


Article

Tree Sampling for Detection of Information Source in Densely Connected Networks

Taewon Min and Changhee Joo * 

Ulsan National Institute of Science and Technology, Ulsan 44919, Korea; mintw0513@unist.ac.kr

* Correspondence: cjoo@unist.ac.kr; Tel.: +82-52-217-2133

Received: 23 April 2019; Accepted: 22 May 2019; Published: 27 May 2019



Abstract: We investigate the problem of source detection in information spreading throughout a densely-connected network. Previous works have been developed mostly for tree networks or applied the tree-network results to non-tree networks assuming that the infection occurs in the breadth first manner. However, these approaches result in low detection performance in densely-connected networks, since there is a substantial number of nodes that are infected through the non-shortest path. In this work, we take a two-step approach to the source detection problem in densely-connected networks. By introducing the concept of detour nodes, we first sample trees that the infection process likely follows and effectively compare the probability of the sampled trees. Our solution has low complexity of $O(n^2 \log n)$, where n denotes the number of infected nodes, and thus can be applied to large-scale networks. Through extensive simulations including practical networks of the Internet autonomous system and power grid, we evaluate our solution in comparison with two well-known previous schemes and show that it achieves the best performance in densely-connected networks.

Keywords: source detection; estimation; approximation

1. Introduction

During the last decade, mobile Internet services have become very popular. In particular, Social Networking Services (SNS) like Facebook and Twitter have emerged and attracted several billions of users. The new services not only help people build their social relationships by sharing their personal or career interests, but also allow them to be involved in the generation of data and news, changing the media landscape. On the other hand, it has been reported that they can be misused to spread rumors and irresponsible statements or to even infect devices with malware. Due to the properties of the Internet such as anonymity and non-persistent connectivity, it seems to be hard to completely prevent them. To this end, the techniques of detecting the source of the information (i.e., rumors or malware) would be an effective measure to prevent the misuse and to block the information spreading over the networks.

Although there have been several studies on the information source detection in the literature [1–9], most of them have assumed a specific type of network topology, i.e., trees, and cannot be directly applied to a more general network topology that most social networking services have. Their extension to general networks is not straightforward and often suffers from high computational complexity. For example, the Maximum Likelihood Estimator (MLE), named rumor centrality [1], has been developed for the source detection in tree networks. Its extension to general networks with cycles requires consideration of all possible spanning tree networks, which increases the complexity exponentially with the network size and makes the algorithm practically infeasible.

As many practical networks have more complex structures with cycles, the infection process in general networks is of great interest. Furthermore, since the networks may have several billions of users [10], the development of low-complexity solution is vital in the practical use of the detection

algorithm. The huge computational complexity of the MLE makes it hard to be applied to a large-sized network. In this work, we investigated the problem of source detection in general networks, in particular in densely-connected networks (by ‘densely-connected networks’, we mean that the ratio of the number of edges to the number of nodes is greater than 1.5), and develop low-complexity approximations with high detection performance. We consider the well-known epidemic Susceptible-Infected (SI) model to investigate the impact of dense connectivity on the infection process. Our main contributions can be summarized as follows.

- We develop a tree generation algorithm such that the infection process likely follows the generated tree. Note that, in a tree network graph, the infection at each node is independent since it should follow a specific path. In contrast, in a general network graph, there are multiple possible paths of the infection due to the loops or cycles in the underlying topology. To this end, we generate a spanning tree that consists of the edges that likely infect a node, by introducing the concept of *detour* nodes.
- For each candidate node, we approximately compute the probability that the chosen spanning tree is indeed the infection path and compare their likelihood to decide the source. By taking into account the edges that are connected to non-infected nodes, we can provide more accurate estimation on the probability.
- We evaluate our solution in several network graphs, including trees and practical network graphs. We demonstrate that the proposed solution achieves better performance in densely-connected networks than other comparable schemes.

The rest of the paper is organized as follows. After a brief overview of the related work in Section 2, we describe the Susceptible-Infected (SI) epidemic model and introduce two previous measures that have been developed for the source detection in Section 3. Our solution is described in Section 4 and evaluated through simulations in a wide variety of networks in Section 5. We conclude our paper in Section 6.

2. Related Works

The authors of [1–3] have studied the information source detection on a regular tree under the Susceptible-Infected (SI) epidemic model, in which the state of a node is either susceptible or infected. Based on a combinatorial quantity named “rumor centrality”, they have developed the Maximum Likelihood Estimator (MLE) and characterized the detection probability on trees. In particular, it has been shown that the detection probability on trees that grow faster than a line is non-trivial and goes to zero as the size of the tree increases. The authors of [4,5] have considered the detection problem with multiple information sources. In particular, in [4], the authors developed an algorithm with quadratic complexity to estimate the actual number of infection sources and to identify them. It has been shown that in a geometric tree, their estimator identifies the true sources with accuracy as the number of infected nodes increases.

The dynamics of the spreading process have been studied in [6–9]. In [6], the information source detection under the Susceptible-Infected-Recovered (SIR) epidemic model has been considered. In this model, a node can be in one of three states of susceptible, infected, and recovered, and an infected node can be recovered with a certain probability. The authors have introduced a sample path-based estimator to find the node that minimizes the maximum distance to an infected node on the tree. In order to capture the epidemic propagation property of the antidote information, the SIR model has been extended to the Susceptible-Infected-Cured (SIC) epidemic model [7], where the antidote makes susceptible and infected nodes cured. The authors have shown that the half-life time of virus over two simple networks of clique networks and star networks is $O(\frac{\log n}{n})$ and $O(\log n)$, respectively, where n is the number of infected nodes. Besides epidemic models, a game-based model has been studied in [9], where each node can adopt the information to maximize its payoff. The work aims

to find good seeds (i.e., information sources) that can maximize the diffusion speed and developed interesting seeding schemes for different network structures.

Most previous works focused on tree networks. In this work, we consider general networks with cycles. As a first step, we consider a simpler model of the SI epidemic model as described in the next section.

3. Epidemic Model

We considered an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of (undirected) edges. We assumed that any pair of two nodes can be connected by at most one edge. Its extension to multiple edges between a pair of nodes would be straightforward. Each node $v \in V$ has two possible states: susceptible (S) and infected (I). Let V_I denote the set of infected nodes, and let E_I denote the set of edges induced by V_I . Initially, all the nodes are susceptible except source node v_s (i.e., initially $V_I = \{v_s\}$). An infected node can infect its neighboring nodes connected by an edge. The edge that is used to infect a node is denoted by *infection edge*. Let us define the *infection path* of a node as the sequence of infection edges from the source to the node.

Consider two susceptible nodes u and v connected through edge $(u, v) \in E$. Suppose that node u is infected first. It remains infected and will try to infect node v , where the infection time takes a random amount of time τ_{uv} , which is independently drawn from an exponential distribution with an identical mean $1/\lambda$. It is based on the assumption that the time for rumor and virus spreading has the memoryless property [1,2]. The procedure repeats between every infected and susceptible node pair. Figure 2b can be considered as a snapshot of infection (or information) spread, where the circles are a node and the lines an edge. Node v is the source; the gray circles indicate an infected node; and the thick solid or dotted lines indicate the infection edges.

We introduce two previous measures that have been developed to detect the source: *Closeness Centrality (CC)* [11] and *Infection Eccentricity (IE)* [6]. CC is a measure of centrality for node v based on the sum of the shortest-path length over all the nodes in the graph. Specifically, the closeness centrality $c(v)$ of node v is defined as:

$$c(v) := \frac{1}{\sum_{u \in V_I} d(v, u)}, \quad (1)$$

where $d(v, u)$ is the length of the shortest path between v and u . IE is an alternative to measure the centrality of a node in tree networks [6]. Letting $e(v)$ denote the infection eccentricity of node v , it is defined as:

$$e(v) := \max_{u \in V_I} d(v, u). \quad (2)$$

One may find the node that minimizes the closeness centrality (or the infection eccentricity). It has been shown that such schemes perform well in tree networks, since the infection of a node occurs only through its specific neighbor node that is closest to the source (i.e., its parent). Furthermore, it is known that the node that minimizes the closeness centrality is the MLE in regular trees [1]. However, when there are multiple paths between the source and a node, (i.e., there are loops in the underlying topology), the two measures may result in poor performance, as we will see in Section 5. To this end, it is important to develop low-complexity schemes for accurate source detection in densely-connected networks.

4. Sample Tree-Based Estimator

We introduced a new estimator that achieves good performance in general graphs and developed a couple of low-complexity algorithms that can successfully approximate the new estimator.

Given the set V_I of infected nodes, let G_I denote the induced graph by V_I from G . Let Γ_I denote the family of all spanning trees that can be obtained from G_I . We define the Sample-Tree-based Estimator (STE) as:

$$v^* := \operatorname{argmax}_{v \in V} P(T_v^* | v), \quad (3)$$

$$\text{where } T_v^* := \operatorname{argmax}_{T \in \Gamma_I} P(T | v), \quad (4)$$

where $P(T|v)$ denotes the probability that the infections occur through tree T starting from node v . Note that the estimator allows a two-step procedure as follows.

1. For each node (v), we solve (4) and obtain the most likely paths (T_v^*) that the infection process has likely followed in G_I .
2. Compute probabilities $P(T_v^* | v)$ that the infections indeed occur through tree T_v^* , and select node v^* with the highest probability as the source.

Note that similar two-step approaches were taken by previous works [1–4,12,13]. Since computing $P(T|v)$ for each tree is computationally expensive, they found a tree in the Breadth First Search (BFS) manner (denoted by the BFS tree) and assumed the BFS tree as the most likely infection paths T_v^* . Note that on the BFS tree, the infection path between the source candidate v and node u corresponds to one of the shortest paths between them. However, it is frequently observed that the infection path is different from the shortest path, which causes substantial estimation errors. Furthermore, even if we have obtained the most likely infection paths T_v^* for each v , we still have to obtain the MLE to compare $P(T_v^* | v)$, which incurs high computational complexity with the network size and thus is hardly used in practice [14].

We tackle each of the two subproblems, aiming to make good approximations with low complexity. We first take into account the probability that an infection path does not follow the shortest path and develop a novel algorithm that constructs a non-BFS tree that the infection paths more likely follow than the BFS tree. Then, we develop a heuristic algorithm that approximates probability $P(T_v^* | v)$ and verify its performance through simulations.

4.1. Generating Trees

We first considered the problem (4). Due to the large size of the search space Γ_I , it is computationally infeasible to take into consideration all the spanning trees [15]. For example, the complete graph with n nodes has n^{n-2} spanning trees. However, since many spanning trees are highly unlikely to be an infection path (e.g., a tree where all nodes are in a line), we may select some candidate trees for computational efficiency.

Most of the previous works have implicitly or explicitly assumed the BFS tree, because a node closer to the source is likely infected earlier. However, the BFS tree is an extreme case, and the infection process is less likely to follow in a general graph. If the network graph is densely connected with cycles, there is a substantial probability for a node to be infected through a non-shortest path.

In order to sample the most likely spanning trees from Γ_I , it is important to understand the infection process on a general graph. In particular, we paid attention to the nodes that were infected through a non-shortest path. We denote such a node by *detour node* and generated a tree by emulating the infection process using the detour nodes.

For an infected node v , let D_v denote its distance (hops) from the source through its infection path, and let S_v denote the length of the shortest path from the source in G_I . Let $\Delta_v := D_v - S_v \geq 0$, which indicates the level of discrepancy between the infection path and the shortest path in the number of hops.

Definition 1. Suppose that node v directly infects node u . Node u is a *detour node* if $\Delta_u - \Delta_v > 0$, i.e., the level of discrepancy between the infection path and the shortest path increases at node u .

We consider the probability P_u^d that an infected node u is a detour node. Let \mathcal{N}_u denote the set of neighboring nodes of u , and let \mathcal{P}_u denote the set of nodes on the shortest path(s) between u and v in

G_I . We can classify the neighbors into two categories: $\mathcal{N}_u := \mathcal{N}_u \cap \mathcal{P}_u$ and $\mathcal{N}_u^c := \mathcal{N}_u \setminus \mathcal{N}_u$. Then, P_u^d equals the probability that a node in \mathcal{N}_u^c infects node u earlier than any nodes in \mathcal{N}_u .

Consider the simplest scenario where node u has $n (= 1)$ shortest paths of length one to node v (i.e., directly connected) and has m disjoint non-shortest paths of length two. In this case, node u is a detour node if it is infected through a two-hop path first. Let X denote the time that a node in \mathcal{N}_u^c infects u . Since the infection time follows an exponential distribution, the probability density $f_X(t)$ that the infection of u by the node takes time t follows the gamma distribution, i.e., $f_X(t) = m^2 t \cdot e^{-mt}$. Similarly, letting Y denote the time that it takes for a node in \mathcal{N}_u to infect u , its probability density can be written as $f_Y(t) = n \cdot e^{-nt}$. Let $P_{u,1,2}^d$ denote the probability of being a detour node by taking a two-hop path instead of a one-hop path. We have:

$$P_u^d = P_{u,1,2}^d = \iint_{X < Y} f_X(x) f_Y(y) dx dy = \frac{m^2}{(m+n)^2}. \tag{5}$$

Although it is an interesting result, it is limited to the case that node u has n one-hop paths and m two-hop paths to the source. Its extension to more general scenarios is not easy, since we may have to consider all the possible infection paths.

However, it is possible to make use of Equation (5) to approximate the detour probability in general networks. When a candidate source node is given, we can emulate the infection process through Equation (5) and sample a spanning tree, which can be considered as infection paths from the candidate source with high probability. We describe our tree generation algorithm, named *Sample-Tree Generation (STG)*.

1. Given the set V_I of infected nodes and candidate source nodes v , initialize $I = \{v\}$, which will be used as the up-to-date set of the infected nodes in our emulation process (Line 2 in Algorithm 1).
2. Randomly pick an edge from E_{temp} that is the set of edges connecting between nodes in I and nodes out of I , i.e., $E_{temp} = \{(u, w) \in E_I | u \notin I, w \in I\}$. Then, we add the edge in the tree, update $I \leftarrow I \cup \{u\}$ (Lines 4–6), and remove the edges from E_{temp} (Line 14). See also Figure 1, where edge (u, w) is chosen from E_{temp} .
3. Among some neighbors in $\mathcal{N}_u \setminus I$ that have an edge to I (Line 7, x and x' in Figure 1), we randomly pick a node (e.g., node x) and set it as a detour node with probability $P_{x,1,2}^d$. The probability is calculated as in Equation (5), where m is obtained from the number of paths between x and any nodes in I , and $n = 1$, respectively. If node x is set as a detour node, edge (u, x) is added to the tree (Line 9).
4. Repeat the procedure until E_{temp} is empty. If it becomes empty, go to Step 2, and reset E_{temp} using the current I .
5. If there is no remaining infected node, it returns T as \hat{T}_v^* , which will be used in the sequel as the most likely infection paths instead of T_v^* .

Note that the above description is an outline, and the detailed procedure can be found in Algorithm 1.

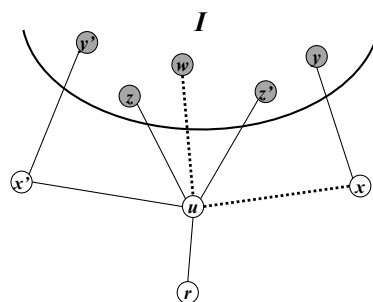


Figure 1. An iteration of Algorithm 1. After edge (u, w) is included in tree T , the D-GENfunction randomly picks (x, u) from $\{(x, u), (x', u)\}$ and adds it in tree T with probability P_x^d .

Algorithm 1 Sample-Tree Generation (STG) algorithm.

```

1: function STG( $v, V_I, E_I$ )
2:    $I \leftarrow \{v\}, T \leftarrow \emptyset$ 
3:   while  $|T| < |V_I| - 1$  do
4:      $E_{temp} \leftarrow \{(u, w) \in E_I | u \notin I, w \in I\}$ 
5:     while  $E_{temp} \neq \emptyset$  do
6:        $u \leftarrow \text{GEN}(T, I, E_{temp})$ 
7:        $V' = \{x \in \mathcal{N}_u | \exists y \in \mathcal{N}_x \cap I \text{ and } y \neq u\}$ 
8:        $E'_{temp} \leftarrow \{(u, x) \in E_I | x \in V'\}$ 
9:        $\text{D-GEN}(u, T, I, E'_{temp})$ 
10:    return  $T$  as  $T_v^*$ 

    /* Add a node to the tree, and update  $E_{temp}$  */
11: function GEN( $T, I, E_{temp}$ )
12:   randomly pick  $(u, w) \in E_{temp}$ 
13:    $I \leftarrow I \cup \{u\}, T \leftarrow T \cup \{(u, w)\}$ 
14:    $E_{temp} \leftarrow E_{temp} \setminus \{(u, z) \in E_{temp} | z \in I\}$ 
15:   return  $u$ 

    /* Add a detour node to the tree */
16: function D-GEN( $u, T, I, E'_{temp}$ )
17:   randomly pick  $(u, x) \in E'_{temp}$ 
18:   set  $x$  as a detour node with prob.  $P_{x,1,2}^d$  as in (5)
19:   if  $x$  is a detour node then
20:      $I \leftarrow I \cup \{x\}, T \leftarrow T \cup \{(u, x)\}$ 

```

Figure 2 shows an example tree construction under our STG algorithm. Suppose that the network graph and the set of infected nodes are given as shown in Figure 2a, where the infected nodes are marked in gray and the edges are shown as a line. Starting from sets $I = \{v\}$ and $T = \emptyset$, we randomly pick an edge from E_{temp} , e.g., (v, c) , and add the edge to T and node c to I . After updating I and removing (v, c) from E_{temp} , we check whether there exists a neighbor of node c that is not included in I and has an edge to I except with node c and select one of them at random. In this case, since we have only one such node (node g), we select node g . Then, we set node g as a detour node with probability $P_{g,1,2}^d$, where $P_{g,1,2}^d = \frac{2^2}{(2+1)^2}$ from Equation (5), since node g has one direct path and two two-hop paths to node v . If node g is not set as a detour node, then we go back and randomly select another edge from E_{temp} and repeat the procedure. If node g is set as a detour node, we add edge (c, g) to T and node g to I , then go back and randomly select another edge from E_{temp} , and repeat the procedure. If E_{temp} becomes empty, we reconstruct it from the current set I . Figure 2b illustrates a spanning tree generated by STG, where thick edges denote the constructed tree, and the number associated with each thick edge denotes the order added in I . The dotted line denotes the edge added as a detour.

When the number of infected nodes is n , the complexity of STG is of order $O(n)$ since we only explore $n - 1$ edges to generate the tree.

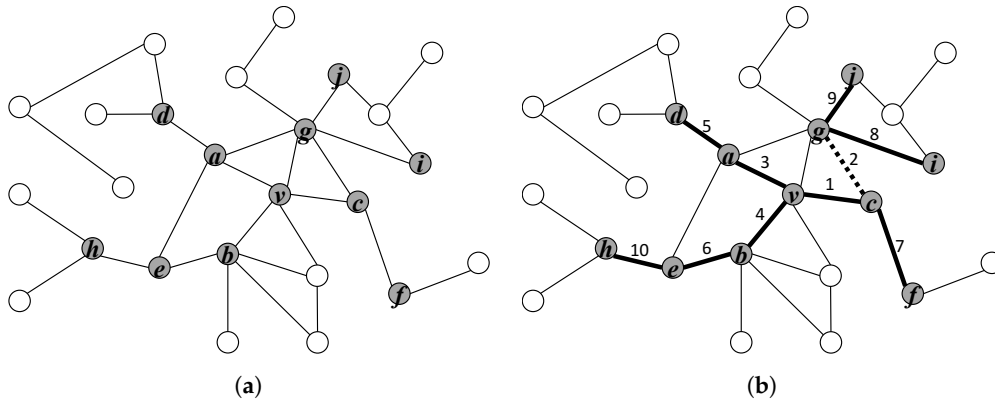


Figure 2. Tree generation under Sample-Tree Generation (STG). (a) An example network graph. Gray circles denote the set of infected nodes V_I . (b) A tree generated by STG. Given v as a root, a tree marked by thick edges is generated under STG. Each number on the edges denotes the order that the edge is added. Due to the detour at (c, g) , it is a non-Breadth First Search (BFS) tree.

4.2. Tree Evaluation

In this section, we consider the conditional probability $P(\hat{T}_v^*|v)$ that, given a candidate source node v , the tree \hat{T}_v^* generated by STG is indeed the infection paths. Due to computational complexity, instead of directly estimating $P(\hat{T}_v^*|v)$ for each $v \in V_I$, we compared their approximated values and found the node with the highest one.

Note that a similar approach was taken in [1], where the proposed scheme computed the number of permutations (of infected nodes) that can lead to the BFS tree if the infections occurred in the order of the permutation. The node with the highest number of such permutations was set as the source. A main drawback of this approach, however, is that it ignores the edges out of the tree and thus causes unavoidable errors in comparing $P(\hat{T}_v^*|v)$. It can be shown by a simple example. Suppose that there are five infected nodes a, b, c, d, e that are linearly connected. Node a has 10 non-infected neighbors, and other infected nodes have zero non-infected neighbors. Then, the scheme of [1] concludes that nodes a and e are equally likely a source node. However, in reality, node e is more likely the source than node a due to the fact that none of a 's neighbors are infected.

Incorporating our intuition, we developed a new evaluation scheme, the *Growth-Tree Estimation (GTE)* algorithm, that approximately compares the probabilities by taking into account all the edges connected to V_I . First, we assumed that the infections occurred stage-by-stage such that all the nodes of depth k in \hat{T}_v^* were infected at stage k . Let V_k^* denote the set of nodes of depth k in \hat{T}_v^* with $V_0^* = \{v\}$, and let $|\cdot|$ denote the cardinality of the set. Let M_k^* denote the set of edges from a node in $\cup_{i=1}^k V_i^*$ to a node out of $\cup_{i=1}^k V_i^*$, i.e., it equals the set of edges between infected nodes and non-infected nodes at the beginning of stage k . Let t_k denote the time for stage k to end for $k \geq 1$. Then, the probability $P_k^*(\hat{T}_v^*)$ that the infections occur through \hat{T}_v^* at stage k can be written as:

$$P_k^*(\hat{T}_v^*) = (1 - e^{-t_k})^{|V_k^*|} \cdot (e^{-t_k})^{|M_{k-1}^*| - |V_k^*|},$$

since the nodes in V_k^* should be infected earlier than the other nodes (non-infected at the beginning of stage k). Then, letting $h(\hat{T}_v^*)$ denote the height of the tree, we can approximate:

$$\begin{aligned} P(\hat{T}_v^*|v) &\approx \prod_{k=1}^{h(\hat{T}_v^*)} P_k^*(\hat{T}_v^*) \\ &= \prod_{k=1}^{h(\hat{T}_v^*)} (e^{t_k} - 1)^{|V_k^*|} \cdot e^{-t_k |M_{k-1}^*|}. \end{aligned}$$

We make another important assumption that $t_k = t$ for all k . Precisely, it is not true, since the time to infect all x nodes independently increases with x . However, if we pay attention to their average

behavior, the median times to infect $|V_k^*|$ nodes are equal across stages. Under the assumption, we have:

$$P(\hat{T}_v^*|v) \approx C_1 \cdot e^{-t \sum_{k=1}^{h(\hat{T}_v^*)} |M_{k-1}^*|}, \tag{6}$$

where $C_1 = (e^t - 1)^{|V_I^*|-1}$ since $\cup_{k=1}^{h(\hat{T}_v^*)} V_k^* = V_I^* \setminus \{v\}$ and V_k^* 's are mutually exclusive.

The importance of Equation (6) is that it allows us to greatly simplify the comparison procedure of $P(\hat{T}_v^*|v)$ for all possible v . Now, we can find the node with the largest $P(\hat{T}_v^*|v)$ by summing the number of edges between infected nodes and non-infected nodes at each stage and report it as the source, i.e.,

$$\hat{v}^* = \operatorname{argmin}_{v \in V_I} \sum_{k=1}^{h(\hat{T}_v^*)} |M_{k-1}^*|, \tag{7}$$

which can be computed quickly. Detailed computation can be found in Algorithm 2, which is denoted as the *Growth-Tree Estimation (GTE)* algorithm. When the number of infected nodes is n , GTE has $O(n \log n)$ complexity, which is much lower than the algorithms that find the MLE on a tree.

Algorithm 2 Growth-Tree Estimation (GTE) algorithm.

```

1: function GTE( $v, T_v^*, E, V_I$ )
2:    $V_0, S_0 \leftarrow \{v\}, N \leftarrow 0, k \leftarrow 0$ 
3:   while  $|S_k| < |V_I|$  do
4:      $M_k^* \leftarrow \{(u, w) \in E | u \notin S_k, w \in S_k\}$ 
5:      $N \leftarrow N + |M_k^*|$ 
6:      $V_{k+1}^* \leftarrow \{u | (u, w) \in T_v^*, u \notin S_k, w \in V_k^*\}$ 
7:      $S_{k+1} \leftarrow S_k \cup V_{k+1}^*$ 
8:      $k \leftarrow k + 1$ 
9:   return  $N$ 

```

Combining the STG algorithm and the GTE algorithm, we developed a new class of source detection schemes denoted as the Sample-Tree-based Estimator (STE). From the complexity of STG and GTE, STE has the complexity of $O(n^2 \log n)$. Although it may be possible to further optimize STG and GTE, we used Algorithms 1 and 2 to show the effectiveness of STE.

5. Simulations

We evaluated the performance of our proposed STE in a wide variety of networks including tree networks. We compared its performance with other heuristic estimators that use Closeness Centrality (CC) [1] and Infection Eccentricity (IE) [6]. Recall that CC detects the source by finding the node that minimizes the distance sum Equation (1), and IE works by finding the node that minimizes the maximum distance Equation (2). We note that the complexity of CC is at least $O(n^2)$, and that of IE can be up to $O(n^3)$.

We first consider tree networks. Although our scheme is aimed for general densely-connected networks with cycles, the performance comparison of the schemes in tree networks and in general networks will clarify differences between CC, IE, and STE. Note that tree networks are a subset of general networks, and thus, an algorithm that performs well in general networks is expected to achieve reasonably good performance in tree networks. We used both regular tree networks, where all nodes had an identical degree of six, and irregular tree networks, where each node had a degree uniformly chosen in [2,10]. For each topology, we ran 1000 simulations, varying the number of infected nodes. The set of infected nodes were constructed by emulating the infection process from node v . To elaborate, given a network topology, we selected the initial source v . For each non-infected neighbor, independently set the infection time following the exponential distribution, and choose the next infected node with the smallest value. We repeated the procedure until we had the intended number

of infected nodes. Once the set of infected nodes was determined, it was given to the source detection schemes without the knowledge of actual source v . In each case, we generated sufficiently large tree networks such that no infected node was a leaf of the tree. We measured the detection probability that the estimated source equaled the actual source, and their distance if they were not equal.

Figure 3 illustrates the performance results of STE, CC, and IE, respectively, in regular trees. The results showed that, in both the detection probability and the average distance (to the actual source), CC showed the best performance, followed by STE and IE. The results for general trees, shown in Figure 4, were similar in that the achieved performance was in the order of CC, STE, and IE. However, due to the variation in node degree, the detection probability decreased, and the average distance slightly increased for each estimator.

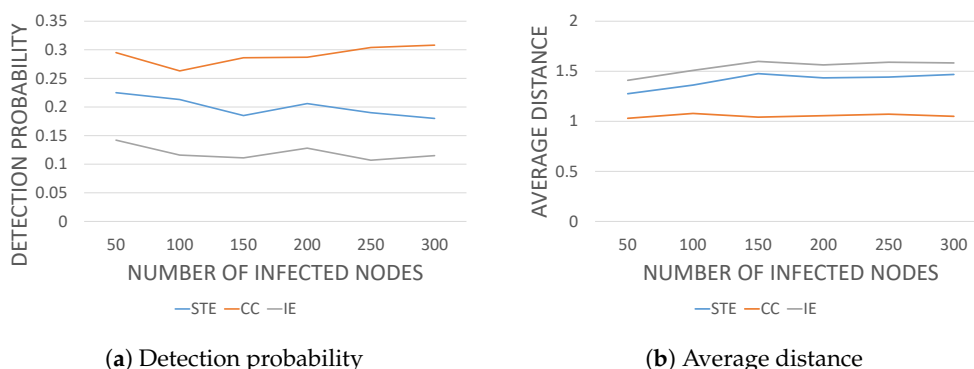


Figure 3. Performance of Sample-Tree-based Estimator (STE), Closeness Centrality (CC), and Infection Eccentricity (IE) in regular trees.

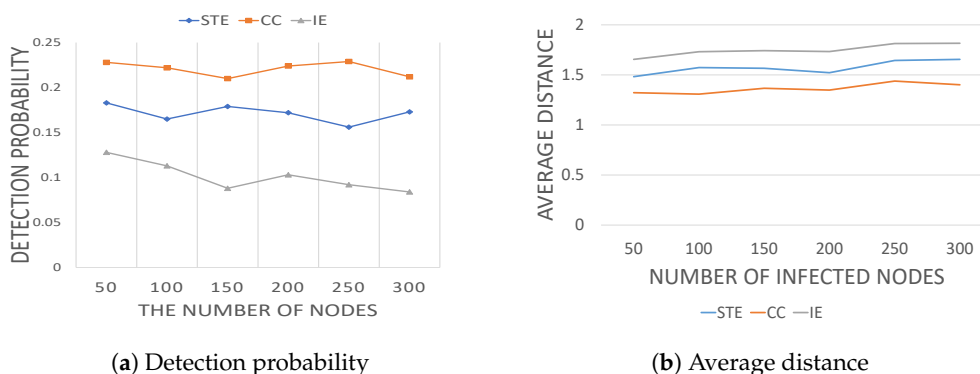


Figure 4. Performance of STE, CC, and IE in general trees.

We now consider more general graphs with cycles, in particular densely-connected network graphs. One is denoted by p -edge networks and can be generated by adding edges to a tree network with probability p . The other is denoted by r -disk networks and can be generated based on the Euclidean distance between nodes. We investigated the performance of the three schemes in the two classes of network graphs. For CC and IE, we calculated the measures over the BFS tree rooted at each candidate node. Since we fixed the (BFS) tree for each infected node, the complexity and the execution time of the three algorithms of CC, IE, and STE was similar. Furthermore, note that, in general non-tree networks, as the network size increases, the detection probability naturally decreases due to the nature of randomness. In our simulations with general graphs, we omitted the detection probability since all the algorithms achieved very small values and used the average distance to evaluate the performance.

We generated a p -edge network as follows. We first generated an irregular tree network of 10,000 nodes, where the degree of each node was uniformly chosen in [2,6] as in the previous simulation. The actual source node (root) is denoted by node v . Then, for each pair of nodes whose $depth$ distance was no greater than 2, we added an edge to the tree with probability p . Changing p from 0.01%,

we varied the density of connectivity and observed its impact on the performance. The network had about 12,000 edges when $p = 0.01\%$, and as we increased p by 0.01%, it had about 3000 additional edges. From our definition, we had densely-connected networks for $p \geq 0.02$ where the ratio of the number of edges to the number of nodes was greater than 1.5. For each case, we marked a total of 300 nodes as infected by emulating the infection process from node v and evaluated the performance of CC, IE, and STE. For each topology, we repeated simulations 200 times and present their average.

Figure 5 shows the performance results for p -edge networks. As we increased p , the network connectivity became denser. We can observe that, in contrast to the previous results with trees, where CC showed the best performance, when we added cycles in trees, the detection performance of CC and IE was similar, and STE slightly outperformed them, as shown in Figure 5a. The detailed distribution of distance to the actual source (when $p = 0.05\%$) is shown in Figure 5b, and it clearly shows that STE achieved better detection performance than CC and IE.

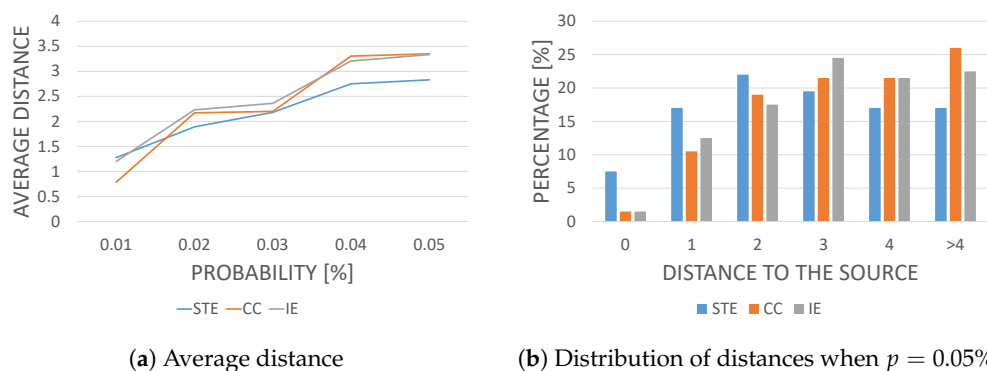


Figure 5. Performance of STE, CC, and IE in p -edge networks.

The performance differences enlarged for the class of r -disk networks. We fixed one node at the center of a circular area with radius one and set it as the source. Then, we added 10,000 nodes uniformly in the area, and made an edge between the nodes whose Euclidean distance was no greater than r . It is also called a unit disk graph [16]. We changed the communication range r starting from 0.0235, which made all 10,000 nodes connected with a total of 50,000 edges, and generated denser networks by increasing the range to its multiples. As r increased, the network graph became denser. In our settings, we observed about 40,000 additional edges per 0.2 increase of r . Again for each r , we emulated the infection process from the source, marked 300 infected nodes, and evaluated the performance of CC, IE, and STE. We repeated the simulations 200 times and present their average.

Figure 6 shows the performance of the schemes in r -disk networks. With the increase of r , the network became denser, and all three schemes achieved better detection performance. However, as shown in Figure 6a, STE detected the source better than CC and IE. Furthermore, the distance distribution when $r = 0.1175$ is shown in Figure 6b and demonstrates that only STE can provide a non-zero probability of exact detection in our settings.

Finally, we are interested in the detection performance in practice. We considered two practical network topologies of the *Internet Autonomous System (IAS) network* [17] and the *power grid network* [18]. The former, whose topology is shown in Figure 7a, has more dense connectivity between the nodes than the latter, whose topology is depicted in Figure 8a.

For the IAS network that consists of 6474 nodes and 13,895 edges, we chose an infection source and emulated the infection process. We conducted 200 simulations, varying the number of infected nodes. In the IAS network, STE clearly outperformed CC and IE, as shown in Figure 7b.

For the power grid network that consists of 4941 nodes and 6594 edges, we repeated the simulations and obtained similar results, as shown in Figure 8b, where STE outperformed CC and IE. However, unlike in the IAS network, their performance difference diminished as the number of infected nodes increased. We conjectured that this was because the power grid network was more

sparsely connected than the IAS network. This is consistent with our results in p -edge networks and r -disk networks, where r -disk networks were more densely-connected than p -edge networks. The performance difference between CC, IE, and STE was outstanding in r -disk networks compared to in p -edge networks.

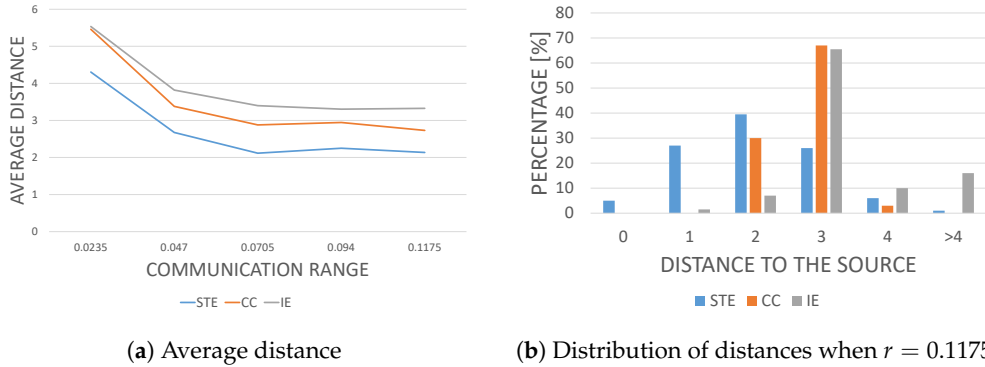


Figure 6. Performance of STE, CC, and IE in r -disk networks.

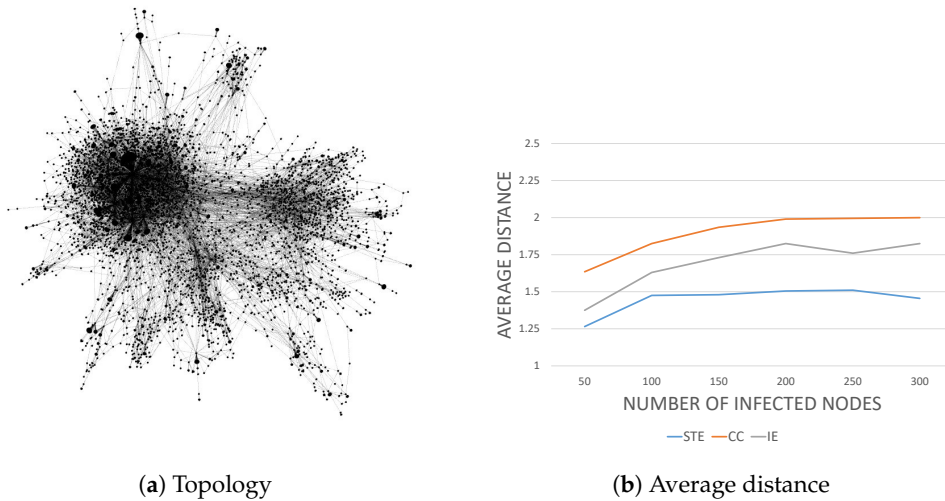


Figure 7. Detection performance in the Internet Autonomous System (IAS) network.

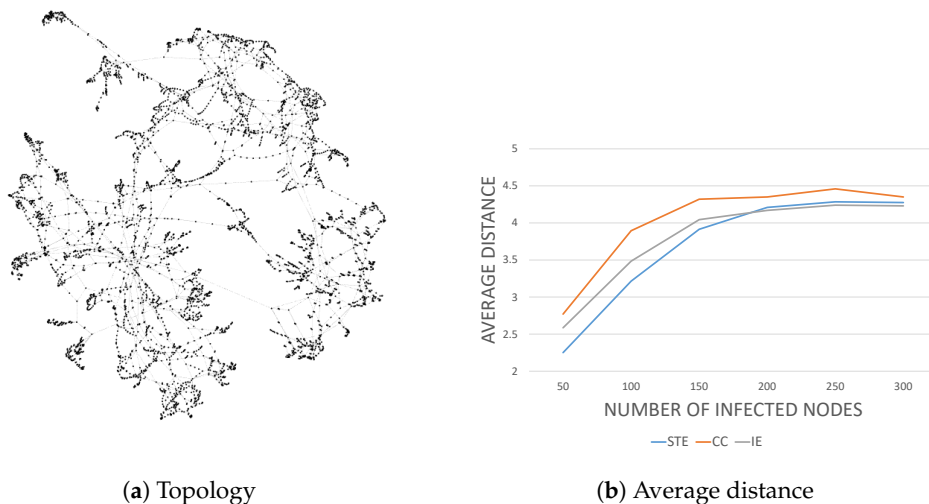


Figure 8. Detection performance in the power grid network.

6. Conclusions

In this work, we have addressed the problem of practical source detection under the susceptible-infected model. We took a two-step approach. For each candidate source, we first found a tree through which the nodes were likely infected. Then, we provided an approximate measure to compare the probability that the infection process followed the tree given each candidate source. In our scheme, we introduced the concept of a detour node to better emulate the infection process and developed a good approximation to compare the probability of infection paths. The complexity of the proposed scheme was $O(n^2 \log n)$, which is much lower than the previous detection schemes. We show through simulations that, in densely-connected networks, our sample-tree-based estimator outperformed the previous methods that were developed for tree networks. We also showed that the proposed scheme achieved good performance in the practical settings of the IAS network and the power grid network.

Finally, we remark that our investigation mainly focused on the impact of dense connectivity in the local neighborhood on the information diffusion. However, there is a number of factors that characterize the diffusion process in general networks, including other topological properties and non-uniform infection probability over edges. For example, it is of great interest to understand the infection process in social networks.

Author Contributions: Conceptualization, T.M. and C.J.; methodology, T.M. and C.J.; software, T.M.; validation, T.M. and C.J.; formal analysis, T.M.; investigation, T.M. and C.J.; writing—original draft preparation, T.M. and C.J.; writing—review and editing, C.J.; visualization, T.M.; supervision, C.J.

Funding: This work was supported by the research fund of the Signal Intelligence Research Center supervised by the Defense Acquisition Program Administration and Agency for Defense Development of Korea.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shah, D.; Zaman, T. Detecting sources of computer viruses in networks: Theory and Experiment. *SIGMETRICS Perform. Eval. Rev.* **2010**, *38*, 203–214. [[CrossRef](#)]
2. Shah, D.; Zaman, T. Rumors in a network: Who's the culprit? *IEEE Trans. Inf. Theory* **2011**, *57*, 5163–5181. [[CrossRef](#)]
3. Shah, D.; Zaman, T. Rumor centrality: A universal source detector. In Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, London, UK, 11–15 June 2012; pp. 199–210.
4. Luo, W.; Tay, W.P.; Leng, M. Identifying infection sources and regions in large networks. *IEEE Trans. Signal Proc.* **2013**, *61*, 2850–2865. [[CrossRef](#)]
5. Chen, Z.; Zhu, K.; Ying, L. Detecting multiple information sources in networks under the SIR model. *IEEE Trans. Netw. Sci. Eng.* **2016**, *3*, 17–31. [[CrossRef](#)]
6. Zhu, K.; Ying, L. Information source detection in the SIR model: A sample-path-based approach. *IEEE/ACM Trans. Netw.* **2016**, *24*, 408–421. [[CrossRef](#)]
7. Wang, J.; Wang, W. To live or to die: Encountering conflict information dissemination over simple networks. In Proceedings of the IEEE INFOCOM, San Francisco, CA, USA, 10–14 April 2016.
8. Prakash, B.A.; Vreeken, J.; Faloutsos, C. Spotting culprits in epidemics: How many and which Ones? In Proceedings of the IEEE ICDM, Brussels, Belgium, 10–13 December 2012.
9. Ok, J.; Jin, Y.; Shin, J.; Yi, Y. On maximizing diffusion speed in social networks: Impact of random seeding and clustering. In Proceedings of the 2014 ACM International Conference on Measurement and Modeling of Computer Systems, Austin, TX, USA, 16–20 June 2014; pp. 301–313.
10. Most Famous Social Network Sites Worldwide as of January 2018, Ranked by Number of Active Users (in Millions). *Statista*. Available online: <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/> (accessed on 15 February 2018).
11. Sabidussi, G. The centrality index of a graph. *Psychometrika* **1966**, *31*, 581–603. [[CrossRef](#)]

12. Dong, W.; Zhang, W.; Tan, C.W.; Yi, Y. Rooting out the rumor culprit from suspects. In Proceedings of the IEEE ISIT, Istanbul, Turkey, 7–12 July 2013.
13. Pinto, P.C.; Thiran, P.; Vetterli, M. Locating the source of diffusion in large-scale networks. *Phys. Rev. Lett.* **2012**, *109*, 068702. [[CrossRef](#)] [[PubMed](#)]
14. Zhu, K.; Ying, L. Information source detection in networks: Possibility and impossibility results. In Proceedings of the IEEE INFOCOM, San Francisco, CA, USA, 10–14 April 2016.
15. Kapoor, S.; Ramesh, H. Algorithms for enumerating all spanning trees of undirected and weighted graphs. *SIAM J. Comput.* **1995**, *24*, 247–265. [[CrossRef](#)]
16. Sharma, G.; Joo, C.; Shroff, N.B.; Mazumdar, R.R. Joint congestion control and distributed scheduling for throughput guarantees in wireless networks. *ACM Trans. Model. Comput. Simul.* **2010**, *21*, 5:1–5:25. [[CrossRef](#)]
17. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graphs over Time: Densification laws, shrinking diameters and possible explanations. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August 2005.
18. Watts, D.J.; Strogatz, S.H. Collective dynamics of “small-world” networks. *Nature* **1998**, *393*, 440–442. [[CrossRef](#)] [[PubMed](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).