electronics

Article

Inferring the Hidden Cascade Infection over *Erdös-Rényi* (ER) Random Graph

Jaeyoung Choi

School of Computing, Gachon University, Seongnam-si 13120, Korea; jychoi19@gachon.ac.kr;
Tel.: +82-31-750-5829

Abstract: Finding hidden infected nodes is extremely important when information or diseases spread rapidly in a network because hints regarding the global properties of the diffusion dynamics can be provided, and effective control strategies for mitigating such spread can be derived. In this study, to understand the impact of the structure of the underlying network, a cascade infection-recovery problem is considered over an *Erdös-Rényi* (ER) random graph when a subset of infected nodes is partially observed. The goal is to reconstruct the underlying cascade that is likely to generate these observations. To address this, two algorithms are proposed: (i) a Neighbor-based recovery algorithm (NBRA(α)), where $0 \leq \alpha \leq 1$ is a control parameter, and (ii) a BFS tree-source-based recovery algorithm (BSRA). The first one simply counts the number of infected neighbors for candidate hidden cascade nodes and computes the possibility of infection from the neighbors by controlling the parameter α. The latter estimates the cascade sources first and computes the infection probability from the sources. A BFS tree approximation is used for the underlying ER random graph with respect to the sources for computing the infection probability because of the computational complexity in general loopy graphs. We then conducted various simulations to obtain the recovery performance of the two proposed algorithms. As a result, although the NBRA(α) uses only local information of the neighboring infection status, it recovers the hidden cascade infection well and is not significantly affected by the average degree of the ER random graph, whereas the BSRA works well on a local tree-like structure.

Keywords: graph; cascade model; hidden infection; source estimation; BFS tree

Citation: Choi, J. Inferring the Hidden Cascade Infection over *Erdös-Rényi* (ER) Random Graph. *Electronics* **2021**, *10*, 1894. https://doi.org/10.3390/electronics10161894

Academic Editor: Fabio Grandi

Received: 9 June 2021
Accepted: 4 August 2021
Published: 6 August 2021

1. Introduction

Recently, computer viruses, infectious diseases, and the spread of information have become universal occurrences in various types of networks. As examples, COVID-19 is transmitted from person to person, whereas certain rumors have spread over the Internet through the development of social network services, such as Facebook and Twitter. Given such circumstances, it is important to quickly prevent the spread of rumors, malicious information, or viruses through observations. In practice, however, the observation of such spreading events is often incomplete because some nodes or individuals are reluctant to reveal their infection status [1]. Hence, inferring a hidden infection may provide some hints regarding the global properties of the diffusion dynamics [2,3]. Furthermore, it often opens up effective control strategies for the optimal dissemination or mitigation of spread. The research on inferring hidden nodes and structures has recently received significant attention. Netrapalli et al. [4] considered the graph inferring problem, where only the infection times of the nodes are given. In this study, they tried to estimate which neighbors were influential based on multiple cascades spreading through the graph. To do this, they suggested a Maximum Likelihood Estimator (MLE) for recovering influential connections, which guarantees a necessary cascade sample complexity for an Independent Cascade (IC) model. Sundareisan et al. [5] investigated the problem of recovering missing infected nodes and the source nodes of an epidemic. For this, they proposed an algorithm called NETFILL, which solves these two inference problems simultaneously. In addition, Xiao et al. [6] considered a problem for reconstructing the underlying cascade that is likely to generate observations of infection in the graph. They estimated the infection probabilities by generating a sample of the probable cascades, and proposed several algorithms for sampling directed Steiner trees with a given set of terminals. However, it is known [6] that inferring a hidden infected node is extremely difficult, even if the network structure is given. This is because it is necessary to first identify the path through which an arbitrary node is infected in the graph, despite the potential of an exponential number of paths. It is also known [7] that finding a hidden infected node that has not yet been observed is still a difficult problem even if some information regarding which node is infected and when such infection occurred is given. Hence, many prior studies have considered some of the greedy approaches to sampling a candidate infection graph using Steiner tree sampling [6,7]. An Expectation-Maximization (EM) approach applied under a partial observation scenario was also investigated in Reference [8]. He et al. [9] proposed an algorithm that effectively recovers a graph with partial cascade samples by considering a generative model, which is referred to as a Multi-Cascaded Model (MCM). In addition, Choi et al. [10] investigated the classification of causes of infection (random infection versus cascade infection) in a network under a partial observation of the infection status. To do this, they also considered the hidden cascade recovery approach in their proposed iterative classification algorithm. However, only a tree structure has been studied due to its computational complexity. There have been several studies conducted on fault detection as a hidden infection node in a communication network system. Huang et al. [11] investigated the VINI testbed, which supports the choice of paths to monitor, detect, and confirm the existence of a failure, correlating multiple independent observations in a single failure event. Tosic et al. [12] studied the problem of distributed sensor failure detection in networks with a small number of defective sensors, whose measurements differ significantly from the neighboring measurements. Furthermore, some hidden state estimation problems [13,14] have been investigated for sliding mode control. Although these prior studies were conducted on various synthetic or real graphs, there have been a few studies on how the recovering tendency changes when the actual graph changes according to arbitrary parameters. However, this is an important problem in terms of suggesting how to control the spreading cascade when the degree of recovery varies significantly depending on the parameters of the graph.

Hence, in this paper, the recovering problem of a hidden cascade node is considered for the *Erdös-Rényi* (ER) random graph as represented by the average degree λ. The average degree is the average number of neighbors for all nodes in the graph. Although it is a synthetic graph, an ER random graph may have a local tree-like structure with a large diameter or a very dense graph structure with a small diameter as the parameter λ varies. Therefore, as a first step, we study the recovery performance according to the parameter λ for a relatively simple ER random graph rather than other real-world graphs in References [6,7]. The objective is to develop efficient algorithms that find hidden infected nodes in this parameterized ER random graph and to show how the recovery performance changes as the parameter λ changes.

The main contributions of this study can be summarized in more detail as follows:

1. First, the recovering hidden cascade infection problem is studied on the well-known ER random graph under the IC model. As shown in Figure 1, for a given number of nodes, the shape of the ER random graph varies greatly depending on the average degree λ. Hence, the effect of this parameter on the recovery performance was investigated for the first time.
2. Second, to solve the problem, an algorithm that detects whether the node is infected with only neighboring infection information is first designed, which gives the local infection status of the graph. For this, the possibilities that can occur with the infected neighbor nodes are parameterized by a single parameter, and the method of selecting the best parameter is investigated. Next, as global information regarding infection, an efficient algorithm is proposed that uses the location of the estimated cascade sources.

The infection probabilities from the sources are used, which is a new method that has yet to be approached.

3. Third, various simulations are conducted to obtain the recovering performance of the two proposed algorithms. As a result, although the first one uses only local information of the neighboring infection status, it recovers the hidden cascade infection well compared with other baseline algorithms. Further, it is shown that the detection performance of the second graph is also good if the ER random graph has a local tree structure.

The remainder of this paper is organized as follows. Section 2 discusses related studies. In Section 3, the infection and observation models and the goals of the study are introduced. In Sections 4 and 5, the neighbor- and BFS tree-source-based recovery algorithms are presented, respectively. In Section 6, the simulation results are presented, and, in Sections 7 and 8, some concluding remarks and future works are given.

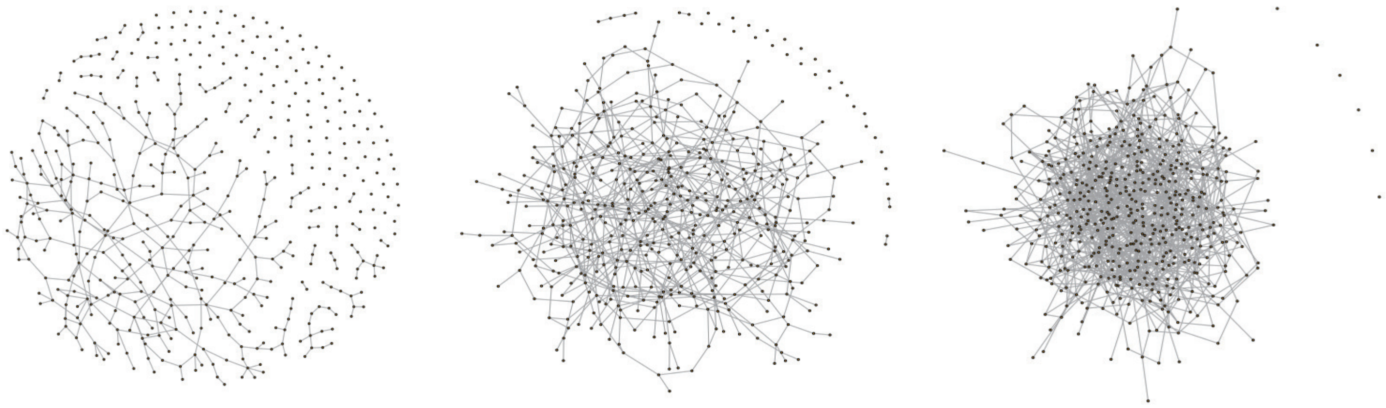


Figure 1. Illustration example of the ER random graph for 500 nodes. (We set the average degree as $\lambda = 1.5$ (**left**), $\lambda = 3$ (**middle**), and $\lambda = 4.5$ (**right**). The figure shows that, for a given number of nodes n, the ER random graph has local tree structures if the average number of degrees $\lambda \approx 1$.).

2. Related Work

The related studies are divided into the following two categories: (1) Hidden node (or structure) inference and (2) cascade source detection.

(1) ***Hidden node (structure) Inference.*** For the graph structure learning, Netrapalli et al. [4] considered the graph inference problem, where only the infection times of the nodes are given. In this study, the authors tried to estimate which neighbors are influential through multiple cascades spreading throughout the graph. To do this, they suggested a MLE for recovering influential connections, which guarantees a necessary cascade sample complexity for the IC model. Pouget et al. [15] studied the graph structure learning from the sparse recovery framework for a general discrete cascade model. They first formulated the problem as a general Linear Threshold (LT) model and proposed an algorithm that recovers the true edges in the graph with high probability. The authors of Reference [16] considered the graph learning problem for continuous-time cascade models. For this, they first introduced the concept of a trace complexity as the number of distinct traces required to achieve high fidelity in reconstructing the topology of an unobserved network. Based on this, they proposed a simple but efficient algorithm for inferring the graph structures. He et al. [9] proposed an algorithm that effectively recovers a graph with partial cascade samples by considering a generative model, referred to as the MultiCascades Model. As the key idea of this model, it uses the commonality between highly correlated diffusion graphs. He et al. [17] also designed some algorithms inferring the influence function from the partial observations of the node activation in the graph. To do so, they considered a probably approximately correct learnability of the influence functions for the missing observations. Rozenshtein et al. [18] has studied the reconstruction of cascades from partial timestamps. They designed an effective algorithm called a CULT to recover from epidemics by formulating a temporal Steiner tree problem. Xiao et al. [7] formulated a cascade-reconstruction problem as a variant of the Steiner tree problem. With their approach, they first estimate a tree that spans all reported active nodes, which satisfies some temporal consistency constraints. Furthermore, they proposed several algorithms and showed that one of them achieves a linear time $O(\sqrt{k})$-approximation guarantee, where k is the number of active nodes. Some researchers have studied the graph learning without the explicit time of infection. Gripon et al. [19] proposed an algorithm that recovers infection edges, which are the edges in the graph that are involved in the infection process from an unordered set of infected nodes. They designed an algorithm to infer such edges in order by comparing each trace into a path using information in which pairs of nodes in the path co-occur most frequently in the observations. Amin et al. [20] investigated the problem of reconstructing the node connectivity using only the initial source nodes and the final diffusion snapshot of infected nodes for the IC model, without any time information of the nodes. Wu et al. [8] considered an EM approach for the Continuous Independent Cascade (CIC) model using partial information on the infection status of the graph, and proposed algorithms that estimate the influence parameters of edges based on the EM. The authors of Reference [5] investigated the problem of recovering missing infected nodes and the source nodes of an epidemic. To do so, they proposed an algorithm called NETFILL, which solves these two inference problems simultaneously. The authors of Reference [6] considered a problem that reconstructs the underlying cascade that is likely to generate observations of infections in the graph. They estimated the infection probabilities by generating a sample of the probable cascades, proposed several algorithms for sampling directed Steiner trees with a given set of terminals.

(2) ***Cascade Source Detection.*** Next, some of the cascade source detection problems are summarized in the following sections. Zhu et al. [21] considered the cascade source detection problem using an ER random graph. They proposed a new source localization algorithm for the IC model, called the Short-Fat Tree algorithm, which selects the node of the Breadth-First Search (BFS) tree, which has the minimum depth but the maximum number of leaf nodes. In addition, they also proposed a new source localization algorithm, called an Optimal-Jordan-Cover (OJC) in Reference [22], which extends the source inferring problem to the case of multiple sources in the graph. The OJC first extracts a subgraph using a candidate selection algorithm that selects the source candidates based on the number of observed infected nodes in their neighborhoods. Considering the heterogeneous SIR diffusion in the ER random graph, they proved that OJC can locate all sources with a probability of 1 asymptotically with partial observations. Next, some studies have utilized partial information on when a random node is infected. The authors of Reference [23] studied the problem of partial timestamps of infection. They solved this problem by formulating a ranking problem on graphs using the likelihood of being the source. In the opposite direction as the previous source detection problem, Fanti et al. [24] first studied how to spread as many anonymous messages as possible while hiding the source. They designed a message-propagation protocol, called an adaptive diffusion model, and then obtained analytical results of the hiding performance of the model. The authors of Reference [25] considered the source-finding problem with observation time information O_t, which contains some set S of nodes with the first infection timestamps $\{t_s\}_{s \in S}$. Tang et al. [26] investigated the problem based on the network topology and a subset of infection timestamps. For a tree network, they first derived the maximum likelihood estimator of the source and unknown diffusion parameters. Using this, they considered an optimization over a parametrized family of Gromov matrices in the design of an estimation algorithm for general graphs. Kumar et al. [27] considered a problem with additional relative information about the infection times of a fraction of node pairs.

3. Model and Goal

3.1. Models

Graph Model. The recovery problem of a hidden cascade node for the ER random graph G, as represented by some graph parameters, is considered. An ER random graph, which is usually denoted by $G := G(n, p)$ on the vertex set V, is a random graph that connects each pair of nodes with probability $p > 0$. This model is parameterized by the number of nodes, $n = |V|$ and p. The expected number of degrees of a node is then computed by $\lambda = (n-1)p$ (see Figure 1). Most results for ER graphs suppose that this is fixed (e.g., $\lambda = 1$ or $\lambda = 2$) or increases slowly (e.g., $\lambda \simeq \log n$). Although it is a synthetic graph, the ER random graph may have a local tree-like structure with a large diameter or an extremely dense graph structure with a small diameter as the parameter λ varies.

Cascade and Observation Models. As a cascade model, a well-known independent cascade model is considered. With this model, three possible states of nodes are considered: Susceptible (S), Active (A), or Inactive (I). First, a susceptible state means that a node can be activated. Next, if a node in the susceptible state is activated at the previous time slot, it becomes an active node, which is in a state in which other susceptible child nodes are activated. The inactive state denotes a state in which a node was activated once earlier, but no longer able to activate other susceptible nodes. In the proposed model, the cascade starts from a subset of seed nodes that have been initially infected in the network. The cascade occurs in a discrete round, $t = 0, 1, \dots, |V|$. At round $t = 0$, the seed nodes are active, and the others are inactive. Then, if a node u receives the information or is infected from one of its infected nodes at time t, the node spreads its own information (or a virus) to its neighbor v with probability $p_{u,v}$ at the next time $t + 1$. It is assumed that the activated nodes are active for only one time slot, and they become inactive in the next time slot. Once a node becomes inactive, it maintains the state until reaching the end of the cascade process. The process stops when no more nodes are infected. For a simpler expression, the probability p_e is also used, where $e \in E$ is the edge among the nodes in the network. When some neighbors are already infected, the probability of infection p_e is regarded as zero. Let $\overline{p} := [p_e]_{e \in E}$ be the diffusion vector over each edge $e \in E$. An infected node v is considered to report its status of infection with probability $\theta_v \in [0, 1]$, labelled based on an observation probability and denoted as $\overline{\theta} := [\theta_v]_{v \in V}$. The node $s^* \in V$ represents the information source, which acts as a node that initiates diffusion, and the set $V_I \subset V$ indicates the set of observed (reported) infected nodes. Further, G_{obs} represents the observed snapshot of the cascade graph at time $t > 0$. Finally, assume that the number of cascade sources is given as a prior by $m > 0$ for a certain analytical tractability. We summarize the explanations of notations used in the paper in Table 1.

3.2. Goal: Estimation of an Infection

The goal of the paper is to recover the hidden (not reported) infected nodes (in the IC model, an infected node indicates both active and inactive nodes), excluding the reported infected nodes V_I. More precisely, for node $v \in V \setminus V_I$ with the ground truth $s(v) \in \{\text{Infected, Susceptible}\}$, the objective is to design an estimator of $s(v)$, denoted by $\hat{s}(v)$ such that $\hat{s}(v) : V \setminus V_I \to \{\text{Infected, Susceptible}\}$, which maps the node $v \in V \setminus V_I$ to the state of infection $\{\text{Infected, Susceptible}\}$ with high accuracy. Here, $\hat{s}(v)$ is called based on the recovery function of $s(v)$.

Performance metric. As a performance measure for the true recovery of the proposed estimation algorithm, the following precision-based accuracy is used: Let V_{HI} be the set of hidden infected nodes (ground truth) and $\hat{H}_{recover}$ be the estimated recovery set of infected nodes by the estimator $\hat{s}(v)$. As a measure of the hidden cascade recovery, a precision metric is defined by

$$P_{Recover}(\hat{s}(v)) := \frac{\sum_{v \in \hat{H}_{recover}} \mathbb{I}\{v \in V_{HI}\}}{|\hat{H}_{recover}|}, \tag{1}$$

where $\mathbb{I}\{v \in V_{HI}\}$ is an indicator function that has only one function for $v \in V_{HI}$; otherwise, it is zero. The notion $|\hat{H}_{recover}|$ is the number of infected nodes for $\hat{H}_{recover}$. This metric measures the rate at which the number of recovered nodes is truly infected among the estimated recovered set $\hat{H}_{recover}$ for the observed snapshot G_{obs} at time $t > 0$. To obtain this, two classification algorithms are proposed. The first is a neighbor-based approach that uses local infection status information. The second approach is a source-location-based approach that uses global infection status information from the cascade source nodes. Various simulations are then conducted, and the results are obtained by varying the model parameters described in Section 6.

Table 1. Summary of notations.

Notation	Explanation
$G(n,p)$	ER random graph with two parameters n and p
λ	Average degree of a node in $G(n,p)$
$p_{u,v}$	Infection probability from node u to v
p_e	Infection probability over edge e
θ_v	Infection reporting probability for node v
$\overline{p}$	Infection probability vector for all edges
$\overline{\theta}$	Infection-reporting probability vector for all nodes
s^*, S	(True) Cascade source, Set of cascade sources
V_I	Set of observed (reported) infected nodes
G_{obs}	Observed cascade snapshot at time t
m	Number of cascade sources
$s_i, \hat{S}$	Estimated (i-th) cascade source, Estimated set of cascade sources
$s(v)$	Infection status of a node v
$\hat{s}(v)$	Estimator of $s(v)$
$P_{Recover}(\hat{s}(v))$	recovery probability of $\hat{s}(v)$
V_{HI}	Set of hidden (not reported)-infected nodes
$\hat{H}_{recover}$	Estimated recovering set
$N_I(v)$	Set of (reported) infected neighbors of a node v
$p_{min,v}$	Minimum diffusion probability among $p_{u,v}$ for all $u \in N_I(v)$
$e(S, V_I)$	Infection eccentricity of node set S over V_I
ζ	Selection threshold
W	Set of nodes with more than ζ observed infected neighbors
W'	Union of W and V_I
g'	Induced graph (connected subgraph of G_{obs}) from W'
G_{s_i}	Connected sub-graph of G including s_i
$T_{BFS}(s_i, G_{s_i})$	BFS tree over G_{s_i} w.r.t. s_i
$C_{s_i}(v)$	Candidate neighbor (infected) set of v in $T_{BFS}(s_i, G_{s_i})$
$T(s_i, G_{s_i})$	Reconstructed graph from $T_{BFS}(s_i, G_{s_i})$
I_S	Union of V_I and V_{HI}
T	Union of reconstructed graphs $T(s_i, G_{s_i})$
$\mathcal{P}(v, s_i)$	Path between node v and source node s_i in $T(s_i, G_{s_i})$
η_v	Pre-defined threshold for determining the infection of node v
$P_{max}(S)$ ($P_{min}(S)$)	Maximum (minimum) infection probability over T from S

4. Neighbor-Based Recovery Algorithm

In this section, a naive approach is first introduced that uses the number of infected neighbor nodes to recover the hidden infection. As the motivation of this approach, if there are many infected neighbors of a node v, it also has a high probability of being infected. However, it is known that Reference [6] is an highly complicated task to determine which path the infection has progressed along when centring on a single node. Therefore, in the following subsection, a simple but efficient algorithm is proposed that estimates whether node v is infected, based on the degree of infection around it.

Neighbor-Based Recovery

In the neighbor-based recovery algorithm, called NBRA(α), the main idea is to properly consider the possibility that a node is infected from already infected nodes. The detailed description of this is as follows.

Algorithm. In NBRA(α), the algorithm first initializes the recovering set $\hat{H}_{recover}$ as an empty set. Then, for the nodes $v \in V \setminus V_I$, it counts the number of observed infected neighbor nodes, denoted by $N_I(v)$, and sets the minimum diffusion probability among $N_I(v)$ by $p_{min,v} := \min_{u \in N_I(v)} p_{u,v}$. Next, a heuristic infection probability $p(v)$ is defined from the infected neighbor nodes of node v. In this algorithm, the probability $p(v)$ is set to zero when there are no infected neighbor nodes, i.e., $N_I(v) = \emptyset$. Otherwise, it sets

$$p(v) := \alpha p_{min,v} + (1-\alpha)[1 - \prod_{u \in N_I(v)} (1 - p_{u,v})], \tag{2}$$

where $0 \le \alpha \le 1$ is a trade-off parameter. Finally, the algorithm puts the node v into the recovering set $\hat{H}_{recover}$ if the probability $p(v)$ is greater than $1/2$ (this approach implies that, if the heuristic infection probability from the infected neighbor nodes is larger than $1/2$, it is regarded as an infected node).

As shown in Algorithm 1, the result recovered set $\hat{H}_{recover}$ depends completely on the probability in Equation (2), which is controlled by the parameter α. The implications for this equation are described using the following three cases:

(i) **Case I: Single candidate neighbor** $(\alpha = 1)$**.** When $\alpha = 1$, it results in $p(v) = p_{min,v}$. This is the case in which there is only one possible candidate, i.e., already infected neighbor, with the minimum diffusion probability among all infected neighbors $N_I(v)$. In this case, it is considered that one neighbor node is infected first, node v is then infected by the neighbor node, and the remaining infected neighbor nodes are, thus, consequently infected. This is the lowest probability of all possible cases of infection order among $N_I(v) \cup \{v\}$, except for the case in which node v is infected first because only one possible infection is considered.

(ii) **Case II: Multiple candidate neighbors** $(\alpha = 0)$**.** When $\alpha = 0$, $p(v) = 1 - \prod_{u \in N_I(v)}(1 - p_{u,v})$ occurs. This is the case in which all nodes in $N_I(v)$ are infected before the infection of v. Hence, there are many possibilities of infection from the infected neighbors, i.e., the node v will be infected if there exists at least one infection successfully among the infected neighbor nodes. This is the greatest probability from $N_I(v) \cup v$, except that the infection to node v comes from other hidden infection nodes because infection from all infected neighboring nodes is considered.

(iii) **Case III: Relaxation case** $(0 \le \alpha \le 1)$**.** When only the infected node v and the infection snapshots for neighboring infected nodes are given, it is difficult to estimate the path of the infection. Hence, in this algorithm, various choices of $0 \le \alpha \le 1$ can be considered. As depicted above, when the value α reaches 1, it estimates that the infection comes from only one possible node that has the minimum diffusion probability among the infected neighbor set $N_I(v)$. However, if the value of α reaches zero, the infection occurs in at least one neighbor infected node. In the numerical section, some simulation results show how the hidden infection changes according to α.

It is not difficult to verify that the complexity of the NBRA algorithm (α) is $O(nd)$, where d is the maximum degree of nodes in the ER random graph. It is known [6] that computing the exact probability $P(v \in V_{HI})$ for node $v \in V \setminus V_I$ is a #P-hard problem. This is due to the fact that there may exist exponentially many possible infection paths in the graph. Hence, in the following theorem, the probability that node $v \in V_{HI}$ will be detected as an infected node is obtained using the NBRA(α).

Algorithm 1 Neighbor-based Recovery Algorithm (NBRA(α))

```
Input: Observed graph $G_{obs}$, Diffusion probability vector $\overline{p}$, Parameter $\alpha$
Output: Recovered Infection Set $\hat{H}_{recover}$
Set $\hat{H}_{recover} = \emptyset$;
for $v \in V \setminus V_I$ do
    Set $|N_I(v)|$ by the number of infected neighbors and set $p_{min,v} := \min_{u \in N_I(v)} p_{u,v}$;
    if $|N_I(v)| = 0$ then
        Set $p(v) = 0$;
    else
        Set $p(v) := \alpha p_{min,v} + (1-\alpha)[1 - \prod_{u \in N_I(v)}(1 - p_{u,v})]$;
        if $p(v) > 1/2$ then
            $\hat{H}_{recover} \leftarrow \hat{H}_{recover} \cup \{v\}$;
        end if
    end if
end for
Return $\hat{H}_{recover}$;
```

Theorem 1. *For any hidden infected node $v \in V_{HI}$ with the number of observed infected nodes $n_v := |N_I(v)| > 1$, if $\alpha \leq \frac{(1-p_{min,v})^{n_v} - 1/2}{(1-p_{min,v})((1-p_{min,v})^{n_v - 1} - 1)}$, the algorithm NBRA($\alpha$) outputs the node v as an infected node with probability one,* i.e., $P\{v \in \hat{H}_{recover}\} = 1$.

A detailed proof is presented in Appendix A. This result indicates how to choose the tradeoff parameter α for a given $|N_I(v)| > 1$ properly in the algorithm. Indeed, for $|N_I(v)| = 1$, it can be seen that, if $p_{min,v} > 1/2$, the algorithm NBRA(α) outputs the node v as an infected node with probability one. The following example shows how to properly choose the parameter α for a given observed graph G_{obs}.

Example 1. *Suppose there are five nodes in the ER random graph, as shown in Figure 2. Assume that node v is a hidden infected node with three infected neighbors, $|N_I(v)| = 3$, among the four neighbors. The infection probabilities $p_{i,v}$ $(i = 1, 2, 3, 4)$ are given in the figure, and $p_{min,v} = 0.3$, because node 4 is not an infected node in the graph. From Theorem 1, if the parameter α is set by*

$$\alpha \leq \frac{(1-0.3)^3 - 1/2}{(1-0.3)((1-0.3)^{3-1} - 1)} = \frac{(1-0.3)^3 - 1/2}{(1-0.3)((1-0.3)^{3-1} - 1)} = \frac{0.157}{0.357} \cong 0.44, \quad (3)$$

then $p(v) > 1/2$ and the algorithm correctly find the hidden cascade node.

In this example, because the number of infected nodes among the neighbor nodes of hidden node v is 3, and $p_{min,v} = 0.3$, if α is less than 0.44, the infection of v can be easily detected. However, it is difficult to find the optimal α for all hidden infected nodes. Hence, in Section 6, the case of finding an appropriate α through simulations is presented. Although this algorithm has the advantage of being relatively simple, there is a limit to the fundamental recovery performance because it uses only the information of neighboring infected nodes. Hence, in the following section, a more complicated and novel approach is presented that uses the information of the location for the cascade sources properly.

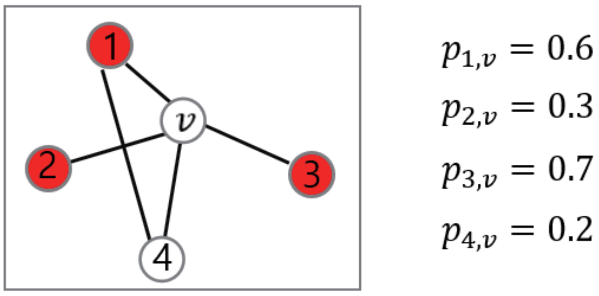


Figure 2. A simple observed graph G_{obs} with infection probabilities $p_{i,v}$, $(i = 1,2,3,4)$ used in Example 1.

5. BFST Source-Based Recovery Algorithm

In the cascade source-based approach, information on the location of the cascade sources is used to recover the hidden infected nodes. To do this, multiple cascade sources need to be estimated using the observed cascade snapshot. The infection status of the hidden infected nodes is then inferred through a reconstruction from the graph based on the BFS tree. We describe this approach in Algorithm 2 and the detailed descriptions for each part are as follows.

5.1. Source Estimation

Optimal Jordan Cover. For cascade source estimation, OJC is adopted as a multiple source estimator, which was first introduced in Reference [22]. This approach was developed for the SIR diffusion model, which is a generalized version of the IC model, by setting the recovery rate to 1. Furthermore, it also considers the partial cascade infection snapshot over the graph. To formally describe this, a hop-distance between a node v and a node set S is defined to be the minimum hop-distance between the node v and any node in S by $d(v,S) := \min_{u \in S} d(u,v)$. Here, $d(u,v)$ is the shortest hop distance between two nodes u and v. Then, the infection eccentricity of the node set S as the maximum hop distance from an infected node in V_I is defined by

$$e(S, V_I) := \max_{v \in V_I} d(v,S). \tag{4}$$

Based on this, the OJC algorithm consists of the following two steps:

(a) **Step 1 (Candidate Set Selection):** In this step, the algorithm first selects the candidate source nodes by considering the number of infected (observed) neighbors. To do that, let $\zeta > 0$ be a positive integer, which is called the selection threshold. Then, define a candidate set of the cascade sources W as the set of nodes whose number of infected neighbors is greater than the predefined threshold ζ. Next, the algorithm sets $W' := W \cup V_I$ by the union of the candidate sets W and the observed infection set V_I. This procedure considers a node that is not an infected node in a limited observation, but is infected with many neighbor nodes. It then sets g' by a connected sub-graph of G_{obs} induced by the node set W'. Such a reconstructed graph is called an induced graph. An induced graph is a subset of nodes in a graph with all edges having endpoints in the node subset. This refers to the part in which the graph is reconstructed as new nodes are added to the infection graph created from the original infected node. However, the induced graph may not be fully connected because of the hidden infected nodes or multiple sources. This becomes a problem when it is necessary to determine the infection eccentricity in the infection graph. Therefore, if the induced graph is separated, a node is selected with minimum eccentricity for each sub-graph (component) and connects it to connect the entire induced graph. Subsequently, the minimum eccentricity of the induced graph g' is computed. This is a modified part of the OJC, which randomly selects such nodes in each component.

This modification is considered to ensure that the central node of each cluster of the cascade does not lose its high probability of becoming a candidate source (although, considering a hidden infection, this will be a natural step to guessing that the high centrality node of a cluster may have a high possibility of becoming a source due to the randomness of hiding its status of infection in the proposed model). The pseudocode of the candidate selection algorithm for selecting W and g' can be found in Reference [22].

(b) **Step 2 (Jordan Cover Selection):** Using the result of the candidate set W in Step 1, the algorithm computes the infection eccentricity of the node set $S \subset W$ with $|S| = m$, as defined in Equation (4) on the sub-graph g'. Next, a combination is selected with the minimum infection eccentricity as a set of sources. Then, the m-Jordan cover $\hat{S}$ is chosen by

$$\hat{S} = \arg \min_{S \subset W, |S|=m} e(S, V_I). \tag{5}$$

Ties are broken by the total distance from the observed infection to the node set, that is, $\sum_{v \in V_I} d(v, S)$.

It is known that the candidate selection step includes all sources in W with high probability and excludes nodes that are more than $t + 1$ hops away from all sources by a properly chosen threshold ζ. By limiting the computations on the induced subgraph g', the computational complexity is significantly reduced. The results show that, under certain conditions, the OJC identifies all sources with a probability one asymptotically in the ER random graph.

5.2. BFS Tree-Based Recovery Hidden Infection

Recovery Hidden Infection. The algorithm infers the infection status of the hidden infection nodes using the location of the estimated sources. To do this, the infection probability is first computed for node $v \in V \setminus V_I$, from the cascade sources. If the probability is larger than a predefined threshold, it regards node v as the hidden infection node. However, computing the infection probability from the source is known to be an NP-hard problem [6] in the general loopy graph because of the exponential number of infection paths. Hence, instead of computing this, the infection probability is approximated after obtaining several BFS trees based on the estimated sources from the underlying ER random graph. This procedure is described in the following two steps.

(a) **Step 1 (Candidate Infection Path Selection):** In this step, the algorithm first constructs BFS trees based on the estimated source node $s_i \in \hat{S}$, $(i = 1, \ldots, m)$ from Phase 1. To do this, let G_{s_i} be the connected subgraph of G as in Figure 3, which contains the source s_i and let $T_{BFS}(s_i, G_{s_i})$ be the BFS tree with respect to the G_{s_i} (Because the underlying graph is a random graph, some disconnections may exist. If there is no separation of G, G_{s_i} is equal to G.). Then, m such BFS trees are obtained on the underlying graph G because there are m sources. If $G_{s_i} \neq G$, i.e., G is disconnected, the BFS tree $T_{BFS}(s_i, G_{s_i})$ is generated only on the connected subgraph G_{s_i}. However, the BFS tree generates a unique candidate infection path from the source s_i to node v. This is quite limited in general graphs that can have multiple paths. With the proposed algorithm, to guarantee such multiple infection paths, some candidate neighbors (here, the candidate neighbor means a neighbor node that can infect the node v among the neighbors in the original graph) of each node from v to the source node are first found among the infected nodes based on the BFS tree. To this end, the candidate neighbor set $C_{s_i}(v)$ is defined in each BFS tree as

$$C_{s_i}(v) := \{u \in N_I(v) | d(s_i, u) \leq d(s_i, v) - 1 \text{ in } T_{BFS}(s_i, G_{s_i})\}, \tag{6}$$

where $d(s_i, u)$ is the hop distance from node s_i to $u \in N_I(v)$. The algorithm then embeds an edge from a candidate infected neighbor node to v. Next, for the chosen nodes in $C_{s_i}(v)$, the candidate neighbor procedure is repeated. It performs this procedure until the candidate neighbor is exactly the estimated source. The graph reconstructed by $T(s_i, G_{s_i})$ is then denoted as in Figure 3. As the reason for this approach, the BFS preserves the infection path in the IC model. Further, it removes the loops in the original graph from the estimated source node to v while guaranteeing the candidate infection paths.

(b) **Step 2 (Recovery Hidden Infection):** Next, the algorithm computes the infection probability for the given cascade sources with the corresponding BFS based on the reconstructed graphs $T(s_i, G_{s_i})$ under the IC diffusion model. To do this, let $I_S := V_I \cup V_{HI}$ be the set of infected nodes (including the observed infection set V_I and unobserved infection set V_{HI}) by the cascade from S. The objective is then to compute the probability $\mathbb{P}(v \in I_S)$, i.e., the probability of infection of node v from the source S. However, it is not easy to compute this probability in a general loopy graph because of the computational complexity, as described above. Hence, the probability $\mathbb{P}(v \in I_S)$ is approximated by $\mathbb{P}_T(v \in I_S)$ over T, where $T = \cup_{s_i \in S} T(s_i, G_{s_i})$. To compute this in the IC model, let $\mathcal{P}(v, s_i)$ be the path between node v and the source node s_i in $T(s_i, G_{s_i})$. Then, we have

$$\mathbb{P}_T(v \in I_S) = 1 - \prod_{s_i \in S} \prod_{\mathcal{P}(v,s_i) \subset T(s_i, G_{s_i})} \left(1 - \prod_{e \in \mathcal{P}(v,s_i)} P_e\right), \tag{7}$$

where P_e is the probability that a diffusion will occur over the edge $e \in \mathcal{P}(v, s_i)$ in $T(s_i, G_{s_i})$. For example, if the diffusion is an IC model with a successful probability of $q > 0$, the probability in Equation (7) becomes

$$\mathbb{P}_T(v \in I_S) = 1 - \prod_{s_i \in S} \prod_{\mathcal{P}(v,s_i) \subset T(s_i, G_{s_i})} \left(1 - q^{|\mathcal{P}(v,s_i)|}\right), \tag{8}$$

where $|\mathcal{P}(v, s_i)|$ is the number of edges over the path $\mathcal{P}(v, s_i)$ in $T(s_i, G_{s_i})$. Using this, the algorithm determines a node as a cascade-infected node if the diffusion probability is greater than a predefined threshold, i.e., $P(v) := \mathbb{P}_T(v \in I_S) \geq \eta_v$ as a cascade infected node. Then, it remains how to choose such a parameter η_v. For this, $\eta_v = (P_{max}(S) + P_{min}(S))/2$ is set. Here, $P_{max}(S) := \max_{v \in V_I} \mathbb{P}_T(v \in I_S)$ and $P_{min}(S) := \min_{v \in V_I} \mathbb{P}_T(v \in I_S)$ are the maximum and minimum infection probabilities over $T = \cup_{s_i \in S} T(s_i, G_{s_i})$, respectively. The reason for the threshold is that it can be a measure of how the diffusion spreads from the current observation.

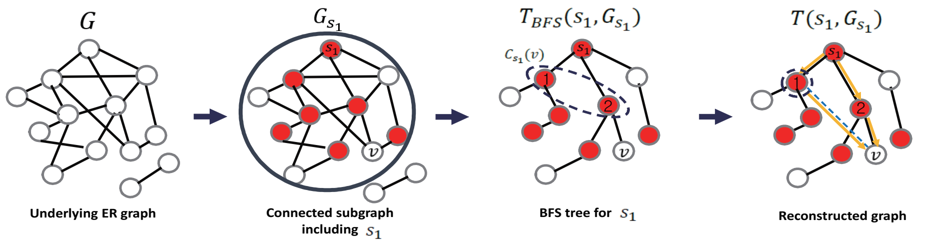


Figure 3. Example of Step 1 (Candidate Infection Path Selection): In the example, the underlying ER random graph G is generated as two disconnected sub-graphs (leftmost). If there is only one cascade source, the cascade occurs only in one connected sub-graph, denoted as G_{s_1} (second from the left). Based on this, the algorithm first constructs the BFS tree $T_{BFS}(s_1, G_{s_1})$ (third from the left). It then selects the set of candidate neighbors $C_{s_1}(v)$ for node v. In this example, the distance from the estimated source s_1 to v is $d(s_1, v) = 2$. Therefore, two nodes (labelled as 1 and 2) are chosen as candidate neighbor nodes because their distance from s_1 is 1. Node 2 is already a neighbor in $T_{BFS}(s_1, G_{s_1})$, and node 1 is finally selected as the candidate neighbor and connects an edge to the node v (rightmost). This is referred to as a reconstructed graph $T(s_1, G_{s_1})$, and the algorithm selects the candidate infection paths including the candidate neighbor node (yellow arrows).

Algorithm 2 BFST Source-based Recovery Algorithm (BSRA)

```
Input: Observed graph $G_{obs}$, Diffusion probability vector $\overline{p}$, Number of sources $m$,
Hidden probability vector $\overline{\theta}$, Threshold $\eta$, Threshold $\zeta$,
Output: Recovered Infection Set $\hat{H}_{recover}$
Phase 1. (Source Estimation)
Set $W = \emptyset$;
For the given $(G_{obs}, V_I, \zeta)$, run the Candidate Set Algorithm in Reference [22] and obtain
the output $(g', W)$, where $g'$ is an induced graph and $W$ is the candidate source set.
for $S \subset W, |S| = m$ do
    Compute the infection eccentricity on $g'$: $e(S, V_I) = \max_{v \in V_I} d(v, S)$;
end for
Set the combination with the minimum infection eccentricity as the set of sources;
    $$\hat{S} = \arg \min_{S \subset W, |S| = m} e(S, V_I).$$
Phase 2. (BFS-based Hidden Cascade Recovery)
Set $G_{s_i}$ by the connected sub-graph of $G$ including the estimated source $s_i \in \hat{S}$;
for each $s_i \in \hat{S}$ do
    Generate a BFS tree $T_{BFS}(s_i, G_{s_i})$ based on the estimated source $s_i$ in Phase 1;
end for
Set $\hat{H}_{recover} = \emptyset$;
for $v \in V \setminus V_I$ do
    Find all candidate neighbors in $T_{BFS}(s_i, G_{s_i})$ and add the edges to $v$;
    Reconstruct a graph $T(s_i, G_{s_i})$ from $T_{BFS}(s_i, G_{s_i})$ by embedding the edges using
Equation (6) recursively and compute $P(v)$ using Equation (7) with respect to $T(s_i, G_{s_i})$
for all $s_i \in \hat{S}$
    if $P(v) \geq \eta_v$ then
        $\hat{H}_{recover} \leftarrow \hat{H}_{recover} \cup \{v\}$ w.p. $1 - \theta_v$;
    end if
end for
Return $\hat{H}_{recover}$;
```

The overall process of Algorithm 2 is depicted as in Figure 4. Next, the computational complexity of the proposed algorithm is described in the following lemma.

Lemma 1. *Let $w := |V(g')|$ be the number of nodes for the induced connected graph g'. Then, the computational complexity of the BSRA is $O(|V_I|(|E| + \max\{n, w^m\}))$.*

Proof. First, by the result in Reference [22] and Stirling's formula, we have $O(|V_I|(|E| + w^m))$ for the Phase 1. Second, we have $O(m(n + |E|))$ for constructing the BFS tree with m estimated sources and $O(n|V_I|)$ for computing the reconstruct the graph after finding the original neighbor nodes in the graph. Hence, we have $O(n|V_I| + m|E|)$ since $m \leq |V_I|$ for the Phase 2. From the fact that $m \leq |V_I|$ again, we finally obtain the total complexity by $O(|V_I|(|E| + \max\{n^2, w^m\}))$, and this completes the proof. □

For the given m, the BSRA is a polynomial-time algorithm, but the complexity increases exponentially in m. To further reduce the complexity, one can consider the K-means algorithm [28] in Step 2 of the source estimation.

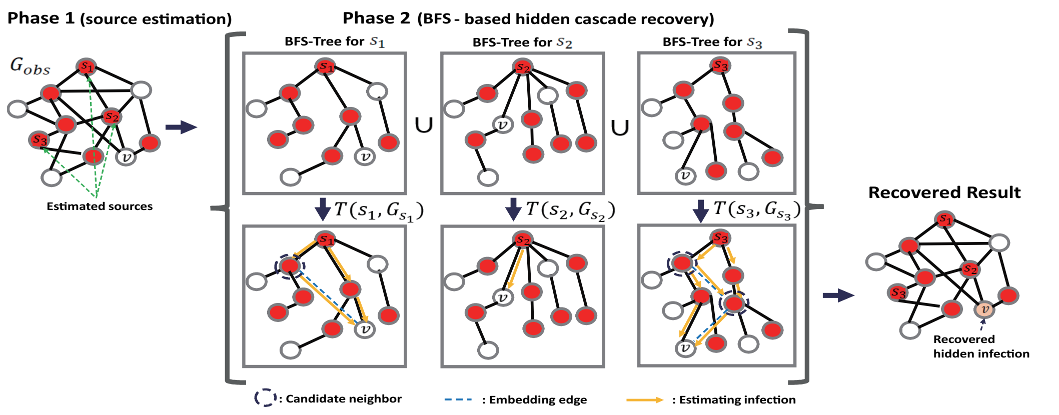


Figure 4. Illustration example of the Algorithm BSRA for $m = 3$. The BSRA first estimates the sources for a given m, then it makes m different BFS trees with respect to each source s_i $(i = 1, 2, 3)$. Next, for each node $v \in V \setminus V_I$, the BSRA reconstructs graph $T(s_i, G_{s_i})$ for each s_i (see Figure 3) and computes the infection probability using $T(s_i, G_{s_i})$. After computing the diffusion probability from the sources, it finally recovers the hidden infected node based on the criteria, as in the right figure.

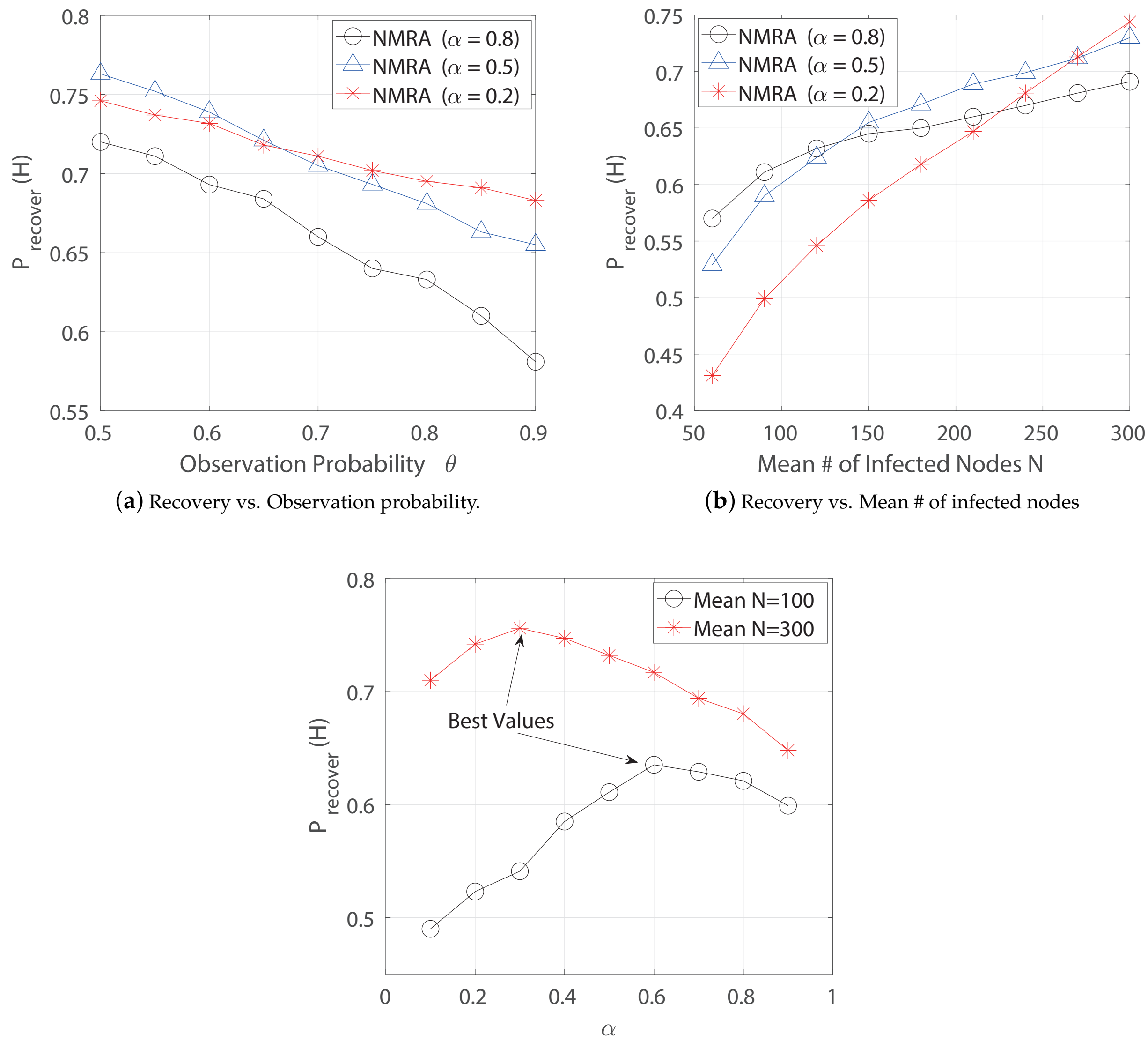
6. Simulation Results

In this section, simulation results of the proposed algorithms are provided on an ER random graph. For this graph, $n = 1000$, and the wiring probability p is set to generate the expected number of degrees of a node as $\lambda \in [1.6, 3.2]$. Furthermore, the cascade diffusion probability p_e for all edges $e \in E$ is chosen from the range $[0.6, 0.9]$, and the reporting probability θ_v is chosen from the range $[0.5, 0.9]$ uniformly. In BSRA, the threshold η_v is set to $\eta_v = (P_{max}(S) + P_{min}(S))/2$, as mentioned above. In the simulation, three cascade sources, i.e., $m = 3$, are considered. The performance of a different infection size x is evaluated. Under the assumption of cascade IC model, it is difficult to obtain the diffusion snapshots for a fixed x of infected nodes. Therefore, as in Reference [21], for each infection size x, the infection samples are generated in which the number of infected nodes is within the range $[0.75x, 1.25x]$. The nodes for the cascade were chosen uniformly at random among all nodes in the network. The value of x is varied from 60 to 300 with a step size 30. The metric in Equation (1) is used as a performance measure for the classification of the proposed estimation algorithm. For comparison, two algorithms, NETFILL [5] and Min-Steiner tree [6], are considered because both algorithms consider the location of the cascade sources to infer the hidden cascade infection. NetFill is a method designed specifically for the SI model, which assumes that a single propagation probability β is common to all edges. The min-Steiner tree is a baseline that constructs the minimum weight of the Steiner tree, where the weight of each edge $e = (u, v)$ is defined as $-\log p_{u,v}$.

6.1. Results

6.1.1. Neighbor-Based Recovery Algorithm

First, the recovery performance for NBRA(α) is presented in Figure 5. For the simulation, the average degree is set to $\lambda = 2$ to generate an ER random graph. In Figure 5a, the recovery probability in Equation (1) is depicted by varying the observation probability θ from 0.5 to 0.9. The results show that the overall recovery probability decreases as the observation probability increases. This is because, if the probability of the observation is small, the number of real infections other than the hidden nodes increases, as does the number of cases where the algorithm finds such nodes. Next, if the observation probability is approximately 0.5, then it will be good to choose a uniform trade-off parameter α. However, when the observation probability increases, a small α $(\alpha = 2)$ is better because the path of many infections can be estimated from the infection snapshot. In Figure 5b, the recovery performance is obtained by varying the number of infected nodes over the graph. The results show that, as the number of average infected nodes increases, the recovery performance also increases. This is because the NBRA algorithm (α) uses the degree of infection of the neighboring nodes to estimate whether an infection is actually occurring. As the number of infected neighbor nodes increases, the possibility of recovering the hidden infected nodes increases. Further, it can be verified that, when the number of infected nodes is small, choosing a large α ($\alpha = 0.8$) is better. If the infections increase, it will be better to use a small α ($\alpha = 0.2$) because of the diversity of the infection paths. In Figure 5c, the best value of α for different numbers of infected nodes in the graph is depicted. As a result, the number of infected nodes is large, and a small value of α will be good for recovering the hidden infected nodes, for the same reason as described before.



(**a**) Recovery vs. Observation probability.

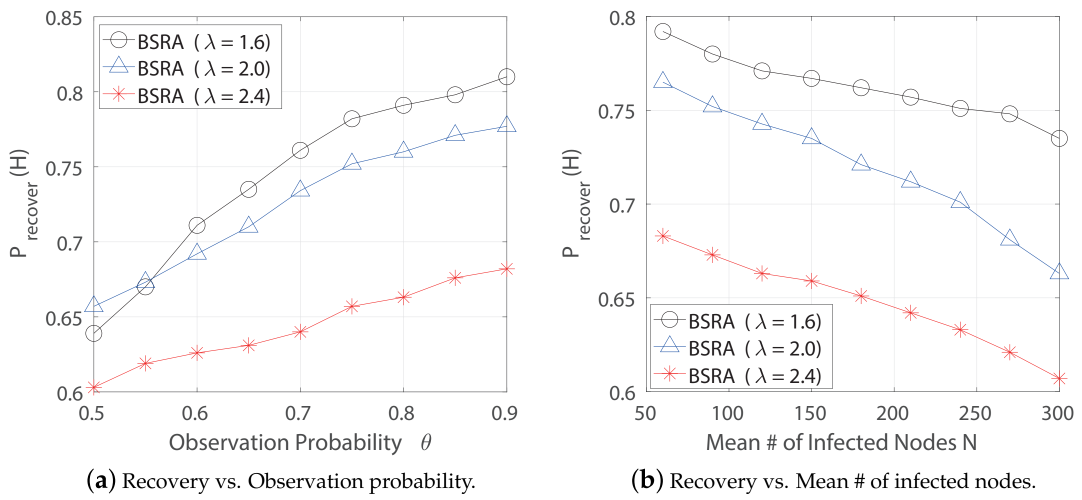
(**b**) Recovery vs. Mean # of infected nodes

(**c**) Recovery vs. Parameter α.

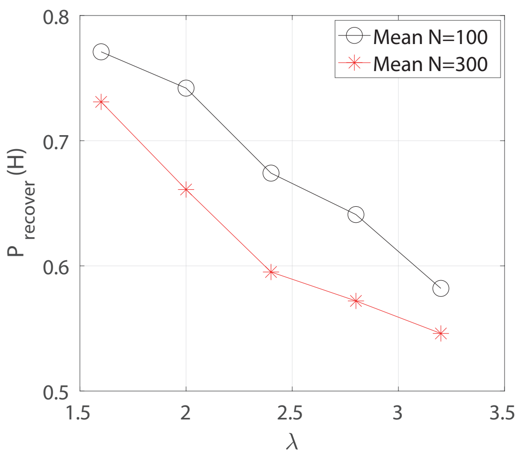
Figure 5. Recovery probabilities of NBRA(α) for $m = 3$. ((**a**) Recovery as the observation probability varies from 0.5 to 0.9. (**b**) Recovery by varying the mean number of infected nodes from 60 to 300. (**c**) Recovery by varying the parameter α from 0.1 to 0.9. Here, N indicates the mean number of infected nodes).

6.1.2. BFST Source-Based Recovery Algorithm

Next, the recovery performance of the BSRA is shown in Figure 5. For the simulation, the parameter η is set to 2 for the source estimation. As the first result of the BSRA, the recovery probability is obtained by varying the observation probability θ in Figure 6a. In contrast to the result of the NBRA (α), the overall recovery probability increases as the observation probability increases. This is because, when a partial snapshot is given, the BSRA first finds nodes that are estimated to be sources in the entire infection graph. The diffusion probability is then compared based on this to find hidden infections. It is known [22] that the performance of estimating such sources is better when the number of observed nodes is large. In addition, it can be verified that the recovery performance is better when the parameter of the ER random graph is $\lambda = 1.6$ than in other cases. This is because, when $\lambda \cong 1.5$, the underlying ER graph has a local tree structure, as shown in Figure 1, in which the proposed BFS tree-based algorithm works well. In Figure 6b, the recovery probability is presented by varying the mean number of infected nodes N. The result shows that the probability decreases as N increases owing to the fact that the source estimation accuracy decreases when the number of infected nodes becomes large in general [22]. The result also shows that the probability of recovery when $\lambda = 1.6$ is the largest among the three values ($\lambda = 1.6, \lambda = 2.0, \lambda = 2.4$) because of the local tree structure of the underlying graph. In Figure 6c, the result is obtained by varying the parameter λ from 1.6 to 3.2 for the mean number of infected nodes as 100 and 300, respectively. For both cases, the recovery probability decreases as λ increases from 1.6 for both cases. Furthermore, owing to the better source estimation result when a small number of infected nodes occurs in the graph, the overall recovery probability when $N = 100$ is better than that when $N = 300$.



(**a**) Recovery vs. Observation probability. (**b**) Recovery vs. Mean # of infected nodes.



(**c**) Recovery vs. Parameter λ.

Figure 6. Recovery probabilities of NBRA(α) for $m = 3$ and $\zeta = 3$. ((**a**) Recovery as the observation probability varies from 0.5 to 0.9. (**b**) Recovery by varying the mean number of infected nodes from 60 to 300. (**c**) Recovery as varying the parameter λ from 1.6 to 3.2).

6.1.3. Performance Comparison

To compare the performance of the recovery probability in Equation (1), four algorithms are considered: (i) NBRA(α), (ii) BSRA, (iii) Steiner, and (iv) NETFill. In the evaluation, each algorithm propagated the infection until 200 nodes in the graph on average. For NBRA(α), $\alpha = 0.4$ and $\beta = 0.5$ is set for NetFill. For Steiner, the min-Steiner tree method in Reference [6] is used. With this algorithm, however, a single source is considered; thus, the min-dist is used as the source estimation step, which selects the "centroid" node with the minimum weight in Reference [6] to estimate multiple sources as ranked values. The min-Steiner tree is then applied for the top-three sources separately and obtains the results. Finally, the recovery probabilities for each algorithm are listed in Tables 2 and 3 as the observation probability θ and the parameter λ, respectively. As a result, in Table 2, the recovery probability of BSRA is higher than that of NETFILL, which is designed for the SI-diffusion model. Further, the results show that the performance of NBRA(α) is not inferior to that of the other three approaches, despite using only the local neighbor infection state information. Finally, in Table 3, it is concluded that the overall recovery performance decreases as the parameter λ increases. However, BSRA performs well compared to the other methods.

Table 2. Recovery probabilities of four algorithms: (i) NBRA(α), (ii) BSRA, (iii) Steiner, and (iv) NETFILL as varying the observation probability θ.

θ	NBRA(α)	BSRA	Steiner	NETFILL
0.5	0.69	0.65	0.71	0.61
0.6	0.68	0.67	0.69	0.57
0.7	0.65	0.72	0.67	0.55
0.8	0.62	0.75	0.64	0.53
0.9	0.61	0.77	0.63	0.54

Table 3. Recovery probabilities of four algorithms: (i) NBRA(α), (ii) BSRA, (iii) Steiner, and (iv) NETFILL as varying the ER graph parameter λ.

λ	NBRA(α)	BSRA	Steiner	NETFILL
1.6	0.59	0.73	0.69	0.64
2.0	0.62	0.71	0.67	0.56
2.4	0.65	0.67	0.63	0.51
2.8	0.64	0.63	0.59	0.48
3.2	0.65	0.58	0.55	0.44

7. Conclusions

In this paper, a cascade infection-recovery problem was studied through a partial observation over an ER random graph. To do so, two algorithms, (i) a neighbor-based recovery algorithm and (ii) BFS tree source-based recovery algorithm, have been proposed. In the first algorithm, the number of infected neighbors were counted for candidate hidden cascade nodes and the possibility of infection was computed from the neighbors. In the second algorithm, the cascade sources were estimated, and the infection probability was computed from the BFS tree approximation for the underlying ER random graph with respect to the sources. Due to computational complexity, the approximated probability is considered for comparing the pre-defined criteria for recovery. The results show that the neighbor-based algorithm recovers the hidden cascade infection well and is not significantly affected by the average degree of the ER random graph. Furthermore, the BFS source-based method works well on a local tree-like structure.

8. Limitations and Future Works

The main advantage of the neighbor-based algorithm is that it does not require infection information spread over the entire graph to estimate whether a node is truly infected. In addition, it was confirmed that the results are not bad compared to the recovery performance of the other algorithms. In the case of the source-based algorithm, the recovery performance outperforms the comparison algorithms by adopting the OJC, which is known to be an extremely good technique for finding cascade sources in the ER graph. However, owing to mathematical intractability, the theoretical result obtained for the neighbor-based algorithm has a limitation in computing the probability of considering an actual uninfected node as infected. Furthermore, for the source-based method, when there is a loop in the graph, it is difficult to accurately calculate the infection probability from the source. Hence, a study on how to overcome these two limitations will be area of future work.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1G1A1099466).

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. Proof of Theorem 1

From the algorithm NBRA(α), we have the detection probability of the hidden infected node $v \in V_{HI}$ by

$$\begin{aligned} \mathrm{P}\{v \in \hat{H}_{recover}\} &= \mathrm{P}\{p(v) > 1/2\} \\ &= \mathrm{P}\{\alpha p_{min,v} + (1-\alpha)[1 - \prod_{u \in N_I(v)} (1 - p_{u,v})] > 1/2\}. \end{aligned} \tag{A1}$$

Let $A := \{\alpha p_{min,v} + (1-\alpha)[1 - \prod_{u \in N_I(v)}(1 - p_{u,v})] > 1/2\}$, and let $A' := \{\alpha p_{min,v} + (1-\alpha)[1 - (1 - p_{min,v})^{|N_I(v)|}] > 1/2\}$, where $p_{min,v} := \min_{u \in N_I(v)} p_{u,v}$. Then, we see that, if A$'$ is satisfied, then A is also satisfied, i.e., $A' \subseteq A$. Using this fact and by denoting $n_v := |N_I(v)|$, we have

$$\begin{aligned} A' &:= \{\alpha p_{min,v} + (1-\alpha)[1 - (1 - p_{min,v})^{n_v}] > 1/2 \\ &\Leftrightarrow \alpha p_{min,v} + (1-\alpha) - (1 - p_{min,v})^{n_v} + \alpha(1 - p_{min,v})^{n_v} > 1/2 \\ &\Leftrightarrow \alpha(1 - p_{min,v})((1 - p_{min,v})^{n_v - 1} - 1) > (1 - p_{min,v})^{n_v} - 1/2 \\ &\Leftrightarrow \alpha < \frac{(1 - p_{min,v})^{n_v} - 1/2}{(1 - p_{min,v})((1 - p_{min,v})^{n_v - 1} - 1)}, \ (*), \end{aligned} \tag{A2}$$

where the last step is from the fact that $(1 - p_{min,v})((1 - p_{min,v})^{n_v - 1} - 1) < 0$. Then, we have that, if α satisfies the condition $(*)$, then $P(A') = 1$, which implies $P(A) = 1$. This completes the proof.

References

1. Woo, J.; Ok, J.; Yi, Y. Iterative learning of graph connectivity from partially-observed cascade samples. In Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20), online, 11–14 October 2020.
2. Liu, Y.; Bao, Z.; Zhang, Z.; Tang, D.; Xiong, F. Information cascades prediction with attention neural network. *Hum. Cent. Comput. Inf. Sci.* **2020**, *10*, 13. [CrossRef]
3. Feng, X.; Zhao, Q.; Liu, Z. Prediction of information cascades via content and structure proximity preserved graph level embedding. *Inf. Sci.* **2021**, *560*, 424–440. [CrossRef]
4. Netrapalli, P.; Sanghavi, S. Finding the Graph of Epidemic Cascades. *ACM SIGMETRICS Perform. Eval. Rev.* **2012**, *40*, 211–222. [CrossRef]
5. Sundareisan, S.; Vreeken, J.; Prakash, B.A. Hidden Hazards: Finding Missing Nodes in Large Graph Epidemics. In Proceedings of the 2015 SIAM International Conference on Data Mining (SDM), Vancouver, BC, Canada, 30 April–2 May 2015.
6. Xiao, H.; Aslay, C.; Gionis, A. Robust Cascade Reconstruction by Steiner Tree Sampling. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2017.
7. Xiao, H.; Rozenshtein, P.; Tatti, N.; Gionis, A. Reconstructing a cascade from temporal observations. In Proceedings of the 2018 SIAM International Conference on Data Mining (SDM), San Diego, CA, USA, 3–5 May 2018.
8. Wu, X.; Kumar, A.; Sheldon, D.; Zilberstein, S. Parameter Learning for Latent Network Diffusion. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013.

9. He, X.; Liu, Y. Not Enough Data? Joint Inferring Multiple Diffusion Networks via Network Generation Priors. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017.
10. Choi, J. Identification of Individual Infection Over Networks With Limit Observation: Random vs. Epidemic? *IEEE Access* **2021**, *9*, 74234–74245. [CrossRef]
11. Huang, Y.; Feamster, N.; Teixeira, R. Practical Issues with Using Network Tomography for Fault Diagnosis. *ACM SIGCOMM Comp. Commun. Rev.* **2008**, *38*, 53–58. [CrossRef]
12. Tosic, T.; Thomos, N.; Frossard, P. Distributed sensor failure detection in sensor networks. *Signal Process.* **2013**, *93*, 399–410. [CrossRef]
13. Nasiri, M.; Mobayen, S.; Zhu, Q. Super-Twisting Sliding Mode Control for Gearless PMSG-Based Wind Turbine. *Complexity* **2019**, *2019*, 6141607. [CrossRef]
14. Jafari, M.; Mobayen, S. Second-order sliding set design for a class of uncertain nonlinear systems with disturbances: An LMI approach. *Math. Comput. Simul.* **2019**, *156*, 110–125. [CrossRef]
15. Abadie, J.P.; Horel, T. Inferring Graphs from Cascades: A Sparse Recovery Framework. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 7–9 July 2015.
16. Abrahao, B.; Chierichetti, F.; Kleinberg, R.; Panconesi, A. Trace Complexity of Network Inference. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013.
17. He, X.; Xu, K.; Kempe, D.; Liu, Y. Learning Influence Functions from Incomplete Observations. In Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelon, Spain, 5–10 December 2016.
18. Rozenshtein, P.; Gionis, A.; Prakash, B.; Vreeken, J. Reconstructing an Epidemic Over Time. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
19. Gripon, V.; Rabbat, M. Reconstructing a Graph from Path Traces. In Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, 7–12 July 2013.
20. Amin, K.; Heidari, H.; Kearns, M. Learning from Contagion (Without Timestamps). In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014.
21. Zhu, K.; Ying, L. Information source detection in networks: Possibility and impossibility result. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2017.
22. Zhu, K.; Chen, Z.; Ying, L. Catch'Em All: Locating Multiple Diffusion Sources in Networks with Partial Observations. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), San Francisco, CA, USA, 4–9 February 2017.
23. Zhu, K.; Chen, Z.; Ying, L. Locating the contagion source in networks with partial timestamps. *Data Min. Knowl. Discov.* **2016**, *30*, 1217–1248. [CrossRef]
24. Fanti, G.; Kairouz, P.; Oh, S.; Ramchandran, K.; Viswanath, P. Metadata-Conscious Anonymous Messaging. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016.
25. Liu, X.; Fu, L.; Jiang, B.; Lin, X.; Wang, X. Information Source Detection with Limited Time Knowledge. In Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Catania, Italy, 2–5 July 2019.
26. Tang, W.; Ji, F.; Tay, W. Estimating Infection Sources in Networks Using Partial Timestamps. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 3035–3049. [CrossRef]
27. Kumar, A.; Borkar, V.S.; Karamchandani, N. Temporally Agnostic Rumor-Source Detection. *IEEE Trans. Signal Inf. Process. Netw.* **2017**, *3*, 316–329. [CrossRef]
28. Hartigan, J.A.; Wong, M.A. Algorithm as 136: A k-means clustering algorithm. *J. R. Stat. Soc. Ser. C Appl. Stat.* **1979**, *28*, 100–108. [CrossRef]