provenances as soon as an observation arrives. This way, it is important to detect diffusion provenances as early as possible to enable early warning, real-time awareness, and real-time response of malicious information spreading. For example, Fig. 1 shows the propagation of the rumor of *"Malaysian Flight 370 has been found"* on Twitter. The rumor was

[1] http://rt.com/business/tweet-hackers-wall-street-us-326/

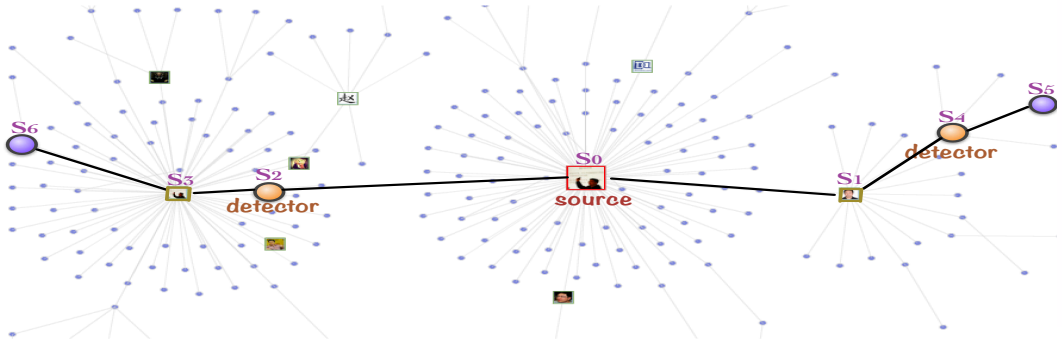


Figure 2: An example of twitter diffusion path. At the unknown time $t = t^*$, the information provenance S_0 initiates the diffusion of a tweet. The propagation time delay between any two nodes is τ and the time window $T = [t^*, t^* + 3\tau]$.

retweeted more than 3,500 times in a single day (3/24). It can be seen that the rumor can be propagated to a large population in a very short time. Hence, it is imperative to detect a rumor provenance promptly.

Motivated by the urgent demand of real-time and continuous diffusion provenance detection in networks, we propose to use regression learning as the basic detection model. The regression learning is favorable for real-time applications due to the freely available prior distribution. Fig. 2 shows a network propagation with only one provenance S_0. We extract a propagation path with seven nodes $\{S_0, \cdots, S_6\}$. Assume that we have the privilege to observe nodes S_2 and S_4 (detectors). The two nodes are activated at time points $t^* + \tau$ and $t^* + 2\tau$ respectively. The goal is *to use the least square regression to minimize the error rate between the observed time delay and the estimated time delay*. Assume at time $t_i \in T = [t^*, t^* + 3\tau]$, the i^{th} node is observed. At any time t_i, we have, $\min_x \sum_{i=2,4} [(t^* + i * \tau) - a_i^T x]^2, \quad s.t. : \ x \geq 0$, where $x \in \mathbb{R}^7$ is the target variable with element x_i denoting the probability that node i is the diffusion provenance, $a_i \in \mathbb{R}^7$ is a column vector with element a_{ij} denoting the propagation path length from the detector i to the j^{th} node, e.g., $a_2 = (1, 2, 0, 1, 3, 4, 2)^T$ and $a_4 = (2, 1, 3, 4, 0, 1, 5)^T$. The objective function is convex and non-negative and achieves its minimum value 0 when $x = (1, 0, 0, 0, 0, 0, 0)^T$. For offline detection, since data from both nodes S_2 and S_4 are known, we can obtain the result as $x = (1, 0, 0, 0, 0, 0, 0)^T$, which correctly indicates that S_0 is the diffusion provenance. However, for online detection, we first estimate x by only observing data from the detector S_2 and the result is $(0, 0.8, 0, 0.2, 0, 0, 0)^T$. When data from S_4 arrive, the result of x is updated to $(1, 0, 0, 0, 0, 0, 0)^T$. We can see that an online algorithm demands dynamic and continuous computation of x, and the result of online learning is expected to approximate (or equal to) the offline result.

Compared to offline identification methods, online identification models have the following challenges:

- *Challenge 1: how to design an online identification model?* The online identification model needs to address five questions, the unknown number of diffusion provenance k, both activated and inactivated detectors, the unknown initial propagation time t^*, the uncertain propagation path, and the uncertain propagation time

delay. The unknown number of diffusion provenance k expects a sparse solution of x. Both activated and inactivated detectors lead to partially labeled data and a non-convex objective function. The uncertain propagation path and propagation time delay demand an aggregate Gaussian distribution to describe the estimated propagation time delay.

- *Challenge 2: how to design a stochastic learning algorithm to solve the online identification model?* Because the detectors are activated sequentially while a decision needs to be made once a detector is activated, the algorithm needs to digest data in a continuous and converging way. Ideally, the results of the online algorithm approximate those of the offline algorithms.

- *Challenge 3: how to evaluate the performance of the proposed online identification method?* Given the unique characteristics of the problem, various data are demanded to evaluate performance.

In this paper, we propose a new online regression learning model to identify diffusion provenances in large networks. To solve *Challenge 1*, we use an l_1 non-convex regression learning model built on top of an aggregate Gaussian propagation time delay, where the network transmission weights are inferred a prior from offline collected cascades. To tackle *Challenge 2*, we present an Online Stochastic Sub-gradient algorithm (OSS for short) that can converge to local minima. Also, we evaluate the model using four synthetic network data to address *Challenge 3*.

The contributions of the work are twofold:

- We present a new online regression learning model to identify diffusion provenances in large networks. The proposed model can handle the issues of the unknown number of diffusion provenances k, the partially activated detectors, the unknown initial propagation time t^*, the uncertain propagation path, and the uncertain propagation time delay.

- We present an online stochastic algorithm to solve the proposed online regression learning model. The algorithm uses a stochastic sub-gradient decent algorithm to continuously detect the provenances.

The remainder of the paper is organized as follows. Section 2 surveys related work. Section 3 introduces the related preliminaries. The regression learning and online algorithm are given in Sections 4 and 5 respectively. Section 6 empirically evaluates the algorithms. We conclude the paper in Section 7.

2. Related Work

Malicious information such as rumors and viruses has been observed recently propagating in networks, which incurs privacy and security concerns [9, 10] and motivates the research of diffusion provenance detection. To date, existing works on diffusion provenance detection focus on offline detection, where a snapshot of a large network or data harvested from detectors are assumed available in advance. In order to design an online detection algorithm,

three technical questions need to be answered: 1) how to design stochastic propagation models, 2) how to design an objective function for online detection, and 3) how to design an online algorithm as the solution. We survey related work regarding the three aspects.

In terms of stochastic propagation models in diffusion provenance locating, existing works simulate the spreading by using infection models such as the *Susceptible-Infected-Recovered* (SIR) [8] model, the *Susceptible-Infected* SI [7] model, and others [11, 12]. On the other hand, recent works [13, 14, 15] claimed that modeling propagation cascades and information diffusion using continuous-time diffusion networks can provide accurate models.

Based on the stochastic models, several learning models were proposed to infer the provenances. For example, a recent work [7] provided a systematic survey of locating rumor provenances in a network, and presented a *rumor centrality estimator* to estimate the rumor provenances by assigning a score to each infected node. The work [16] studied the problem of a single rumor provenance locating with *priori knowledge*. Most existing estimators are based on either topological centrality measures [17] or distance measures between observed data. Then, maximum likelihood estimator can be used to infer the provenances.

A limitation of the above works is that they all assume that the infection status of the nodes (i.e., labels) is known a prior. For example, the work [4, 7] considered the multiple infection provenances estimation problem and assumed that the number of infection provenances is unknown in advance. Some work [6] assumed that not all nodes are infected and only a subset of detectors are used for the provenance estimation.

For online algorithms, online learning have been extensively studied in machine learning. Typical methods include the Passive-Aggressive (PA) [18] and *truncated gradient* algorithms [19]. However, these online learning algorithms are based on linear and convex optimization, which do not fit our non-convex online learning problem.

Despite the complexity of inferring the diffusion provenances in a network, a simple heuristic is to say that the provenance is the center of the network [20]. There are many notions of network centrality, but a very common one is known as distance centrality, e.g., betweenness centrality [21], closeness centrality [22] and Bonacich centrality [23]. Betweenness centrality measures a nodes centrality in a network. The infection closeness centrality heuristic claims the node with the maximum infection closeness is the source. Bonacich Centrality is a measure of the influence of a node in a network. One may argue that the most influential nodes are more likely to be the provenances of the diffusion. However, actually, an almost isolated node that has few connections to the most influential nodes is probably the source. Therefore, traditional central-based algorithms are hardly applicable.

To sum up, none of the aforementioned works can be directly used to address the online diffusion provenance detection problem studied in this work.

3. Preliminaries

Consider a network $G = \{V, E\}$, where the vertex set V has N nodes, and the edge set E has L edges. In the network, we have two types of data for model training:

1) *Offline data of cascades*: a set C of cascades $\{c_1, \cdots, c_n\}$, where each cascade c is a sequence of activated times $\{t_1, \cdots, t_N\}$ within a given time window T. Each time point t_i records the i^{th} activated node. For nodes that are not activated within window T, the activated time is unknown.

2) *Online data collected from detectors*. We budget a small subset of detectors, denoted by $\mathcal{D} = \{d_i\}_{i=1}^{m}$, to monitor the network. During the monitoring time window $[t^*, t^* + T]$, where t^* is the initial propagation time and T is the size of the window, there are a subset of detectors activated, denoted by D_a, and the remaining inactivated detectors are denoted by D_u. We aim to estimate the locations of provenances, denoted by a random vector $s^* \in \mathrm{R}^N$, given the status of all the detectors $D = \{(d_1, t_1 - t^*), \cdots, (d_m, t_m - t^*)\}$, where t_m is the activated time point of detector d_m and label $t_m - t^*$ denotes the time delay. Note that time labels of inactivated nodes are unknown, and their time delay exceeds window T.

We adopt an SI propagation model to describe how the infection spreads in network G. The reasons for using this kind of propagation model are that most posts on social networks are usually not removed, i.e., an infected node stays infected. Thus, SI is an appropriate model for online postings and modeling opinion dynamics on social networks. In the SI model, each node in a network has three possible states: susceptible (s), infected (I) and non-susceptible (n). As the infected nodes are those nodes that possess the infection, and will remain infected throughout, an infected detector cannot be recovered. Therefore, a detector cannot receive the same information multiple times and will not send the same information multiple times to the same node.

Now we infer the **edge transmission probability matrix** W based on the cascades collected offline. Given a cascade $c_t \in C$, we use $f(c_t|W)$ to denote the likelihood function of observing the cascade c_t under an unknown transmission matrix W. The likelihood function consists of the joint probability of activated nodes $v_i \in V$ ($t_i \leq T$) and inactivated nodes $v_m \in V$ ($t_m > T$). For an activated node $v_i \in V$ activated at time $t_i \leq T$ from an adjacent node $v_j \in V$ and $t_j < t_i$, the probability of observing v_i activated by v_j at time t_i is a joint probability of v_j infecting v_i at time t_i and v_i is not activated by any other neighbors v_k which have been already activated by the time t_i, i.e., $t_k < t_i$. By summing up all the possible neighbors v_j in the network, i.e., summing up all v_j with $t_j < t_i$, we can achieve the probability $f(c_{t_i}|W)$ of a node v_i activated at time t_i as in Eq. (1),

$$f(c_{t_i}|W) = \sum_{\substack{v_j \in g(v_i) \\ t_j < t_i}} \Big[P(t_j \to t_i|W) \prod\nolimits_{\substack{v_k \in g(v_i) \\ v_k \neq v_j, t_k < t_i}} P(t_k \nrightarrow t_i|W) \Big]. \tag{1}$$

where $g(v_i)$ denotes the set of neighbors to v_i, and $P(t_j \to t_i|W)$ denotes the probability of the j^{th} activated node v_j activates its neighbor v_i at time t_i. We can use three well-known parametric functions for $P(t_j \to t_i|W)$, such as the widely used *exponential*, *power-law* and *Rayleigh* models [13]. Without loss of generality, we use the exponential model, where $P(t_j \to t_i|W)$ equals $w_{ji} e^{-w_{ji}(t_i - t_j)}$ if $t_j < t_i$ and 0 otherwise. The probability $P(t_j \nrightarrow t_i|W) = 1 - P(t_j \to t_i|W)$.

Based on Eq. (1), the probability of observing all the activated nodes in a cascade c_t is as follows,

$$f(c_{t \leq T}|W) = \prod\nolimits_{v_i : t_i \leq T} f(c_{t_i}|W). \tag{2}$$

Because inactivated nodes also provide information for transmission weight estimation, the probability of observing an entire cascade c_t , $f(c_t|W)$, is a joint probability of observing all the activated nodes in the cascade c_t and unobserving the remaining inactivated nodes as in Eq. (3),

$$f(c_t|W) = f(c_{t\leq T}|W) \times \left(\prod\nolimits_{v_m:t_m>T} \prod\nolimits_{\substack{v_i\in g(v_m) \\ t_i\leq T}} P(t_i \nrightarrow t_m|W) \right). \tag{3}$$

The likelihood function of observing all cascades C is the product of all the likelihoods of each cascade given in Eq. (3),

$$L(W) = log \prod_{c_t\in C} f(c_t|W) = \sum_{c_t\in C} log f(c_t|W) \tag{4}$$

Then, the edge transmission matrix W can be estimated as follows,

$$W^* = argmax\, L(W),\ s.t., W \geq 0. \tag{5}$$

For simplicity, we denote $\phi(W; j, i) = P(t_j \rightarrow t_i|W)$, then Eq. (4) can be rewritten as in Eq. (6),

$$\begin{aligned} L(W) = \sum_{c_t\in C} \Big(\sum_{\substack{v_i \\ t_i\leq T}} log \sum\nolimits_{\substack{v_j\in g(v_i) \\ t_j<t_i}} \frac{\phi(W; j, i)}{1-\phi(W; j, i)} + \sum\nolimits_{\substack{v_i \\ t_i\leq T}} \sum\nolimits_{\substack{v_k\in g(v_i) \\ t_k<t_i}} log[1-\phi(W; j, i)] \\ + \sum\nolimits_{\substack{v_m: \\ t_m>T}} \sum\nolimits_{v_i\in g(v_m)} log[1-\phi(W; i, m)] \Big) \end{aligned} \tag{6}$$

The likelihood in Eq. (6) is convex, and thus the first-order gradient guarantees a global optimum. By letting the derivative of Eq. (6) to 0, we obtain $\phi'(W_{ji}) = 0$ for each cascade. That is, $W_{ji} = \frac{1}{t_i-t_j}$ for each observed pair of neighboring nodes. For each inactivated node $t_m > T$ in a cascade, as we don't observe the exact time of t_m, we approximately let $t_m = T$ and $W_{im} = \frac{1}{T-t_i}$. Moreover, we set the default propagation probability of each edge to be $W_{ji} = \frac{1}{T}$ in case that we don't observe any propagation data between nodes v_j and v_i in a cascade. Then, by averaging over $|C|$ cascades, we can obtain the final result as follows,

$$W^*_{ji} \propto \begin{cases} \frac{1}{\kappa} \sum_{c_t\in C} I_{c_t}(v_i, v_j) \frac{1}{t_i-t_j}, & t_i \leq T, t_j < t_i; \\ \\ \frac{1}{\kappa} \sum_{c_t\in C} I_{c_t}(v_i, v_j) \frac{1}{T-t_i}, & t_i > T, t_j \leq T. \end{cases} \tag{7}$$

where $I_{c_t}(v_i, v_j)$ is an indicator and equals to 1 if the cascade c_t satisfies the time constraint. κ is the total number of node pairs that meet the constraint. If the given time constraints are not met, the default weigth W_{ji} is $1/T$.

4. Regression Model

In this part, we formulate the objective function for diffusion provenances detection based on *online data collected from the detectors*. We assume that the prior distribution of s^* is uniform over the network, i.e., any node in the network is equally likely to be the provenance. Thus, the location of the provenances can be recovered by

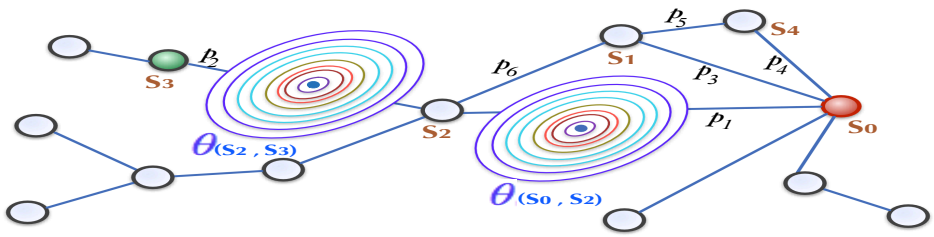


Figure 3: Gaussian time delay and the shortest-path propagation. The propagation path from the provenance s_0 to detector S_3 is approximated by the shortest path $\mathcal{P}(S_0, S_3) = \{S_0 \to S_2 \to S_3\}$. The propagation delay is an aggregate Gaussian distribution of paths p_1 and p_2.

maximizing the likelihood function of the observed status of the detectors D given provenances $s \in G$, as shown in Eq. (8),

$$s^* = \arg\max_{s \in G} P(D|s) = \arg\max_{s \in G} P(D_a|s)\Big[1 - P(D_u|s)\Big] \tag{8}$$

where $P(D_a|s)$ denotes the probability of observing D_a given a set of spread provenances s, and $1 - P(D_u|s)$ denotes the probability that D_u is inactivated under provenances s. Thus, $P(D_a|s)[1 - P(D_u|s)]$ denotes the joint probability of both observing the activated nodes D_a and inactivated nodes D_u.

Because malicious information such as virus and rumors are often spread under *the snowball phenomenon* [24], it is reasonable to approximately use the shortest-path spread, denoted by $\mathcal{P}(i, j)$ between nodes $v_i, v_j \in V$.

Because the offline obtained edge transmission matrix W may suffer unstable change when used online, we further use a random variable $\theta_{i,j}$ to describe delay time along edge $e_i \in E$. The random variables are independent identically distributed with Gaussian distribution $\theta_{i,j} \sim N(\mu_{i,j}\sigma^2_{i,j})$, where the mean $\mu_{i,j}$ are from matrix W.

Figure 3 shows an example of the Gaussian time delay and the shortest-path propagation. In the worst case, the search for the shortest path takes time $O(N^2)$.

Thus, the probability $P(D_a|s)$ in Eq. (8) can be rewritten as follows,

$$P(D_a|s) = \prod_{i=1}^{m_a} \sum_{\prod_s^a} P(\prod_s^a|s)P(d_a|\prod_s^a) = \prod_{i=1}^{m_a} P(\theta_{s,d_a}), \tag{9}$$

where m_a is the number of observers during the time window, and θ_{s,d_a} is the time delay. Based on the Gaussian distribution $\theta_{i,j} \sim N(\mu_{i,j}\sigma^2_{i,j})$, the mean and variance of the random variable θ_{s,d_a} are $\mu_a^T x$ and $(\sigma_a^2)^T x$ respectively. Let $\Lambda = (\sigma_a^2)^T x$, the Eq. (9) can be converted to Eq. (10),

$$P(D_a|s) = \prod_{i=1}^{m_a} (2\pi\Lambda)^{-1/2} e^{-(t_a - t^* - \mu_a^T x)^2/(2\Lambda)}. \tag{10}$$

Similarly, for unobservers, let $\Lambda' = (\sigma_u^2)^T x$, we have

$$P(D_u|s) = \prod_{i=1}^{m_u} (2\pi\Lambda')^{-1/2} e^{-(t_u - t^* - \mu_u^T x)^2/(2\Lambda')}. \tag{11}$$

Thus, Eq. (8) can be rewritten as

$$P(D|s) = \prod_{i=1}^{m_a} (2\pi\Lambda)^{-1/2} e^{-(t_a - t^* - \mu_a^T x)^2/(2\Lambda)} \cdot \prod_{i=1}^{m_u} [1 - (2\pi\Lambda')^{-1/2} e^{-(t_u - t^* - \mu_u^T x)^2/(2\Lambda')}]. \tag{12}$$

Let all time delays share the same variance σ^2, then the log-likelihood function in Eq. (12) is

$$lnP(D|s) = -\frac{1}{2\Lambda}\sum_{i=1}^{m_a}(t_a - t^* - \mu_a^T x)^2 + \frac{1}{2\Lambda}\sum_{i=1}^{m_u}(t_u - t^* - \mu_u^T x)^2 + C, \tag{13}$$

where C is a constant, $t_{a(u)} - t^*$ is the observed (unobserved) time delay.

Then, maximizing the log-likelihood[2] in Eq. (13) is tantamount to minimizing a quadratic regression function in Eq. (14), where the target vector $t \in \mathrm{R}^{m_a}$ is the observed time delay, i.e., $t_a - t^*$ for detector d_a, the coefficient matrix $A = (\mu_1^T, \cdots, \mu_{m_a}^T) \in \mathrm{R}^{N*m_a}$ is the shortest path time delay *w.r.t.* nodes $d_a \in D_a$. $\mathcal{T} \in \mathrm{R}^{m_u}$ is a vector of value T, which approximates the time delay of nodes D_u, i.e., $t_u - t^* \approx (T + t^*) - t^* = T$. Matrix $B = (\mu_1^T, \cdots, \mu_{m_u}^T) \in \mathrm{R}^{N*m_u}$ denotes the shortest time delay *w.r.t.* inactivated nodes $d_u \in D_u$. The parameter $\lambda > 0$ controls the weight. The constraints guarantee the optimal solution sparse and non-negative.

$$s^* = \arg\min_x \frac{1}{2}\|t - A^T x\|_2^2 - \lambda\|\mathcal{T} - B^T x\|_2^2 \quad s.t.: \ e^T x \le \tau, \ \ x \ge 0. \tag{14}$$

The above formulation relaxes traditional discrete optimization on graphs. Such a relaxation leads to efficient algorithms. Now we intuitively **explain** the above problem from the viewpoint of *multi-criteria quadratic programming*. The first objective function, $\|t - A^T x\|_2^2$, aims to fit time delay of activated detectors D_a, the second objective function $\|\mathcal{T} - B^T x\|_2^2$ aims to fit time delay of inactivated detectors D_u. Because D_u are not actually activated during the time window T, a negative parameter $-\lambda < 0$ is used to avoid the fitting. The trade-off *Pareto solutions* can be obtained by varying parameter λ.

5. Online Algorithm

In this section, we present an Online Stochastic Sub-gradient algorithm to solve the regression model in Eq. (14). The proposed regression model, compared to the classical regression learning, has several new challenges: 1) the dependent variable t is implicit, because the initial propagation time is unknown, 2) the l_1 non-convex objective function expects sparse and fast convergent algorithm, and data collected from detectors arrive continuously. To solve the challenges, we use the *Relative Time Difference of Arrivals* and *Online Sub-gradient based on convex approximation* as the solutions.

5.1. Relative Time Difference

The dependent variable t in Eq. (14) is implicit, because the initial propagation time t^* is unknown. To solve this challenge, we can use an "anchor node" to cancel out the initial time t^*. Assume the α^{th} detector d_α is the "anchor node", its activated time is $t_\alpha = t^* + \sum_{e_i \in \mathcal{P}(s^*, d_\alpha)} \theta_i$, where θ_i is the time delay along edges $e_i \in \mathcal{P}(s^*, d_\alpha)$.

[2] the log-likelihood is $\ln P(D|s) \propto -1/2\sum_{i=1}^{m_a}(t_a - t^* - \mu_a^T x)^2 + \lambda\sum_{i=1}^{m_u}(t_u - t^* - \mu_u^T x)^2$, where $t_a - t^*$ is the observed time delay, and the same goes to inactivated nodes d_u. The second part is obtained because $ln(1 - e^x) \approx -e^x \approx -x$.

Then, the *Relative Time Difference of Arrivals* (RTDA) between d_α and the k^{th} ($k \neq \alpha, 1 \leq k \leq m_a$) detector d_k is $\bar{t}_k := t_k - t_\alpha = \sum_{e_i \in \mathcal{P}(s^*, d_k)} \theta_i - \sum_{e_i \in \mathcal{P}(s^*, d_\alpha)} \theta_i$.

Based on RTDA, Eq. (14) has an elementary column change, i.e.,

$$\overline{A}(:, c_k) := A(:, c_k) - A(:, c_\alpha), \quad \overline{B}(:, c_k) := B(:, c_k) - B(:, c_\alpha). \tag{15}$$

Thus, the dimension of $\bar{t}$, $\overline{A}$ and $\overline{B}$ are $\bar{t} \in \mathbb{R}^{m_a - 1}$, $\overline{A} \in \mathbb{R}^{N*(m_a-1)}$ and $\overline{B} \in \mathbb{R}^{N*(m_u-1)}$ respectively. Then, Eq. (14) can be relaxed to Eq. (16),

$$s^*(x) = arg \min_x \frac{1}{2} \|\bar{t} - \overline{A}^T x\|_2^2 - \lambda \|\overline{B}^T x\|_2^2 + \rho \|x\|_1^1. \tag{16}$$

5.2. Convex Approximation

To address the non-convex challenge, we wish to find a sequence of convex programs, through which the non-convex function can be approximated and converge to a local minimum. To obtain a convergent sequence, at step $k+1$, the concave part $-\lambda \|\overline{B}^T x\|_2^2$ is linearly approximated by using the differential at the previous iterative point x_k, i.e.,

$$\frac{\partial}{\partial x}(-\lambda \|\overline{B}^T x\|_2^2)|_{x_k}, \quad x \geq -2\lambda \overline{B}^T \overline{B} x_k x. \tag{17}$$

At step $k+1$, we only solve a convex optimization as follows:

$$x_{k+1} \leftarrow \left\{ \min_{x \geq 0} \frac{1}{2} \|\bar{t} - \overline{A}^T x\|_2^2 - 2\lambda \overline{B}^T \overline{B} x_k x + \rho \|x\|_1^1 \right\}. \tag{18}$$

Lemma 1. The non-convex program in Eq. (16) converges under iterations $\{x_1, \cdots, x_k, \cdots\}$ generated by Eq. (18).

Proof. Denote Eq. (16) as $J(x)$, which is a combination of the convex part $J_{vex}(x) = \frac{1}{2} \|\bar{t} - \overline{A}^T x\|_2^2$ and the concave part $J_{cav}(x) = -\lambda \|\overline{B}^T x\|_2^2$. At step $k+1$, we have $J_{vex}(x_{k+1}) + J'_{cav}(x_k) x_{k+1} \leq J_{vex}(x_k) + J'_{cav}(x_k) x_k$. Moreover, due to the definition of concavity, $J_{cav}(x_{k+1}) \leq J_{cav}(x_k) + J'_{cav}(x_k)(x_{k+1} - x_k)$. Adding both sides of the above two inequalities, we obtain the result $J(x_{k+1}) \leq J(x_k)$. Therefore, Eq. (16) under the sequence $\{x_1, \cdots, x_k, \cdots\}$ generated by Eq. (18) is convergent. □

5.3. Online Sub-gradient

We use the *sub-gradient* method to solve the l_1 regularization problem in Eq. (16). Let $\mathcal{J}(\mathbf{x}) = F(\mathbf{x}) + G(\mathbf{x}) = \frac{1}{2} \|\bar{t} - \overline{A}^T x\|_2^2 - 2\lambda \overline{B}^T \overline{B} x_k x$, which is an approximation of the first two parts in Eq. (16) by using the CCCP programming [25] at the concave part $-\lambda \|\overline{B}^T x\|_2^2$. The sub-gradient of $\mathcal{J}$ is as follows,

$$\nabla_j \mathcal{J}(x) = F'_j(x) + G'_j(x) = \frac{1}{2} [\sum_{i=1}^{m_a} (\bar{t}_i - a_i{}^T x)^2]'_j + [-2\lambda \sum_{i=1}^{m_u} (b_i{}^T b_i x_k x)]'_j = -(\overline{A}^T \bar{t} - \overline{A}^T \overline{A} x)_j - 2\lambda \overline{B}^T \overline{B} x_k.$$

Function 1 Online Detecting During the Time Window

Input:

$\overline{A}$: Observation matrix;

$\bar{t}$: Time delay;

ϵ: Stop criteria;

η_t: Learning rate;

Output:

$\mathcal{X}^* = \{x_1, \cdots, x_j\}$: The indicator vector $\mathcal{X}^*$;

$x_{k+1} \leftarrow$ a best guess;

$\mathcal{L}(\mathbf{x}) = \boldsymbol{f}(\mathbf{x}) = \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2$

$\hat{s}^*(x) = arg\min_{x \geq 0} \mathcal{L}(x) + \rho\|x\|_1^1$

repeat

  $x_k \leftarrow x_{k+1}$;

  for $i = 1$ *to* m_a **do**

    $\nabla\mathcal{L}(x) = (\bar{t}_i - a_i x)^T(-a_i)$;

    $x_{k+1} \leftarrow \left\{x_k - \eta_t(\nabla\hat{s}^*(x))\right\}$;

  end for

until $\|x_{k+1} - x_k\| \leqslant \epsilon$

$\mathcal{X}^* \leftarrow x_{k+1}$ in a descending order;

j^* =argmax$_j$ $|x_j^* - x_{j-1}^*|$;

return $\mathcal{X}^* = (x_1, \cdots, x_{j^*})$;

Now Eq. (16) turns to $s^*(x) = arg\min_x \mathcal{J}(x) + \rho\|x\|_1^1$ and this objective function is non-differentiable. Assume $\overline{X} = (\overline{x}^1, \cdots, \overline{x}^n)^T$ is the global optimal point. Consider the j^{th} variable $\overline{x}^j$. The first-order optimality conditions are:

$$\begin{cases} \nabla_j \mathcal{J}(x) + \rho sign(\overline{x}^j) = 0, & s.t.\ |\overline{x}^j| > 0 \\ \left\{ \nabla_j \mathcal{J}(x) + \rho e : e \in [-1, 1] \right\}, & s.t.\ \overline{x}^j = 0 \end{cases} \tag{19}$$

where $sign(\overline{x}^j) = 1, \overline{x}^j > 0; -1, \overline{x}^j < 0; 0, \overline{x}^j = 0$.

These conditions can be used to define a sub-gradient for each $\overline{x}^j$:

$$\nabla_j s^*(x) = \begin{cases} \nabla_j \mathcal{J}(x) + \rho sign(\overline{x}^j), & |\overline{x}^j| > 0 \\ \nabla_j \mathcal{J}(x) + \rho, & \overline{x}^j = 0, \nabla_j \mathcal{J}(x) < \rho \\ \nabla_j \mathcal{J}(x) - \rho, & \overline{x}^j = 0, \nabla_j \mathcal{J}(x) > \rho \\ 0, & \overline{x}^j = 0, -\rho \leq \nabla_j \mathcal{J}(x) \leq \rho \end{cases}. \tag{20}$$

Thus, based on the three solutions, the sub-gradient method uses iterations:

$$x_{k+1} = x_k - \eta_t(\nabla s^*(x)), \tag{21}$$

Function 2 Online Detecting After the Time Window

Input:

$\overline{A}$: Observation matrix;

$\overline{B}$: Unobservation matrix;

$\bar{t}$: Time delay;

ϵ: Stop criteria;

η_t: Learning rate;

Output:

$\mathcal{X}^* = \{x_1, \cdots, x_j\}$: The indicator vector $\mathcal{X}^*$;

$x_{k+1} \leftarrow$ a best guess;

repeat

  $x_k \leftarrow x_{k+1}$;

  $x_{k+1} \leftarrow \left\{x_k - \eta_t(\nabla s^*(x))\right\}$;

until $\|x_{k+1} - x_k\| \leqslant \epsilon$

$\mathcal{X}^* \leftarrow x_{k+1}$ in a descending order;

$j^* = \underset{j}{\operatorname{argmax}} |x_j^* - x_{j-1}^*|$;

return $\mathcal{X}^* = (x_1, \cdots, x_{j^*})$;

where the parameter $\eta_t > 0$ is the learning rate. In our analysis, we only consider constant learning rate with a fixed $\eta_t > 0$.

5.4. The Online Stochastic Sub-gradient (OSS) Algorithm

In this part, we design an Online Stochastic Sub-gradient detection algorithm to continuously infer the diffusion provenances. Algorithm 1 summarizes the solution to Eq. (16), where the sparse non-convex regression is solved by iteratively calculating the convex program in Eq. (18) and the non-smooth l_1 program in Eq. (19). In terms of online learning, we use stochastic sub-gradient iterations to calculate $\nabla_j \mathcal{J}(x)$ during the time window T, i.e.,

$$\nabla_j \mathcal{J}(x) = (\bar{t}_i - a_i x)^T (-a_i). \tag{22}$$

As shown in Algorithm 1, the proposed online algorithm calls two functions for continuous learning. Function 1 corresponds to the online computation within the time window T where the detectors are activated one by one. Function 2 corresponds to the one-time computation after the time window T. The corresponding overall framework can be found in Figure 4.

Based on the input of the proposed algorithm and the overall framework in Figure 4, there are some assumptions for the OSS algorithm. 1) At least 1 detector needs to be activated during the time window. We used the time delay of both activated detectors (matrix A and D_a) and inactivated detectors (matrix B and D_u) in the proposed algorithm. During the time window, Function 1 is only called for activated detectors, and each is activated sequentially. After

Algorithm 1 The Online Detection Algorithm

Input:

$\mathcal{G}$: Network Graph;

$\mathcal{D}$: Detectors $\{D_a \cup D_u\}$;

u: The propagation mean parameters $\mu = (\mu_1, \cdots, \mu_{|E|})^T$;

σ: The propagation variance parameters $\sigma^2 = (\sigma_1^2, \cdots, \sigma_{|E|}^2)^T$;

ϵ: Stop criteria;

Output:

$\mathbf{s}^* = \{s_1, \cdots, s_m\}$: A set of diffusion provenances $\mathbf{s}^*$;

$d_{\alpha(\beta)} \leftarrow$ anchor$(D_{a(u)})$; // randomly pick anchors
if $t < T$ **then**
  for each $d_i \in D_a$ **do**
    for each $v_j \in \mathcal{V}$ **do**
      $\mathcal{P}(d_j, v_j) \leftarrow$ BFS $(\mathcal{G}, d_i, v_j)$; // breath first
      $A_{ij} = \sum_{e_k \in \mathcal{P}(d_i, v_j)} \mu_k$; // matrix A
      $B_{ij} = \sum_{e_k \in \mathcal{P}(d_i, v_j)} \mu_k$; // matrix B
      if $i > \alpha$ **then**
        for each $k \leq m_a$ **do**
          $\overline{A}(:, c_k) \leftarrow A(:, c_k) - A(:, c_\alpha)$; // matrix $\overline{A}$
          $\bar{t} := t_k - t_\alpha$; // build $\bar{t}$
        end for
        $\mathcal{X}^* \leftarrow$ Function 1$(\overline{A}, \bar{t}, \epsilon, \eta_t)$; // Call Function 1
      end if
    end for
  end for
else
  for each $k \leq m_u$ **do**
    $\overline{B}(:, c_k) \leftarrow B(:, c_k) - B(:, c_\beta)$; // matrix $\overline{B}$
  end for
  $\mathcal{X}^* \leftarrow$ Function 2$(\overline{A}, \overline{B}, \bar{t}, \epsilon, \eta_t)$; // Call Function 2
end if
return $\mathbf{s}^* \leftarrow \mathcal{G}(\mathcal{X}^*)$;

the time window, Function 2 is called because data from both the activated and inactivated detectors are available. Therefore, the proposed algorithm can be used to infer diffusion provenances when there is at least one activated detector (*i.e.,* the activated detector set D_a is non-null and Function 1 can be called). 2) The network structure is known prior, *i.e.,* the network graph G, the propagation mean μ and variance σ.

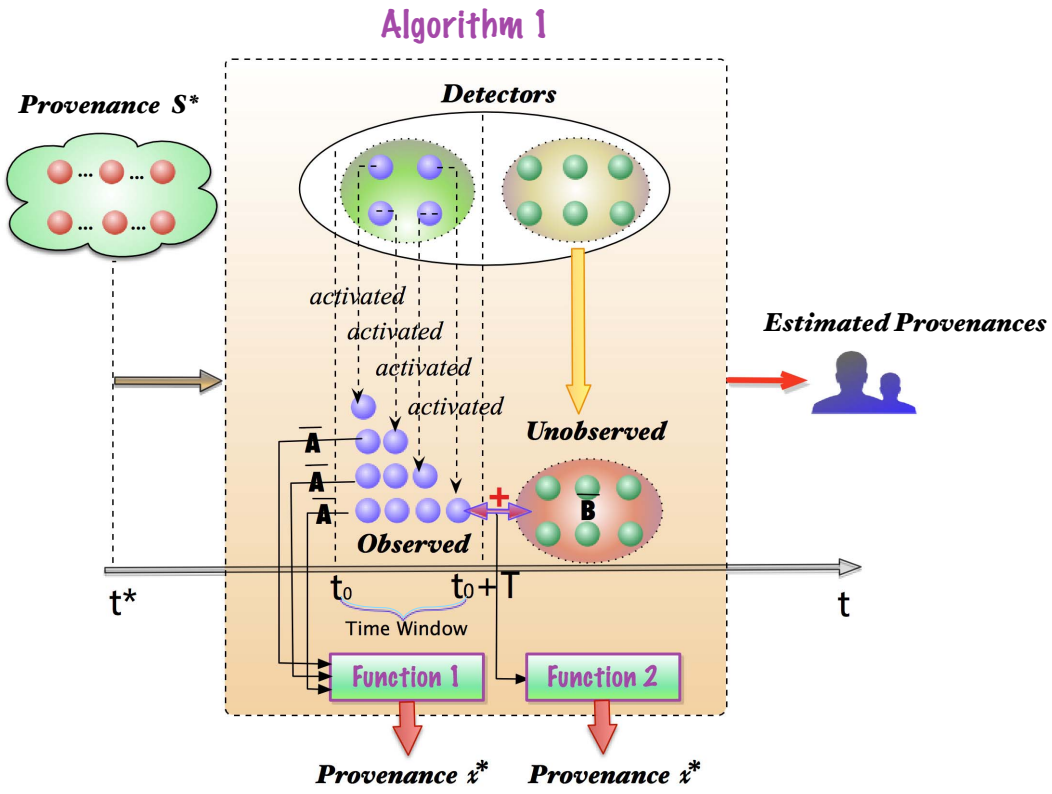


Figure 4: An illustration of the *OSS* algorithm. Detectors are split into two sets. The first set is observed (activated) within the time window, and the second set is unobserved (inactivated) outside the time window. Function 1 is called by OSS within the time window T, and Function 2 is called after the time window T.

In the sequel, we introduce the two functions respectively.

Function 1: During the time window T, detectors are activated sequentially. To conduct continuous detection, once a detector is activated, the regression learning model is used for provenance estimation. At this stage, data from the inactivated nodes are unavailable, the parameter λ in Eq. (16) equals 0, i.e.,

$$s^*(x) = arg\min_{x\geq 0} \frac{1}{2}\|\bar{t} - \overline{A}^T x\|_2^2 \ + \ \rho\|x\|_1^1. \tag{23}$$

Each row of A and t in Eq. (23) denotes the shortest path and time delay from an observed detector to other detectors. By choosing an anchor node, Eq. (15) is used to calculate the matrix $\overline{A}$ and $\bar{t}$ in Eq. (23). Due to the increasing of the number of activated detectors, matrix $\overline{A}$ and $\bar{t}$ in Eq. (23) increase dynamically.

Function 2: After the time window T, data from both the activated and inactivated nodes are available. Thus, we can use Eq. (16) to locate the diffusion provenances.

We use Figure 4 to show the procedure of the online learning: (1) During the time window T, the detectors are activated one by one. Once a detector is activated, we use Function 1 to estimate the provenance; (2) After the time

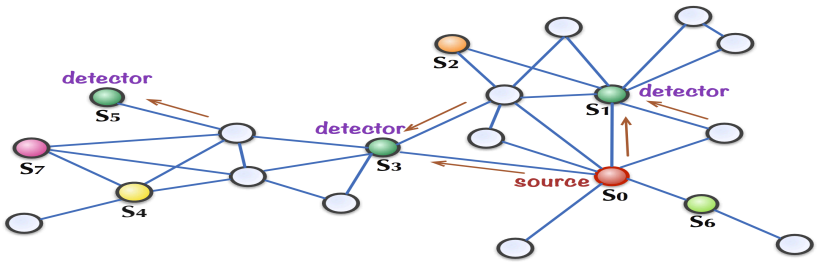


Figure 5: A network with one provenance S_0, and three detectors S_1, S_3 and S_5. At the unknown time $t = t^*$, propagation starts from S_0. Time delay along each edge is $\theta_i \sim N(1, 0.01)$. Monitoring time window $T = [t^*, t^* + T]$, where T is the size of the time window. Detector S_1 is activated at time point $t = t^* + \frac{1}{2}T$. Detectors S_3 and S_5 are inactivated during time window T. We only consider eight nodes $\{S_0, \cdots, S_7\}$.

window, we obtain all the activated detectors and inactivated detectors, and Function 2 can be used for detection.

Example: Consider a network in Figure 5. Assume $t^* = 1, D = 3$, and the time window T is $[1, 5]$. Also, we assume detectors S_1, S_3 and S_5 are observed at time $t_1 = 1, t_3 = 3$ and $t_5 = 5$ respectively. Let S_1 be the anchor node. At $t = 1$, we have $A = [1, 0, 1, 2, 4, 4, 2, 4]^T$. Then, at $t = 3$, S_3 is activated, and we update the matrix A and t as follows,

$$A = \begin{bmatrix} 1 & 0 & 1 & 2 & 4 & 4 & 2 & 4 \\ 1 & 2 & 2 & 0 & 2 & 2 & 2 & 2 \end{bmatrix}^T, \quad t = \begin{bmatrix} 1 - t^* \\ 3 - t^* \end{bmatrix}.$$

The A and t are used to calculate $\overline{A}$ and $\bar{t}$ by Eq. (12),

$$\overline{A} = \begin{bmatrix} 0 & 2 & 1 & -2 & -2 & -2 & 0 & -2 \end{bmatrix}^T, \quad \bar{t} = [2].$$

At the last step, $t = 5$ and S_5 is observed. The variables $\overline{A}$ and $\bar{t}$ are continuously updated to

$$\overline{A} = \begin{bmatrix} 0 & 2 & 1 & -2 & -2 & -2 & 0 & -2 \\ 2 & 4 & 3 & 0 & -2 & -4 & 2 & -2 \end{bmatrix}^T, \quad \bar{t} = \begin{bmatrix} 2 \\ 4 \end{bmatrix},$$

$$\overline{B} = \begin{bmatrix} 0 & 2 & 2 & 0 & 0 & 0 & -2 & 0 \\ 2 & 4 & 3 & 0 & -3 & -2 & 2 & -4 \end{bmatrix}^T.$$

The corresponding online solutions of x approximate the offline optimal solution when the time window T ends. We will leave the competitive boundary analysis in the future work. We have used Figure 5 to show how to calculate matrix $\overline{A}$, $\overline{B}$ and t in the above example. Figure 5 shows the shortest path from nodes $\{S_0, \cdots, S_7\}$ to anchor node S_1. For example, the elements value in matrix A are based on calculating shortest path to S_1 in Figure 5.

6. Experiments

In this section, we report experimental results on two synthetic network data sets and two real-world data sets. The experiments are designed to validate 1) the optimal parameters for the new model, 2) the superiority of the

Table 1: List of the four data sets.

DataSet	Nodes	Edges	Avg. Degree	Avg. Path length	Avg. Clustering Coefficient	Other parameters
Albert-Barabasi	1000	3990	7.98	3.172	0.037	$n = 4$
Small World	1000	3999	7.99	5.079	0.473	$\alpha = 4, p = 0.1$
Twitter	10269	36173	30.54	5.702	0.627	$\mu = 0.01, \sigma = 0.01$
Facebook	3320	10352	19.78	3.892	0.671	$\mu = 0.01, \sigma = 0.01$

proposed model compared with benchmark methods, and 3) the performance of the proposed methods.

6.1. Experimental Data

We use two synthetic data sets (*Albert Barabasi [26] and Small world [27]*) and two real-world data sets (*Twitter* and *Facebook* from SNAP [3]) for parameter study, performance testing and algorithm comparison. The parameter settings are listed in Table 1.

Barabasi-Albert generates random scale-free networks using a preferential attachment mechanism. The network begins with an initial connected network containing β_0 nodes. New nodes are added to the network one at a time. Each new node is connected to $\beta \leq \beta_0$ existing nodes with a probability proportional to the number of links that existing nodes already have. Formally, the probability p_i that the new node is connected to node i is $p_i = k_i / \sum_j k_j$, where k_i is the degree of node i and the sum is made over all pre-existing nodes j. In this model, we set the parameter $n = 4$, which denotes the number of edges created by each new node.

Small World is defined as a network in which the typical distance ζ between two randomly chosen nodes (the number of steps required) grows proportionally to the logarithm of the number of nodes N in the network, that is $\zeta \propto \log N$. We use parameter α to denote that each node is connected to α nearest neighbors in the topology, and p denotes the rewiring probability. In particular, we set $alpha = 4$ and $p = 0.1$ in our experiments.

The original datasets only contain network structure information without time propagation labels. We use Gaussian distribution $N(1, 0.001)$ as the time delay distribution. We simulated the propagation by randomly setting five diffusion sources.

We used the Independent Cascade Model to generate cascades. When node u becomes active, it has a single chance of activating each currently inactive neighbor v. The attempt to activate succeeds independently with a probability of p_{uv}, set to $p_{uv} = 0.5$ as a constant in our experimental setting. Because the cascades are generated by the Independent Cascade Model, each node has a single chance of activating each currently inactive neighbor, so this is one-to-one communication. Thus, the datasets can be considered as half-real data.

[3] http://snap.stanford.edu/data

6.2. Experimental Setup

We assess our methods *w.r.t* the average distance (hops) between actual locations of the diffusion provenances and the estimated locations. Smaller hops indicate that the algorithms have higher accuracy. Let S^* denote the actual provenance, and the estimated provenance set as $\hat{S}$. Since the size of S^* may not be equal to that of $\hat{S}$, we measure their distance $h(S^*, \hat{S})$ by calculating the average number of hops between each element $s_i \in \hat{S}$, $h(S^*, \hat{S}) = \frac{1}{|S^*|} \sum_{i=1}^{|\hat{S}|} \|\hat{s}_i - f(S^*, \hat{s}_i)\|^2$, where $f_i(S^*, \hat{s}_i)$ selects the node in S^* that is closest to $\hat{s}_i$, and $\|\hat{s}_i - F_i(S^*, \hat{s}_i)\|^2$ denotes the hops between nodes $\hat{s}_i$ and $f(S^*, \hat{s}_i)$.

6.3. Experimental Results

Parameter study. We first test the parameters in Eq. (16) *w.r.t.* the number of diffusion provenances k, the number of detectors m, the length of monitoring time window T, and the parameters λ and ρ. The default parameters are: the number of diffusion provenances $k = 5$, the number of detectors $d = 20\% * N$, the monitoring time window $T = 5$ minutes. The selection of provenances and detectors are random. The propagation time delay along each edge follows a Gaussian $\theta_i \sim N(1, 0.01)$. Figure 6 demonstrates the model performance *w.r.t.* different parameters on the synthetic and real-world datasets. We replicated the algorithms five times and used the mean as the evaluation indicator.

The number of diffusion provenances k. From Figure 6(A), we can see that the error rate (hops) decreases *w.r.t.* the number of diffusion provenances. We can conclude that the more diffusion provenances, the higher probability to be found out at least one provenance.

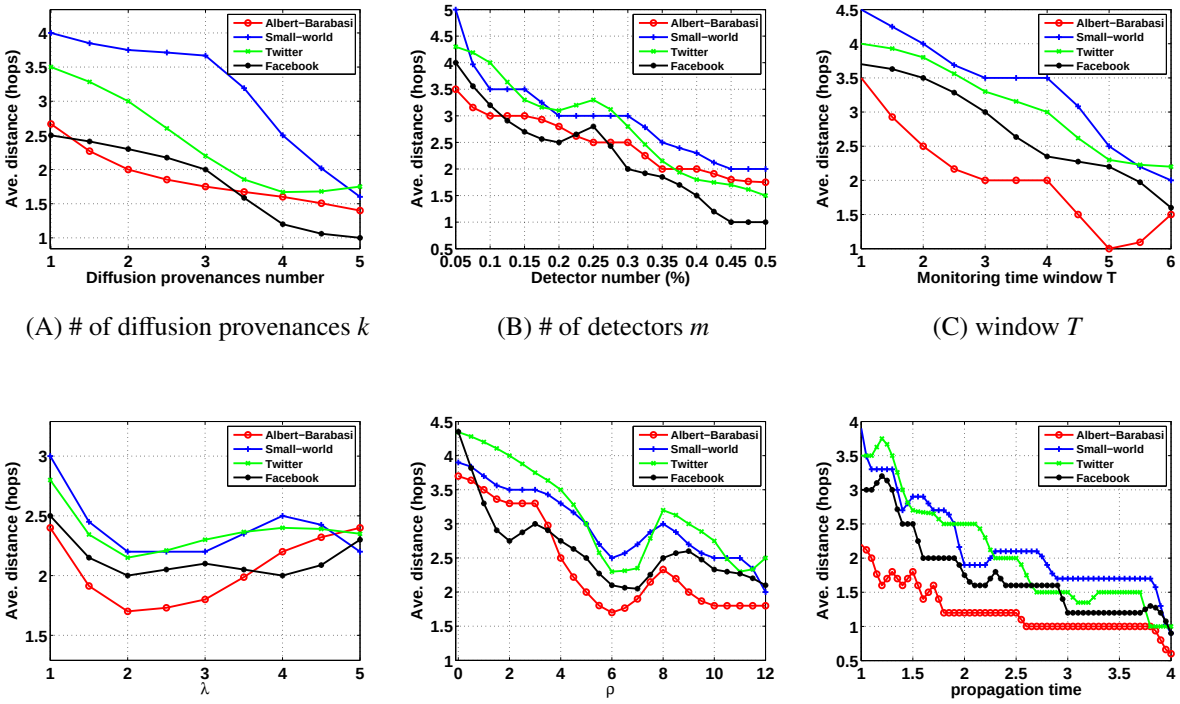
The number of detectors m. Figure 6(B) shows the average distance (hops) *w.r.t* the number of detectors. The result demonstrates that the accuracy improves *w.r.t* the number of detectors.

The monitoring time window T. The monitoring time window indicates the detection time span. Figure 6(C) shows that the error hops drop with the time window.

The parameter λ. The parameter λ controls the preference between the activated nodes (convex part) and the inactivated nodes (concave part). As shown in Figure 6(D), the parameter λ should weigh the convex part and the concave part. If λ is selected too large, it overfits the inactivated nodes; otherwise, it overfits the activated nodes.

The parameter ρ. $\rho > 0$ is the regularization parameter, and restricts the size of x. If ρ is too large, the algorithm surfers from time cost, especially on large scale networks. From Figure 6(E), we observe the minimal hops when ρ is 6.

Continuous detection. We test the proposed online algorithm on the synthetic and real-world date sets. Also, we empirically study the learning rate parameter η_t in the sub-gradient algorithm. Figure 6(F) shows the performance of the algorithm *OSS*. The hops reduce along with the propagation time because more activated detectors lead to better performance. At the end of the time window, the hops shrink as the inactivated detectors are obtained.



(A) # of diffusion provenances k (B) # of detectors m (C) window T

(D) parameter λ (E) parameter ρ (F) online detection

Figure 6: Parameter study on synthetic and real-world data sets by using the proposed regression learning model and Online Stochastic Subgradient algorithm. The number of hops (error rate) *w.r.t*: (A) the diffusion provenances number k; (B) the detector number m; (C) the monitoring time window T; (D) the parameter λ; (E) the parameter ρ. (F) the online detection. The average distance on synthetic/real-world data sets *w.r.t* propagation time.

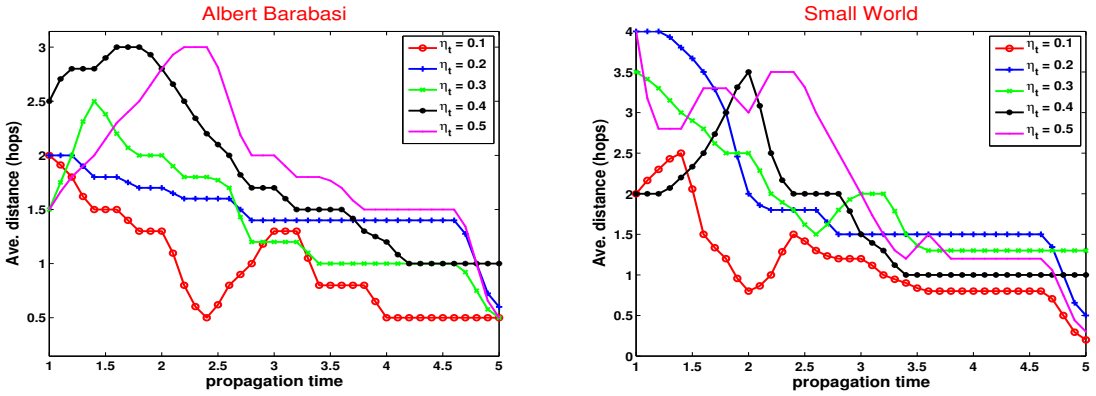


Figure 7 demonstrates the *OSS* algorithm with learning rates η_t. The algorithm reaches the least average distance when η_t is 0.1.

Compare with benchmark methods. We compare the proposed non-convex sparse regression model (NSR

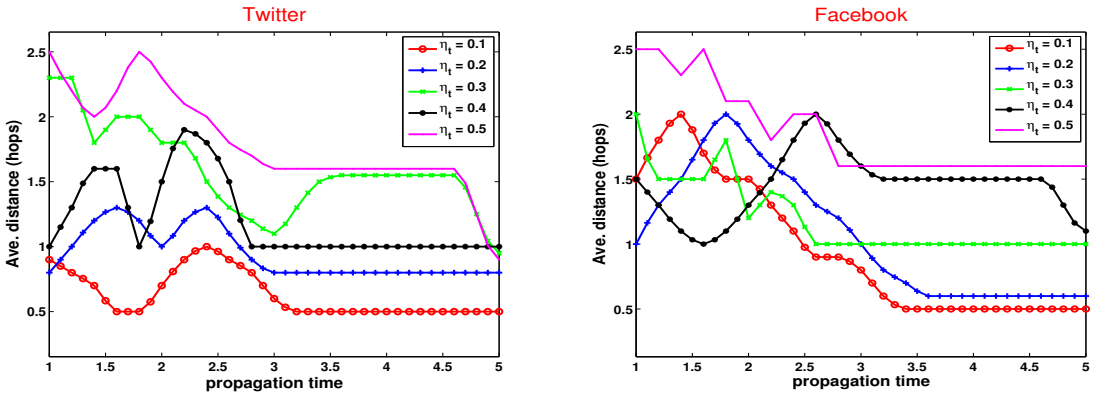


Figure 7: Parameter η_t in the proposed Online Stochastic Sub-gradient *OSS* algorithm.

Table 2: Comparisons on the four data sets.

		$K = 5$				$K = 10$			
	Detectors (m)	0.01	0.05	0.1	0.2	0.01	0.05	0.1	0.2
Albert-Barabasi	NSR	**3.47±0.64**	**2.33±0.66**	**2.21±0.50**	**1.47±0.24**	3.40±0.21	**2.19±0.48**	2.08±0.45	**1.52±0.26**
	CVXR	3.48±0.25	2.37±0.31	2.38±0.69	2.17±0.34	3.72±0.51	2.38±0.47	**2.05±0.58**	1.78±0.47
	CAVR	3.82±0.66	2.77±0.68	2.99±0.26	2.10±0.43	3.86±0.44	2.98±0.63	2.93±0.57	2.20±0.39
	Betweenness	3.48±0.34	3.19±0.26	3.07±0.58	3.25±0.20	3.02±0.41	3.82±0.23	3.92±0.44	3.78±0.30
	Bonacich	3.48±0.33	3.30±0.33	3.30±0.25	3.00±0.30	**3.00±0.33**	3.55±0.30	3.00±0.44	3.44±0.23
Small-World	NSR	4.31±0.56	**3.32±0.68**	**2.30±0.28**	**2.02±0.65**	**3.27±0.48**	**3.05±0.53**	**2.28±0.30**	1.86±0.48
	CVXR	4.44±0.48	3.51±0.20	3.35±0.60	2.08±0.30	3.32±0.58	3.14±0.33	2.79±0.55	**1.49±0.20**
	CAVR	4.53±0.41	3.62±0.27	3.47±0.20	2.10±0.68	3.82±0.61	3.17±0.21	2.95±0.47	2.03±0.34
	Betweenness	**3.27±0.38**	3.46±0.48	3.11±0.22	3.09±0.40	3.32±0.60	3.17±0.66	3.26±0.29	3.52±0.25
	Bonacich	3.30±0.50	3.45±0.44	3.00±0.33	3.00±0.33	3.30±0.55	3.10±0.50	3.30±0.44	3.50±0.22
Twitter	NSR	4.33±0.51	**3.67±0.24**	**2.50±0.35**	**2.33±0.47**	**3.67±0.44**	**3.33±0.52**	2.01±0.52	**1.50±0.31**
	CVXR	3.77±0.69	4.25±0.65	3.0±0.65	2.37±0.35	**3.67±0.33**	3.67±0.55	2.43±0.40	1.62±0.67
	CAVR	4.50±0.43	4.39±0.60	3.09±0.29	2.57±0.65	3.77±0.46	3.75±0.49	2.99±0.46	2.67±0.49
	Betweenness	4.40±0.24	3.75±0.27	3.20±0.23	3.73±0.36	3.96±0.20	3.87±0.31	3.56±0.37	3.41±0.63
	Bonacich	**3.66±0.45**	3.66±0.55	3.00±0.33	3.50±0.30	3.66±0.25	3.62±0.40	3.55±0.44	3.33±0.25
Facebook	NSR	3.5±0.31	**3.20±0.21**	3.21±0.25	**2.10±0.33**	**2.80±0.32**	**2.67±0.16**	**1.85±0.51**	**1.33±0.60**
	CVXR	3.67±0.33	3.32±0.51	3.09±0.56	2.66±0.43	3.02±0.48	2.87±0.48	1.92±0.32	1.70±0.66
	CAVR	3.85±0.27	3.67±0.35	3.24± 0.66	2.52±0.59	3.50±0.31	3.06±0.54	2.12± 0.45	2.03±0.40
	Betweenness	**3.37±0.59**	3.21±0.23	3.02±0.44	3.83±0.21	3.63±0.23	3.26±0.32	3.44±0.46	3.19±0.23
	Bonacich	3.50±0.33	3.33±0.25	**3.00±0.44**	3.00±0.30	3.55±0.50	3.55±0.30	3.30±0.44	2.88±0.30

for short) with three benchmark methods: 1) the convex-based sparse regression method (VEXR for short) [28], which uses activated nodes for regression and locating; 2) the concave-based sparse regression method (CAVR for short) [28], which uses inactivated nodes for regression; 3) *Betweenness centrality* [21], which measures a node's centrality in a network. We use betweenness centrality to measure how often a node appears on the shortest path. The betweenness centrality of a node v is given by the function: $g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the total number of the shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v; and 4) *Bonacich centrality* [23], also known as Eigenvector Centrality, measures the influence of a node in a network.

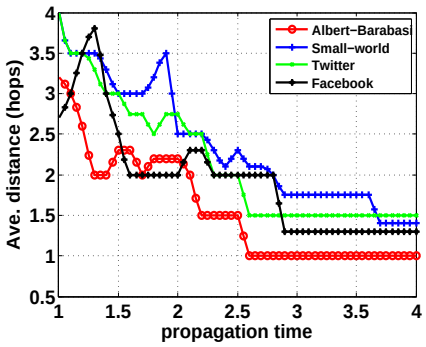


Figure 8: The online detection under the Linear Threshold propagation process.

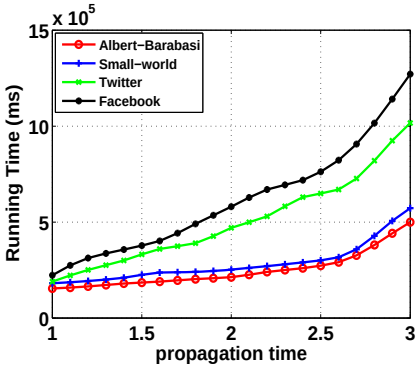


Figure 9: Running time *w.r.t* the propagation time on the synthetic and real-world data sets.

The influence of nodes is based on assigning relative scores to all nodes in the network. The centrality score of node v is defined as: $x_v = \frac{1}{\lambda}\sum_{t\in M(v)} x_t = \frac{1}{\lambda}\sum_{t\in G} \alpha_{v,t} x_t$, where G is network, $A = (\alpha_{v,t})$ be the adjacency matrix, i.e., $\alpha_{v,t} = 1$ if node v is linked to node t, and $\alpha_{v,t} = 0$ otherwise, $M(v)$ is a set of the neighbors of v, and λ is a constant (we set it as 1 in our experiment). We use UCINET 6 for Windows to compute the Bonacich centrality.

We compare the four methods on the two synthetic networks and two real-world networks. The parameters are set as default. We test two settings, the number of diffusion provenances $k = 5$ and $k = 10$ on the four synthetic datasets. For each group, we randomly sample different numbers of detectors m. From the results in Table 2, the reslus can be summarired as follows.

1). the proposed NSR model, by leveraging both activated and inactivated nodes, performs better than both CVXR and CAVR models on most of the data sets, achieving the lowest expected hops, especially when m is large. For example, on the ForestFire data set, NSR achieves the best result of 1.33 hops when $k = 10$ and $m = 0.2 * N$.

2). The NSR model performs better than the heuristic *Betweennes* model when the sample rate m/N is high. This is because the betweenness algorithm aims to find the central nodes in a network, and it cannot make proper use of information from detectors. However, when the sample rate is low, betweenness shows better results than the other three.

3). When the number of detectors increases, the error rates of NSR, CVXR and CAVR reduce significantly, among which NSR reduces the most. The performance of the betweeness model is stably poor.

4). The accuracy of centrality increases with an increased the number of sources, because centrality finds the most influential nodes in a social network. However, sometimes the source may not be among the most influential nodes. For example, malicious information may be misforwarded by one user and then spread

by the other influential users. That is, centrality approaches tend to discover key infrastructure nodes or super-spreaders, but do not necessarily pinpoint the diffusions source. The accuracy increase is because more provenances have a higher probability that the influential nodes are provenances.

5). The NSR model outperforms the other benchmarks in most cases, because centrality finds the most influential nodes in a social network. However, sometimes the source may not be among the most influential nodes. For example, malicious information may be misforwarded by one user from and then spread by other influential users. That is, centrality approaches tend to discover key infrastructure nodes or super-spreaders but do not necessarily pinpoint the diffusions source. By leveraging both activated and inactivated nodes, NSR can achieve better results than VEXR (only uses activated nodes) and CAVR (only uses inactivated nodes).

6). The NSR model outperforms the other benchmarks in most cases, because centrality finds the most influential nodes in a social network. However, sometimes the source may not be among the most influential nodes. For example, malicious information may be misforwarded by one user from and then spread by other influential users. That is, centrality approaches tend to discover key infrastructure nodes or super-spreaders but do not necessarily pinpoint the diffusions source. By leveraging both activated and inactivated nodes, NSR can achieve better results than VEXR (only uses activated nodes) and CAVR (only uses inactivated nodes).

Online algorithm in the different propagation processes. We evaluate our algorithm in the different propagation processes, *i.e.*, Linear Threshold propagation process. A node v has threshold $\theta_v \sim U[0,1]$, and is influenced by each neighbor t according to a weight b_{vt} such that: $\sum_{t\ neighbor\ of\ v} b_{v,t} \leq 1$. A node v becomes active when at least θ_v fraction of its neightbors are active $\sum_{t\ active\ neighbor\ of\ v} b_{v,t} \geq \theta_v$. Compared with Figure 6(F), Figure 8 shows the proposed online algorithm can achieve similar results in different propagation process.

Online algorithm time cost. Figure 9 plots the average running time of the algorithm *OSS* on the synthetic and real-world datasets. The running time raises moderately at the beginning due to the increase of activated nodes (Function 1), and then increases sharply in the end because all the activated and inactivated detectors are used for calculation (Function 2).

7. Conclusions

This paper discusses a solution for discovering the diffusion provenances in social networks in online setting. We proposed a real-time source detection algorithm by placing detectors randomly across a network. Our approach converts the problem to a regression problem, and uses an online stochastic sub-gradient algorithm to solve regression as the information is gathered from detectors in real time. This work focuses on online detection rather than

offline, to meet the practical needs for early warning, real-time awareness and real-time responses to malicious information spreading within an online social network. This work can therefore be used in time-critical security monitoring applications, such as locating false rumors in business areas.

Although our algorithm for detecting the diffusion provenances detection achieves high accuracy and meets the need for a real-time response, some limitations exist that need improvement in the future work: 1) the simulation study used real network data but the propagation process was synthetic, dubbed half-real data, so experiments on real network propagation data need to be undertaken; and 2) the proposed algorithm, based on sub-gradients, is a straightforward solution for online learning problems, however more state-of-the-art online learning algorithms could be applied to this online diffusion provenance problem, such as passive-aggressive (PA), second-order perceptron (SOP) and confidence-weight learning (CW).

This work inspires some interesting directions for future work: 1) the problem could be further extended by using popular stochastic epidemic models such as the SR and SRI models; 2) previous mobile social network mining techniques could be used to harness geographical information to identify a culprits physical location.

References

[1] Security focus report: Spam in today's business world, Tech. rep., TREND MICRO (2011).

[2] B. A. Prakash, J. Vreeken, C. Faloutsos, Spotting culprits in epidemics: How many and which ones?, in: ICDM, 2012, pp. 11–20.

[3] A. Y. Lokhov, M. Mézard, H. Ohta, L. Zdeborová, Inferring the origin of an epidemic with a dynamic message-passing algorithm, Phys. Rev. E 90 (2014) 012801.

[4] W. Luo, W. P. Tay, M. Leng, How to identify an infection source with limited observations, IEEE J. Sel. Topics Signal Process. 8 (4) (2014) 586 – 597.

[5] P. C. Pinto, P. Thiran, M. Vetterli, Locating the source of diffusion in large-scale networks, Phys. Rev. Lett. 109 (2012) 068702.

[6] N. Karamchandani, M. Franceschetti, Rumor source detection under probabilistic sampling, in: ISIT, 2013, pp. 2184–2188.

[7] D. Shah, T. Zaman, Rumors in a network: Who's the culprit?, IEEE Trans. Inf. Theor. 57 (8) (2011) 5163–5181.

[8] K. Zhu, L. Ying, Information source detection in the sir model: A sample-path-based approach, IEEE/ACM Trans. Netw. PP (99) (2015) 1–1.

[9] F. J. Ortega, J. A. Troyano, F. L. Cruz, C. G. Vallejo, F. EnríQuez, Propagation of trust and distrust for the detection of trolls in a social network, Computer Networks 56 (12) (2012) 2884–2895.

[10] P. H. Las-Casas, D. Guedes, J. M. Almeida, A. Ziviani, H. T. Marques-Neto, Spades: Detecting spammers at the source network, Computer Networks 57 (2) (2013) 526–539.

[11] M. Khosroshahy, M. K. M. Ali, D. Qiu, The sic botnet lifecycle model: A step beyond traditional epidemiological models, Computer Networks 57 (2) (2013) 404–421.

[12] C.-D. Wang, J.-H. Lai, P. Yu, Neiwalk: Community discovery in dynamic content-based networks, IEEE Trans. Knowl. Data Eng. 26 (7) (2014) 1734–1748.

[13] M. Gomez-rodriguez, D. B. B. Schlkopf, Uncovering the temporal dynamics of diffusion networks, in: ICML, 2011, pp. 561–568.

[14] N. Du, L. Song, M. Gomez-Rodriguez, H. Zha, Scalable influence estimation in continuous-time diffusion networks., in: NIPS, 2013, pp. 3147–3155.

[15] J. Zhao, J. C. Lui, D. Towsley, X. Guan, Whom to follow: Efficient followee selection for cascading outbreak detection on online social networks, Computer Networks 75 (2014) 544–559.

[16] W. Dong, W. Zhang, C. W. Tan, Rooting out the rumor culprit from suspects, in: ISIT, 2013, pp. 2671–2675.

[17] H. Zhuge, J. Zhang, Topological centrality and its e-science applications, J. Am. Soc. Inf. Sci. Technol. 61 (9) (2010) 1824–1841.

[18] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive-aggressive algorithms, J. Mach. Learn. Res. 7 (2006) 551–585.

[19] J. Langford, L. Li, T. Zhang, Sparse online learning via truncated gradient, J. Mach. Learn. Res. 10 (2009) 777–801.

[20] D. Shah, T. Zaman, Detecting sources of computer viruses in networks: theory and experiment, in: ACM SIGMETRICS Performance Evaluation Review, Vol. 38, 2010, pp. 203–214.

[21] M. Barthelemy, Betweenness centrality in large complex networks, The European Physical Journal B-Condensed Matter and Complex Systems 38 (2) (2004) 163–168.

[22] K. Okamoto, W. Chen, X.-Y. Li, Ranking of closeness centrality for large-scale social networks, in: Frontiers in Algorithmics, 2008, pp. 186–195.

[23] P. Bonacich, Power and centrality: A family of measures, American journal of sociology (1987) 1170–1182.

[24] E. Onur, H. Deliç, C. Ersoy, M. U. Çaglayan, Measurement-based replanning of cell capacities in gsm networks, Computer Networks 39 (6) (2002) 749–767.

[25] P. Zhang, B. J. Gao, P. Liu, Y. Shi, L. Guo, A framework for application-driven classification of data streams, Neurocomputing 92 (2012) 170 – 182.

[26] A.-L. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.

[27] D. J. Watts, S. H. Strogatz, Collective dynamics of'small-world'networks., Nature 393 (6684) (1998) 409–10.

[28] N. E. Aguilera, L. Forzani, P. Morin, On convex regression estimators, arXiv preprint arXiv:1006.2859.