IEEE Access
Multidisciplinary Rapid Review Open Access Journal

Received March 21, 2021, accepted May 13, 2021, date of publication May 17, 2021, date of current version May 25, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3081574

Identification of Individual Infection Over Networks With Limit Observation: Random vs. Epidemic?

JAEYOUNG CHOI, (Member, IEEE)
School of Computing, Gachon University, Seongnam 13120, South Korea
e-mail: jychoi19@gachon.ac.kr

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant 2019R1G1A1099466, and in part by Gachon University research fund of 2019 under Grant GCU-2019-0777.

ABSTRACT Popular information or dangerous viruses have recently been observed to spread rapidly through a highly connected network structure. For example, malicious viruses and rumors are spreading rapidly through online social network platforms over the Internet and diseases with high transmission power are rapidly spreading through human contact. Apart from these, some infections may occur individually regardless of the effect of the network such as computer failure. In the context of these infections coexisting, it is one of crucial problem to distinguish whether the infection with externally similar symptoms is caused by a cascade or by itself because if the infection is the cascade that is spreading through the network, it is necessary to stop this spread as soon as possible. In this paper, we study this classification problem to determine whether it is infected from a cascade or randomly when the infection snapshot is partially given. We propose two approaches for the problem (*i*) Neighbor-based approach as the use of local infection status information and (*ii*) Source-based approach as the global infection status information. The first one is that we just count the number of connected infection paths from L-hop distance to an infected node by using some criteria whereas, the second one is that we use the information of the location of cascade sources to infer the infection cause of each infected node by computing the infection probability from the sources. We perform various simulations to obtain the classification performance of the two proposed algorithms. As a result, the method of estimating a global cascade source shows better performance than that of the former one, which uses only the infection information of local neighboring nodes if the sampling rate of cascade infections is sufficient.

INDEX TERMS Cascade infection, random infection, source estimation, inference algorithm.

I. INTRODUCTION

Recently, as a network connectivity rapidly increases online or offline, information or viruses are rapidly spreading. Examples include propagation of infectious diseases through physical contact, technology diffusion, computer virus/spam infection in the Internet, and tweeting and retweeting of popular topics by popular online social network services [1]. As a result of the spread of viruses through the network, computers can break down and humans can become infected with diseases like COVID-19. However, although these infections occur on networked computers or people, they are not necessarily due to the effects of the network. Sometimes, it is not a virus given by other nodes around it, but infects itself or breaks down [2]–[5]. As a concrete example of these two different modes of“sickness”, consider a computer network that undergoes cascaded failures due to virus/worm propagation (the epidemic) vs. random failures due to misconfiguration whose stochastic behavior is external to the network itself (independent infection) [9]. The former case is usually referred to as Cascade Infection (CI), and the latter case is referred to as Random Infection (RI). For the RI, it is enough to find the node and repair or fix it, but in the CI, it is important to quickly find out where the infection started and how far it is spreading. However, if a certain node in the network is infected and the symptoms are similar,

The associate editor coordinating the review of this manuscript and approving it for publication was Donghyun Kim.

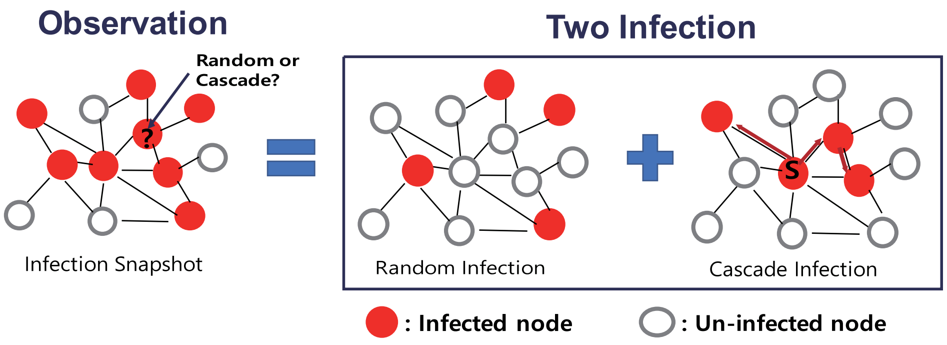


FIGURE 1. **Illustration example of our classification problem: The main objective is to classify the cause of infection for each infected node when the observable infection snapshot is given (Left). The infection comes from Random Infection (Middle) or Cascade Infection (Right) or both. (Here, S denotes the source of cascade infection).**

it is often difficult to distinguish whether the cause is proceeding from the former cascade or spontaneously generated like the latter. For example, is not easy to classify flu or COVID-19 showing similar symptoms with only information about which node in the network is infected. Hence, in this paper, we focus on the following formalized problem: For a single snapshot in time, we are informed that a given subset of nodes has a particular virus. The complete information about which nodes is infected as well as the exact sickness times among the observed nodes are not available. Given complete knowledge of the network topology, the problem is to determine if a node is infected by an epidemic, spreading through the network, or if a node is infected via an independent infection mechanism that is external to the network being considered, and not propagated through the edges of the graph.

As a prior work, the authors [9] first considered a method how to distinguish two different infection behaviors such as random infection and cascade infection in social networks. Their problem is that for a given infection snapshot with limited observation, how to determine whether the infection phenomenon is cascade or random. In other words, the assumption of the work is there exists only one kind of infection process among cascade and random over the network. To do that, they considered a simple intuition that if the sick nodes are uniformly spread out on the network, a random infection is likely at work, while if they are ''clustered'' in some appropriate sense, then it is more likely that we have an infection on our hands. However, it did not consider the cause of infection at a node level when two infection models are coexisting in the network. These considerations are more realistic and complex compared to [9] because if a node is infected where the cascade infection and the random infection can occur at the same time, it is not easy to distinguish the cause of infection at the ''node level'' where only the infection snapshot is given in the network. To the best of our knowledge, our work is the first to attempt on the identification the cause of infection at a node level under the coexisting different infection processes. To do that, we will use two information from the snapshot (*i*) Infected neighbors and (*ii*) Infection sources. The number of infected neighbors gives information on whether the infection comes from by cascade or not since the cascade infection occurs from its infected neighbors. Further, if we estimate the cascade sources in the network properly, it will be also helpful to classify the infection cause because we can obtain some infection trajectory of the cascade from the sources. Hence, in this paper, we consider the classification problem using these two information.

We summarize our main contributions in more detail as follows:

- First, we consider the method of classifying the cause of infection in the network at the individual node level under the limited observation of infection status. Especially, we consider two infection models (*i*) random infection and (*ii*) cascade infection, where the first one is the infection caused by random at the individual node level and the second one is the infection caused by the connection (or contact) over the network.
- Second, we propose two approaches for the classification problem (*i*) neighbor-based approach and (*ii*) source-based approach. In the neighbor-based approach, we count the number of connected infection paths from L-hop distance to the infected node and determine the cause of infection by considering a portion of such infection path using some predefined criteria. In the cascade source-based approach, we use the information of the location of cascade sources to infer the infection cause of each infected node.
- Third, we perform various simulations to obtain three performance measures of the proposed algorithms (*i*) cascade source detection (*ii*) accuracy of recovering the hidden cascade infection, and (*iii*) accuracy of the infection classification. As a result, the method of estimating a global cascade source and comparing the infection probability for each node shows better classification performance than the former one, which uses only the infection information of local neighboring nodes if there is sufficient sampling of the cascade infection.

The remainder of this paper is organized as follows. Section II discusses related literature. In section III, we will introduce two infection models and our classification goal. In section IV and V, we propose two classification algorithms such as the neighbor-based one and source-based one, respectively. In section VI, we depict the simulation results and conclude the paper in section VII.

II. RELATED WORK

In this section, we will divide the related researches into following two categories: (1) Inference of infection (node inference and model inference) and (2) Cascade source detection.

A. INFERENCE OF INFECTION

Research on the infection inference has been steadily progressing from the past to the present. The infection inference means determining whether a node in the network is infected or not or estimating the existence with the behavior of some random infection or cascade over the network. We denote the former one by infection node inference and the latter one by infection model inference (ex, random infection or cascade infection). As the infection node inference, there have been several studies named fault detection in the communication network system. Huang *et al.* [2] investigated on VINI testbed, which supports choosing paths to monitor, detecting and confirming the existence of a failure, correlating multiple independent observations into a single failure event. To do this, they adopted existing binary networking tomography methods to extract failures. Park *et al.* [3] suggested an edge device capable of collecting, processing, storing and analyzing data is constructed by using a single-board computer and a sensor. Especially, they considered the Long Short-Term Memory (LSTM) recurrent neural networks as a fault detection model. Babaie *et al.* [4] suggested a self-diagnosing method in determining the status of nodes to reduce the effect of neighboring node's data. Tosic *et al.* [5] studied the problem of distributed sensors' failure detection in networks with a small number of defective sensors, whose measurements differ significantly from the neighbor measurements. For this, they proposed a novel distributed detection algorithm based on gossip mechanisms and on Group Testing (GT), which is used in centralized detection problems. Sundareisan *et al.* [6] studied the problem of recovering the missing infections as well as the source nodes (so-called 'culprits') of an epidemic. As a cascade diffusion, they considered Susceptible-Infected (SI) diffusion model and they showed that both these problems can be efficiently solved simultaneously by their proposed algorithm NETFILL. Sadikov *et al.* [7] solved the problem of correcting the missing data in information cascades. The problem is formalized that, for a given fraction C' of the complete cascade C, how to estimate the properties of the complete cascade C, such as its size or depth. To do this, they proposed a tree-based approach and analyzed the correcting performance. Lamprier *et al.* [8] proposed a topological recurrent generative model of a cascade, which embeds the history of diffusion in infected nodes as hidden continuous states.

Chris *et al.* [9] first considered a method how to distinguish two different infection behavior such as random infection and cascade infection in social networks. Given a complete knowledge of the network topology, they studied how to determine if the virus is an epidemic, spreading through the network, or random infection that nodes have become infected via an independent infection mechanism that is external to the network being considered, and not propagated through the edges of the graph. Lo *et al.* [10] considered a model-based fault diagnosis framework for the wireless sensors. They showed that the detection of sensor nonlinearities is equivalent to solving the largest empty rectangle (LER) problem, given a set of features extracted from an analysis of sensor outputs. Zhao *et al.* [11] investigated a prediction of both node infection and infection order without the knowledge about the underlying cascade model and the network. For this, they designed a novel model called Deep Collaborative Embedding (DCE) for information cascade prediction, which can capture not only the node structural property but also two kinds of node cascading characteristics such as cascading context and cascading affinity. Islam *et al.* [12] extended the state-of-the-art in purely data-driven cascade analysis in multiple directions. They presented a new approach to explore the diffusion dynamics of the cascade comprehensively by only leveraging the observable cascade information. Shen *et al.* [13] showed that feature selection approaches can also be used for training an efficient object detector. Especially, they introduced a Greedy Sparse Linear Discriminant Analysis (GSLDA) for computational efficiency. Ducci etal [14] suggested a novel tree-structured neural network, which we call Cascade-LSTM for detecting the hidden information.

B. CASCADE SOURCE DETECTION

The source inference problem is also very closely related to our research. As a prior work, Shah *et al.* [15], [16] first studied the single source estimation problem over a infinitely connected tree network. Especially, they derived a graphical centrality metric called *rumor centrality*, which is a simple topology-dependent metric for a given diffusion status information in a regular tree. Under the SI diffusion, they obtained a meaningful result of detecting the source using the highest rumor centrality node, named rumor center. Zhu *et al.* [17] solved this problem under the Susceptible-Infected-Removed (SIR) model and used an infection path approach to infer the source, which is a *Jordan center*. Based on their prior result, they extended the problem to the case of sparse observations [18]. Choi *et al.* [20] considered some side information for this problem, which used querying to find the source for given untruthful answers. They analytically obtain that how many queries are necessary and sufficient to achieve a required detection probability. Luo *et al.* [19] solved this problem under the SI model, in which partial observation of infection is given. They first showed that a Jordan center,

TABLE 1. Taxonomy of related works.

Classification	Random Infection	Cascade Infection
Infection Node Inference	VINI [2], LSTM [3], Petri-net [4], GT [5], This paper	NetFill [6], k-tree [7], RNN [8], DeepDiffuse [12], This paper
Infection Model Inference	Ball & Tree [9], LER [10]	Ball & Tree [9], DCE [11], DeepDiffuse [12], GSLDA [13], Cascade-LSTM [14]
Cascade Source Detection	Not applicable	NetFill [6], Rumor Center [15], [16], Jordan Center [17], [18], [19], Query [20], SFT [21], NETSLEUTH [22], OJC [23], Time-Varying [24], This paper

i.e., a node with the minimum distance to the set of observed infected nodes is still an estimator for the source node associated with the most likely infection path that with the limited observations. Zhu *et al.* [21] first considered the problem for the *Erdös-Rényi* (ER) random graph and designed a Maximum a Posterior (MAP)-based source localization algorithm, called the Short-Fat Tree (SFT) algorithm. It selects the node such that the Breadth-First Search (BFS) tree from the node has the minimum depth but the maximum number of leaf nodes. They also established some performance guarantees of SFT under the Independent Cascade (IC) model for both tree networks and the ER random graph.

As an extension of the single source detection, there have been some studies that tried to solve the problem under the multiple cascade sources are given in the network by appropriate set estimation methods. Prakash *et al.* [22] studied this problem by adopting a well-known concept, named Minimum Description Length (MDL) principle, to find the set of seed nodes and virus propagation ripple, which describes the infected graph most succinctly. They proposed an efficient and powerful algorithm NETSLEUTH to identify the sets of seed nodes given a snapshot under SI model. Zhu *et al.* [23] proposed a multiple sources localization algorithm, named Optimal-Jordan-Cover (OJC). It first extracts a subgraph using a candidate set selection algorithm that selects source candidates based on the number of observed infected nodes in their neighborhoods. They proved that OJC can locate all sources with probability one asymptotically for the ER random graph with partial observations under the general SIR diffusion model. Different from the previous static underlying network, Hu *et al.* [24] developed a general framework to find the cascade sources in time-varying networks from a small set of messenger nodes. They showed that large degree nodes derived more valuable information than small degree nodes, a result that contrasts that for static networks.

To the best of our knowledge, our paper is the first to attempt of the individual node level classification for the infection cause (random infection or cascade infection) for a given partial infection snapshot. Especially, we consider two classification algorithms such as the neighbor-based approach as the use of local infection status information and the source-based approach as the global infection status information.

III. MODEL AND GOAL

In this section, we will describe two infection models, which are focused on this paper and formulate the goal of the work. To do this, we first consider the underlying network as an undirected graph $G = (V, E)$, where V is a set of nodes with $|V| = n$ and E is the set of edges of the form (i, j) for $i, j \in V$. Each node represents an individual in human social networks or a computer host on the Internet, and each edge corresponds to a social relationship between two individuals or a physical connection between two Internet hosts [20]. Under the given network structure, we consider two infection models (i) random infection and (ii) cascade infection, which are based on how a node becomes infected in what follows.

A. MODEL: TWO INFECTION MODELS

1) RANDOM INFECTION (RI)

As a random infection model, we simply consider that the random infection for a node over a network occurs at some time epoch $t \geq 0$. More precisely, a node $v \in V$ in the network is infected with probability $q_v > 0$ at some epoch over the time horizon. That is, a susceptible node $v \in V$ can be infected by some of the external reason (no effect of the network) with probability q_v. Hence, it is independent of all other nodes as well as other cascade diffusion in the network. We let $\overline{q} := [q_v]_{v \in V}$ be a vector of RI for each node v over the network. We further consider that a RI node v reports its status of infection with probability $\theta_v^r \in [0, 1]$ and we denote $\overline{\theta}^r := [\theta_v^r]_{v \in V}$ by the vector form of each reporting probability for the RI.

2) CASCADE INFECTION (CI)

As a cascade infection model, we consider a well-known IC model, which the detailed process is described as follows. In this model, three possible states of nodes are considered: susceptible (S), active (A) and inactive (I). First, in the susceptible state, a node can be activated from the infection of one of already activated neighbors. Next, if a node in the susceptible state is activated at the previous time slot, it becomes an active node, which is in the state that activate other susceptible child nodes. The inactive state denotes a state which is once activated earlier, but unable to activate other susceptible nodes anymore. In the traditional IC model, it is natural to assume that the activated nodes are active for only one time slot, and they become inactive at the next time slot. Once a node becomes inactive, it maintains the state until the end of the cascade process. In our model, we assume that if a node u received the information or infected from one of its infected nodes at time t, the node spread its own information (or some virus) to its neighbor v with probability $p_{u,v}$ at the next time $t + 1$. For more simple expression, we also use the probability p_e, where $e \in E$ is the edge among nodes in the network. We let $\overline{p} := [p_e]_{e \in E}$ be the diffusion vector over each edge $e \in E$. We further consider that a CI node v reports its status of infection with probability $\theta_v^c \in [0, 1]$ and we denote $\overline{\theta}^c := [\theta_v^c]_{v \in V}$. We let $s^* \in V$ be the information source, which acts as a node that initiates diffusion and we denote $V_I \subset V$ by the set of infected nodes. Further, we denote G_N by the infection sub-graph, which is consists of infected nodes and edges when there are N infected nodes are observed in the graph. In this paper, we are interested in the case when G is a tree as a first step for the classification of infection cause.

We assume that the CI governs the RI, *i.e.*if a node is infected RI in a former time, and also infected by cascade diffusion over the network, the node is regarded as the cascade infection node. The reason for this assumption is that the cascade diffusion can be continued by such a node and we need to classify this node as a cascade infected node to track the cascade progress. We assume that the number of cascade sources is given as prior by $m > 0$ for some analytical tractability.

B. GOAL: ESTIMATION OF THE INFECTION CAUSE

The goal of the paper is to estimate the cause of infection (Random or Cascade) for the infected node. More precisely, for the infected node $v \in V_I$ and for the ground truth $s(v) \in \{\text{Random, Cascade}\}$, we want to design some estimator of $s(v)$, denoted by $\hat{s}(v)$ such that $\hat{s}(v) : V_I \rightarrow \{\text{Random, Cascade}\}$, which mapping the infected node $v \in V_I$ to the cause of infection $\{\text{Random, Cascade}\}$ with high accuracy.[1] We call $\hat{s}(v)$ by the classification function of $s(v)$.

Performance metric. As a performance measure for the classification of our estimation algorithm, we use the following average bit-wise accuracy:

$$P_{classify}(\hat{s}(v)) := \frac{1}{|V_I|} \sum_{v \in V_I} \mathbb{P}\Big[s(v) = \hat{s}(v)\Big], \tag{1}$$

where $|V_I|$ is the number of infected nodes in the network. To obtain this, we will propose two classification algorithms. The first one is a neighbor-based approach using the local infection status information and the second one is a source location based approach, which use global infection status information from the cascade source nodes. Then, we perform various simulations and obtain the results as varying model parameters in SectionVI.

[1]Although a node can be infected by random infection and cascade diffusion simultaneously, we classify an infected node only for one of them as "cascade infection".

IV. NEIGHBOR BASED APPROACH

In this section, we first introduce a simple approach using the observed neighbor infection state. This approach does not use both of infection models to classify the infection cause. In the following subsection, we will describe the naive algorithm, which is based L-hop ($L \geq 1$) neighbors infection status.

A. L-HOP BASED ALGORITHM

Algorithm. The key idea of L-hop based algorithm, named by LHBA(L), in Algorithm1 is that we count the number of connected infection paths from L-hop distance to the infected node by using proper criteria. Here, the L-hop infection path means a path consisting of all infected nodes with L-hop distance from the infected node. This is somewhat intuitive because the infection path may guarantee some history of the cascade diffusion over the network. To do this, we let $\mathcal{P}_L(v)$ be the set of the L-hop infection paths from the chosen node v and denote $|\mathcal{P}_L(v)|$ by the number of such paths. Further, we let $\Omega_L(v)$ be the set of all paths from the node v within L-hop distance. Denote $\rho_L(v) := |\mathcal{P}_L(v)|/|\Omega_L(v)|$ by the portion of infection paths among all paths from the node v within L-hop distance. Then, we perform the following comparing: if $\rho_L(v) > \tau_{L,v}$ *i.e.*the number of paths (infection trajectory) is greater than some threshold $\tau_{L,v}$, we decide the node v is infected by the cascade. Otherwise, we regard the infected node v comes from the random infection. As a special case, for $L = 1$, we just consider the number of infected neighbors as a criteria, which means $|N(v) \cap V_I| > \tau_{1,v}$, where $N(v)$ is the set of the neighbor of the node v, we decide the node v is infected by the cascade. As a criteria, we let $\tau_L := [\tau_{L,v}]_{v \in V_I}$ be the vector of the threshold $\tau_{L,v}$ for each $v \in V_I$. In this algorithm, we have two fundamental questions of choosing parameters as follows:

(*i*) **Parameter L: How far do we look at?.** Each node v restricts its potential ancestors the nodes within at most L hops away from v. This truncation corresponds to an effort to avoid excessive estimation which incurs not only long running time complexity but also loss of inference accuracy, considering the fact that one of the important design choices is to determine the distance from v within which (observed) nodes' infection status are considered as the cascade. We control this design choice in our algorithm by using a truncation parameter L as a knob, provided as an input of the algorithm. For example, if $L = 1$, it is just the case to determine where the node's infection status came from based on the number of infections of the neighboring node. When determining the cascade with a proper value of L, the existence of an L-hop infection path can help predict the existence of a cascade much more than just looking at neighbor nodes. However, a large value of L is used in an environment where all infections are not reported, the existence of the corresponding path will be diminished, and this may become an inappropriate criterion for estimating the cause of infection.

Algorithm 1: *L*-Hop Based Algorithm (LHBA)

```
Input: Diffusion snapshot $G_N$, Parameter $L$, Threshold
       vector $\tau_L$
Output: Two Classified Infection Set $V_{RI}$ and $V_{CI}$
Set $V_{RI} = V_{CI} = \emptyset$;
for $v \in V_I$ do
    if $\rho_L(v) > \tau_{L,v}$ then
        $V_{CI} \leftarrow V_{CI} \cup \{v\}$;
    else
        $V_{RI} \leftarrow V_{RI} \cup \{v\}$;
Return $(V_{RI}, V_{CI})$;
```

(*ii*) **Threshold vector τ_L: How do we decide?.** It is reasonable to set the different threshold for different path lengths because the expected number of hidden nodes on the path can be varied depending on the length of the path. Hence, we control this design choice in our algorithm by using a parameter τ_L as a knob, provided as an input of the algorithm. For example, in the case of $L = 1$, the value to be used for inferring the cause of the infection should be estimated based on just how infected among the total number of neighboring nodes, but if the value L increases, the threshold is not just the number of infected neighboring nodes since it is based on the number of infection paths, a different τ_L should be used.

Intuitively, the neighbor-based approach does not use any infection models and their parameters. This algorithm just uses some local snapshot information for the classification of the infection. This is simple but limit to infer the cause of infection. Hence, in the following section, we will consider a more complicated and novel approach that uses the two models and parameters properly with the inference of the diffusion source locations.

V. SOURCE LOCATION BASED APPROACH

In the cascade source-based approach, we use the information of the location of cascade sources to infer the infection cause of each infected node. To do this, we perform the following two steps in an iterative manner:

(1) **Source estimation:** First, for a given partial (limited) infection snapshot G_N, we first estimate multiple diffusion sources, which result in the diffusion under the existing the random infection.

(2) **Recovering hidden infection:** Using the estimated sources of cascade in the above step, we compute the cascade diffusion probability from the sources. Then, we recover hidden cascade infection nodes using the diffusion probability by pre-defined criteria.

We repeat the above two steps (1) and (2) until the estimated sources set is satisfied some stopping criteria. After finishing the iteration, we finally classify the infected node by using the pre-defined criteria of diffusion probability. We will describe each of these parts in the following subsections.

Algorithm 2: Modified-OJC (MOJC)

```
Input: Snapshot $G'$, Number of sources $m$, Infection
       Probability $\overline{p}$, $\overline{q}$, Threshold $\zeta$
Output: Estimated sources set $\hat{S}$
Count the number of infection clusters (connected
  infection sub-graph) in $G'$ and denote it by $c > 0$;
Set $T_l$, $(1 \le l \le c)$ by the cluster from the infection
  snapshot $G'$, where the size of cluster is less than two
  $i.e. |T_l| < 2$;
Set $W = \emptyset$;
$(g', W)$ = Candidate Set Algorithm [23]
for $v \in V_I$ do
    Compute the infection eccentricity on $g'$
        $e(S, V_I) := \max_{v \in V_I} d(v, S).$
Find the combination with the minimum infection
  eccentricity as the set of sources
        $\hat{S} = \arg \min_{S \subset W, |S| = m} e(S, V_I).$
Return $\hat{S}$;
```

A. SOURCE ESTIMATION

Modified-OJC. As a multiple sources estimator, we adopt the Optimal Jordan Cover (OJC) algorithm, which was first introduced in [23]. This approach was developed for the SIR diffusion model, which is a generalized version of the IC-model by setting the recovering rate by one. Further, the proposed approach also considered the partial infection snapshot, which is the consideration in this paper. Hence, we use this algorithm with some modification for our problem and call it by Modified-OJC. To formally describe this, we define a hop-distance between a node v and a node set S to be the minimum hop-distance between the node v and any node in S by $d(v, S) := \min_{u \in S} d(u, v)$, where $d(u, v)$ is the shortest hop-distance between two nodes u and v. Next, we define the infection eccentricity of the node set S as the maximum hop-distance from an infected node in V_I by

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

Based on this, the Modified-OJC algorithm consists of the following three steps:

(*a*) **Step 1 (Extracting Clusters):** In the extracting clusters step, we first find each infection cluster (connected infection sub-graph) by choosing an infected node uniformly at random over the infection graph. That is, if an infected node is selected, we collect all infected nodes which connected to the chosen node. Next, we choose other infected nodes uniformly at random among the infection graph except the extracted previous set of infected nodes. We repeat this procedure until there is no infected nodes in the graph. Then, we count the number of infection clusters in G' and denote it by $c > 0$. Next, we set $T_l, (1 \leq l \leq c)$ by the ordered cluster from the infection snapshot G', where the ordering is performed w.r.t. the cluster size. Then, if the number of nodes in a cluster is less than two, we remove it from the infection graph to find the cascade sources.[2]

(b) **Step 2 (Candidate Set Selection):** Next, the algorithm selects the candidate source nodes by considering the number of infected neighbors. To do this, let $\zeta > 0$ be a positive integer which is called by selection threshold. The candidate set W is the set of nodes with more than ζ observed infected neighbors. Then, we define $W' := W \cup V_I$ and let g' be a connected sub-graph of G' induced by node set W'. An induced graph is a subset of nodes of a graph with all edges whose endpoints are both in the node subset [23]. If the induced graph is not connected, we add a path between clusters by choosing closest infected nodes to form a connected g'. This is because there is unique path between two arbitrary nodes in the tree graph, which is different part in [23].

(c) **Step 3 (Jordan Cover Selection):** Finally, for any m combination of nodes in W in Step 2, we compute the infection eccentricity of the node set as defined in (2) on subgraph g', and select the combination with the minimum infection eccentricity as the set of sources. Then, the m-Jordan cover S_J is computed by

$$S_J = \arg \min_{S \subset W, |S|=m} e(S, V_I). \tag{3}$$

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

It is known that the candidate selection step includes all sources in W with a high probability and excludes nodes that are more than $t+1$ hops away from all sources by a properly chosen threshold ζ [23]. By limiting the computation on the induced subgraph g', the computational complexity is reduced significantly. The result [23] shows that under some conditions, the OJC identifies all sources with probability one asymptotically in the ER random graph, which has a locally tree structure. Based on these facts, we describe the computational complexity of the proposed Modified-OJC 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 Modified-OJC is $O(|V_I|\,(|E| + w^m))$.

Proof: First, we have $O(V_I)$ computation time to select random infected node in V_I for the Step 1. By the result in [23] and Stirling's formula, we have $O(|V_I|\,(|E| + w^m))$ and this completes the proof. ■

B. RECOVERING HIDDEN INFECTION

Recovering Hidden CI. For the estimation of cascade sources more exactly, it is necessary to recover the hidden cascade infected nodes because the observed snapshot may not give sufficient information to localize the sources. In other words, finding hidden nodes is helpful to determine which path the actual infected node has been infected, including the source estimation. To do that, in the recovering hidden infection algorithm RHCI(k), we perform the following two steps:

(a) **Step 1 (Selecting Candidates):** In this step, we first set l forests (clusters) by T_l and for a given positive integer $k > 0$, we let

$$\partial T_l^k := \{v \in V \setminus V_I | d(v, T_l) \leq k\}, \tag{4}$$

be the uninfected (including hidden infections) nodes set within k-hop from the boundary of T_l. The reason why we consider such k-hop uninfected nodes is to reduce the time complexity instead of applying all uninfected node in the graph. In the numerical section, we will show how varying k effect the result. Then, for every node $v \in \partial T_l^k$, we compute the infection probability from the m estimated sources.

(b) **Step 2 (Recovering Hidden Infection):** Next, we compute the infection probability for given sources under the IC diffusion model as follows. Let S be the set of diffusion sources in the network and let I_S be the set of infected nodes (including observed and un-observed nodes) by the cascade from S. Then our approach is to compute the probability $\mathbb{P}(v \in I_S)$. To do this, we let $\mathcal{P}(v, s)$ be the set of paths between a node v and the source node $s \in S$. Then, for a tree graph,[3] we have

$$\mathbb{P}(v \in I_S) = 1 - \prod_{s \in S} \left(1 - \prod_{e \in \mathcal{P}(v,s)} P_e\right), \tag{5}$$

where P_e is the probability that a diffusion occurs over the edge $e \in \mathcal{P}(v, s)$. For example, if the diffusion is IC-model with the successful probability is $p > 0$ then the probability in (5) becomes

$$\mathbb{P}(v \in I_S) = 1 - \prod_{s \in S} \left(1 - p^{|\mathcal{P}(v,s)|}\right), \tag{6}$$

Algorithm 3: Recovering Hidden CI (RHCI(k))

```
Input: Snapshot $G'$, Infection Probability $\overline{p}, \overline{q}$,
       Reporting vector $\overline{\theta}^c$ Threshold $\eta$, Source set $\hat{S}$,
       parameter $k$
Output: Recovered Infection Set $\hat{R}_{HI}$
Set $T_l^k$ by the uninfected nodes set within $k$-hop from
  the boundary of $T_l$ as in (4);
for $v \in \partial T_l^k \cup \hat{R}_{HI}$ do
    Compute $P(v)$ using (5);
    if $P(v) \geq \eta$ then
        $\hat{R}_{HI} \leftarrow \hat{R}_{HI} \cup \{v\}$ w.p. $1 - \theta_v^c$;
    else
        $\hat{R}_{HI} \leftarrow \hat{R}_{HI} \setminus \{v\}$ w.p. $\theta_v^c$;
Return $\hat{R}_{HI}$;
```

[2]The nodes are regarded as RI nodes in this step.

[3]It is known [15] that counting the infection paths in a general loopy graph is a NP-complete problem.

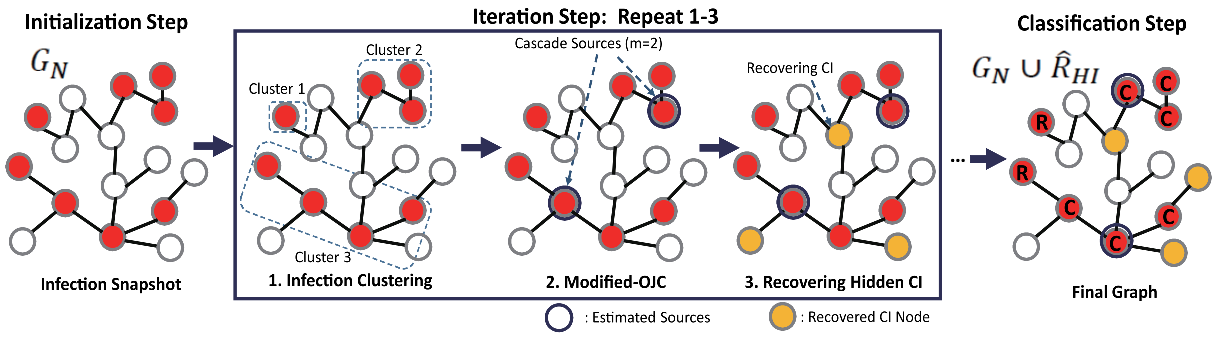


FIGURE 2. Illustration example of the Algorithm IICA(k) for $k = 1$ and $m = 2$. For a given infection snapshot G_N, the algorithm performs clustering by connected sub-infected graphs. In this example, since the number of infected nodes is less than two for the cluster 1, the modified OJC is calculated based on the induced graph by removing the cluster. Then, using the diffusion probability from the estimated sources, it recovers some hidden CI nodes (colored by yellow). The algorithm repeats this procedure until the stopping criteria is satisfied. Finally, it classifies the RI and CI for the final graph. (In the final graph, R and C indicate random infection and cascade infection, respectively).

where $|\mathcal{P}(v, s)|$ is the number of edges over the path $\mathcal{P}(v, s)$. Then, we choose the node that satisfying the diffusion probability is greater than the pre-defined threshold *i.e.*$P(v) := \mathbb{P}(v \in I_S) \geq \eta$ as a cascade infected node. Then, it remains that how to choose such threshold parameter η. As an example, it is possible to set $\eta = (P_{max}(S) + P_{min}(S))/2$, where $P_{max}(S)$ and $P_{min}(S)$ are the maximum diffusion probability and minimum diffusion probability in the infection graph G_N from the estimated source set S, respectively. In the numerical section, we will consider various value of η to see how the performance changes.

Next, we describe the computational complexity of the proposed algorithm in the following lemma.

Lemma 2: The computational complexity of the algorithm RHCI(k) is $O(mn|V_I|)$.

Proof: First, computing the infection probability (5) takes $O(m|V_I|)$ because we assumed m number of cascade sources in the graph and there exists a unique path from a source node to the node v. Further, we see that there are at most $|V_I|$ nodes for each infection path. Second, the recovering step takes $O(n)$ since $|V| = n$. Thus, the total complexity is $O(mn|V_I|)$ and this completes the proof. ■

In the following subsection, we will introduce the infection cause classification algorithm using the previous two sub-algorithms in an iterative manner.

C. CLASSIFICATION ALGORITHM FOR NODE INFECTION

Iterative Infection Classification Algorithm (IICA(k)). As a final step, we describe a classification algorithm for the cause of node infection in this subsection. The main approach is to use two previous algorithms iterative until some convergence criteria is satisfied. The pesudo-code of the algorithm will be given in Algorithm 4. For given infection snapshot G_N, two infection probability vectors $(\overline{p}, \overline{q})$, the algorithm performs the following steps:

(i) **Initialization Step:** First, the algorithm initializes the recovery hidden node set $\hat{R}_{HI}$ by empty set and the induced graph G' by the initial snapshot graph G_N. Next, it also initializes the estimated source set S^0 before the iteration by the empty set. Then, we let ΔS be the difference between estimated source sets for consecutive iterations from Modified-OJC. The algorithm initializes the cardinality of ΔS by $m + 1$.

(ii) **Iteration Step:** In the iterative step, the algorithm is performed as follow: First, for given infection snapshot and model parameters $(G', \overline{p}, \overline{q}, m, \zeta)$, it estimates multiple diffusion sources S_1 using Modified-OJC. Next, it computes the infection probability p_v of each node $v \in I_n$ by cascade from the sources. Using this, the algorithm next recovers the hidden infected node in the sub-algorithm RHCI(k) with $(G', \overline{p}, \overline{q}, \eta, \overline{\theta}^c, \hat{S})$ among the uninfected nodes. During the iterative step, if there is no recovered

Algorithm 4: Iterative Infection Classification Algorithm (IICA(k))

```
Input: $(G_N, \overline{p}, \overline{q}, \eta, \zeta, m, k, \overline{\theta}^c, \overline{\theta}^r)$
Output: Classified infection set $V_{RI}$ and $V_{CI}$ and
        estimated source set $\hat{S}$ and recovered cascade
        set $\hat{R}_{HI}$
Set $\hat{R}_{HI} = S^0 = \emptyset$ and $G' = G_N$;
Set $i = 0$ and $|\Delta S| = m + 1$;
while $|\Delta S|$ decreases do
    $G' \leftarrow G' \cup \hat{R}_{HI}$;
    $\hat{S}$ = Modified-OJC $(G', \overline{p}, \overline{q}, m, \zeta)$;
    $S^{i+1} \leftarrow \hat{S}$ and $\Delta S \leftarrow S^{i+1} \setminus S^i$;
    $\hat{R}_{HI}$ = RHCI $(G', \overline{p}, \overline{q}, \eta, \overline{\theta}^c, \hat{S})$;
    $i \leftarrow i + 1$;
Set $V_{RI} = V_{CI} = \emptyset$;
Compute $P(v)$ using the equation (6); for $v \in V_I$ do
    if $P(v) > q_v$ then
        $V_{CI} \leftarrow V_{CI} \cup \{v\}$;
    else
        $V_{RI} \leftarrow V_{RI} \cup \{v\}$;
Return $(V_{RI}, V_{CI}, \hat{S}, \hat{R}_{HI})$;
```

infected nodes, the output will be the empty set. We repeat this procedure while the number of the difference of estimating source set $|\Delta S|$ is decreased.

(*iii*) **Classification Step:** Finally, in the classification step, we compare the obtained probability of cascade infection in (5) to the random infection probability q_v for each infected node $v \in I_n$. If $p_v > q_v$, we classify it as a cascade infected node and otherwise, we regard the node v as the random infection node. Consequently, the output of the algorithm is the division of two infected nodes set R_I (random infection) and C_I (cascade infection), respectively.

We describe the computational complexity of the proposed algorithm in the following theorem.

Theorem 1: The computational complexity of the IICA(k) is $O(m|V_I|\,(|E| + w^m))$.

Proof: From the stopping criteria of the algorithm and the assumption of m source nodes in the graph, we see that there are at most m iterations in the algorithm. Using the fact that $|E| \geq n - 1$ for the connected tree structure and two previous lemmas, we obtain the result. ■

VI. SIMULATION RESULTS

In this section, we will provide simulation results of our proposed algorithms over general tree structure. For the tree construction, we use a Galton-Watson branch process, where the offspring distribution follows a binomial distribution in [0, 10] and generate nodes up to 1000. We set the cascade diffusion probability by uniformly in the range [0.5, 0.9] and the random infection probability by uniformly in [0.1, 0.3], respectively. We also set the cascade infection reporting probability by uniformly in the range [0.6, 1] and the random infection reporting probability by uniformly in [0.6, 0.9], respectively. For the hidden cascade infection recovery, we set $\eta = (P_{max}(S) + P_{min}(S))/2$, where $P_{max}(S)$ and $P_{min}(S)$ are the maximum diffusion probability and minimum diffusion probability of the observed infection graph G_N with the estimated source set S, respectively and we consider three cascade sources $m = 3$. We evaluate the performance for different infection size x. Under the assumption of two infection model (random infection and cascade IC model), it is not easy to obtain the diffusion snapshots for a fixed x infected nodes. Therefore, as done in [21] for each infection size x, we generate the infection samples where the number of infected nodes are in the range $[0.75x, 1.25x]$. The nodes in RI and the sources for the CI were chosen uniformly at random among all nodes in the network. We vary x from 40 to 400 with a step size 40. Especially, we obtain the mean size of RI infection from 10 and 100 and the cascade infection from 30 to 300, respectively.

A. PERFORMANCE METRICS

For the numerical results of the algorithm IICA(k), we use the following three performance measures:

(a) **Cascade Source Detection:** For the source estimation performance, we use the multiple source detection probability, which is defined [23] by:

$$P_{detection}(S) := \frac{|S^* \cap \hat{S}|}{m}. \tag{7}$$

This metric measures the rate that how many estimated source nodes are exactly the true sources. In addition to this, we also use the error distance [23], that is:

$$d_{error}(S) := \min_{\mathcal{P}\in\Omega(\hat{S})} \sum_{i=1}^{m} \frac{d(s_i, \hat{s}_i)}{m}, \tag{8}$$

where $d(s_i, \hat{s}_i)$ is a distance between true source s_i and estimated source $\hat{s}_i$ and $\Omega(\hat{S})$ is the set of permutations for the estimated source nodes $\mathcal{P} = (\hat{s}_1, \ldots, \hat{s}_m)$.

(b) **Accuracy of Recovering Hidden CI:** As a measure of hidden CI recovery, we use a precision-like metrics which is defined by:

$$P_{recover}(R_{HI}) := \frac{\sum_{v\in\hat{R}_{HI}} \mathbb{I}\{v \text{ is a hidden CI}\}}{|\hat{R}_{HI}|}, \tag{9}$$

where $\mathbb{I}\{\cdot\}$ is an indicator function. This metric measures the rate that how many recovered CI nodes are true infected nodes among the estimated recovered set $\hat{R}_{HI}$.

(c) **Accuracy of Infection Classification:** The infection classification error of the algorithm is given by

$$P_{classify}(V_I) := \frac{1}{|V_I|} \sum_{v\in V_I} \mathbb{I}\{s(v) = \hat{s}(v)\}. \tag{10}$$

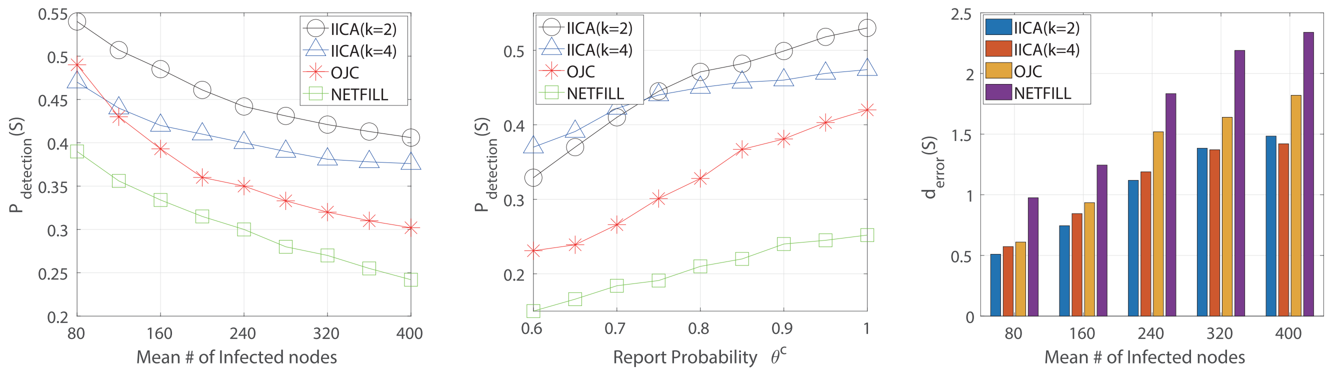
This is an experimental version of the classification performance measure (1), which is the average bit-wise accuracy.

As a comparison, we consider the algorithm OJC [23] and NETFILL [6] which including source estimation and recovering hidden CI.

B. RESULTS

1) CASCADE SOURCE DETECTION

As a first performance measure, we obtain the result of the cascade source inference for our algorithm IICA(k) by using detection probability (7) and error distance (8), respectively. For the evaluation, we set $m = 3$ and $\zeta = 3$. In Figure 3(a), we obtain the cascade source detection probabilities by increasing the mean number of infections. In our algorithm, we use the hop-distance $k = 2$ and $k = 4$ in the recovering sub-algorithm of IICA(k) and we compare our values to two source estimation algorithms such as OJC and NETFILL. In the result, we see that if the number of infections increases the detection probabilities decrease for all algorithms but the IICA($k = 2$) outperforms others. This is because the algorithm is designed to cover the hidden cascade infection as well as random infection, whereas the OJC and NETFILL only considered some parts of the environments. Further, NETFILL is designed for different diffusion model (SI-diffusion model) so that the detection probability is lower



(a) Detection vs. mean # of infections. (b) Detection vs. reporting probability θ^c. (c) Error distance vs. mean # of infections.

FIGURE 3. Source detection probabilities for $m = 3$ and $\zeta = 3$ ((a) Detection probability as varying the mean number of infected nodes (b) Detection probability as varying the reporting probability of cascade infection (c) Error distance as varying the mean number of infected nodes, respectively).

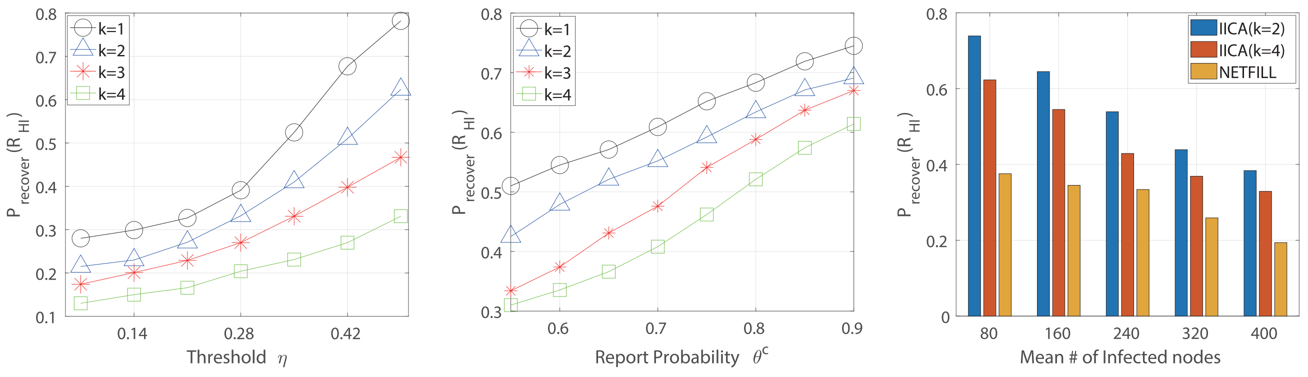
than that of others. In addition, we also check that when the number of infections is not large, OJC is better than IICA(k) for $k = 4$ but as the infection goes on, our algorithm IICA(k) is more robust for detecting the sources. This is because when the observation time is early, there may few random infections and this makes the detection of source easy for the OJC. In Figure 3(b), we obtain the detection probability as increasing the cascade infection reporting probability θ^c from 0.6 to 1. As a result, we have that when the reporting probability is high, the detection probability becomes large for all algorithms. This is because if there are many hidden cascade infection nodes, it may interfere with accurately locating the source. In the Figure 3(c), we obtain the average error distance using the performance metric (8) for the algorithms. We see that the error distance increases as the number of infections grows however, our algorithm IICA(k) for $k = 2, 4$ outperform others.

2) RECOVERING HIDDEN CI

As the performance of recovering hidden infection, we obtain its accuracy by using the performance metric in (9). We compare our IICA(k) to NETFILL, which also considers the scenario of the hidden cascade infection. In the Figure 4(a), we first obtain the recovering rate as varying the threshold η for the for cases of k ($k = 1, 2, 3, 4$). As depict in the figure, we see that when the threshold becomes increases the recovering accuracy also increases because the recovery process proceeds by comparing the diffusion probability with the corresponding threshold, and the threshold strictly sets the recovery criteria. Next, we check that when the number of k becomes larger, the recovering accuracy decreases. This is because the recovering is performed on the k hops away from the infection boundary in the infection cluster. The closer the node is to the cluster, the higher the possibility of infection. Further, if k is large, recovering is performed to a node far from the infection cluster, and this may increase the likelihood of recovering to a node where the infection has not actually occurred. In Figure 4(b), we also obtain the recovering accuracy as varying the CI reporting probability θ^c from 0.55 to 0.9. In the result, we see that if the probability increases the accuracy also increases. The reason is that the more accurately the infected node is sampled, the more accurately the cascade infection cluster is formed, and this is helpful in proceeding with hidden infected node based on the distance of this cluster. However, even in this case, it can be seen that the overall recovery performance decreases as the k increases. As a final result of recovering step, we obtain its accuracy of our algorithm IICA(k) and NETFILL as varying the mean number of infections in the Figure 4(a). In this case, we check that when the number of infections becomes large, the overall performance of recovering decreases for all algorithms. However, our IICA(k) outperforms NETFILL.

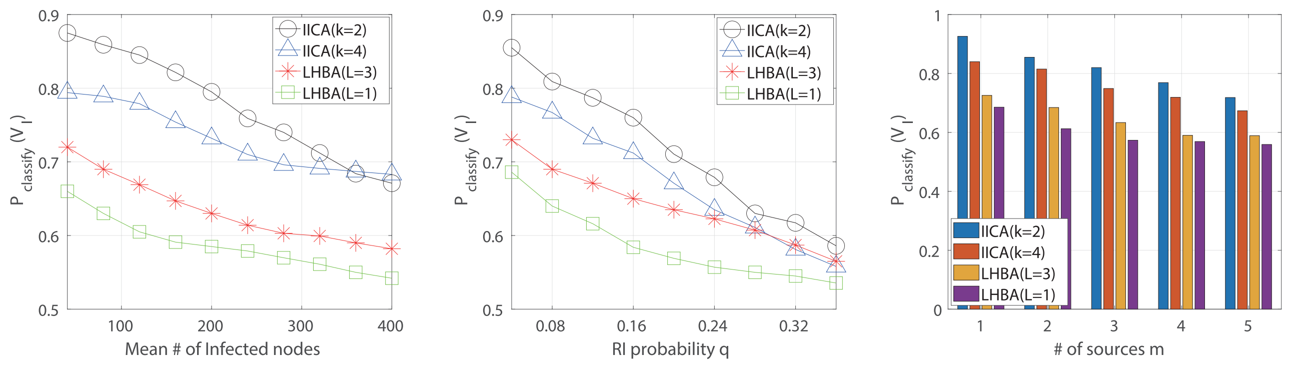
3) INFECTION CAUSE CLASSIFICATION

As a final result, we obtain the classification accuracy of two proposed algorithms by using the performance metric (10): (i) L-hop based algorithm LHBA(L) and (ii) cascade source based algorithm IICA(k). In the Figure 5(a), we first obtain the classification accuracy of IICA(k) for $k = 2, 4$ and LHBA(L) for $L = 1, 3$, respectively. In the result, we see that the classification accuracy decreases as the number of infections increases. This means when the observation time is not large, it is possible to distinguish whether the infection is caused by the cascade or random more easily because random infections are less prevalent in the network and are not mixed in the cascade infection snapshots. Further, we see that the source-based approach (IICA) is better than that of neighbor-based approach (LNBA). This is because the source-based algorithm uses the global information of cascade diffusion based on the location of sources whereas the neighbor-based algorithm only uses the infection status information of the neighbors, which is local information without any model. However, it can be seen that the performance of neighbor-based approach is not bad (above 0.6) even though it uses only the neighbor's infection information. In Figure 5(b), we obtain the result of classification accuracy as varying the random infection probability q from 0.04 to 0.36. As the result, we see that if the random



(a) Recovering vs. the threshold η. (b) Recovering vs. the reporting probability θ^c. (c) Recovering vs. mean # of infections.

FIGURE 4. **Recovering accuracy of hidden cascade infected nodes for $m = 3$. ((a) Recovering accuracy of CI as varying the decision threshold η (b) Recovering accuracy as varying the reporting probability of CI for $k = 1, 2, 3, 4$ and (c) Recovering accuracy of CI as varying the mean number of infected nodes, respectively).**



(a) Classification vs. mean # of infections. (b) Classification vs. RI probability q. (c) Classification vs. # of cascade sources m.

FIGURE 5. **Classification Results. (Classification accuracy for two proposed algorithms for (a) as varying the mean # if infections (b) as varying the RI probability $m = 3$, $\zeta = 3$ and (c) the number of CI sources, respectively).**

infection probability increases over the network, the classification accuracy decreases. This is because, in the IICA(k), if the number of randomly infected nodes in the network increases, it makes it difficult to find the source, and the accuracy of classification may decrease in the process of comparing the probability of infection from the source due to incorrectly measured diffusion probability. In the hop-based algorithm, it is only possible to judge by the degree of infection of the neighboring nodes. Hence, when the two infections are mixed, it can lead to confusion in the classification. As a third result, we obtain the accuracy when the number of cascade sources is different in the Figure 5(c). To do this, we increase the number of sources from one to five and we compare the accuracy between IICA(k) and LHBA(L). In the result, we see that if the number of sources increases the classification accuracy decreases. The reason is that as the number of sources increases, the number of infection clusters increases, and the number and shape of clusters may become more complex when all nodes do not accurately state their infection. As a final result, we obtain the accuracy performance by varying various parameters in Table 2 and Table 3 for the IICA(k) and LHBA(L), respectively. In the Table 2, we obtain the accuracy of classification of LHBA(L) by increasing the threshold parameters τ_1 and τ_3 from 0.1 to 0.4. We see that the performance of the infection path ($L = 3$) is better than that of the number of neighbors ($L = 1$). Further, we obtain that the classification accuracy does not increase when the threshold is larger than some point due to boundary infected nodes. Next, the accuracy decreases as the RI reporting probability θ^r increases because the increasing random infected nodes make confusion to estimate infection cause for LHBA(L), which uses the portion of infections of neighbors. In Table 3, we obtain the accuracy of the classification of IICA(k) by increasing the threshold parameters η and ζ. We see that when the threshold η is set properly

TABLE 2. **Classification accuracy for L-hop based algorithm LHBA(L) for three different parameters: τ_1, τ_3 and θ^r, respectively.**

$P_{classify}$	τ_1	$P_{classify}$	τ_3	$P_{classify}$	θ^r
0.513	0.10	0.581	0.10	0.681	0.60
0.534	0.15	0.593	0.15	0.678	0.65
0.552	0.20	0.628	0.20	0.653	0.70
0.588	0.25	0.643	0.25	0.631	0.75
0.613	0.30	0.621	0.30	0.598	0.80
0.604	0.35	0.616	0.35	0.573	0.85
0.592	0.40	0.597	0.40	0.542	0.90

TABLE 3. Classification accuracy for source based algorithm IICA(k) for three different parameters: η, ζ and θ^r, respectively.

$P_{classify}$	η	$P_{classify}$	$\zeta(\theta^c)$	$P_{classify}$	θ^r
0.582	0.1	0.563	2(0.6)	0.815	0.60
0.637	0.2	0.597	2(0.7)	0.807	0.65
0.713	0.3	0.631	3(0.6)	0.783	0.70
0.761	0.4	0.673	3(0.7)	0.753	0.75
0.803	0.5	0.731	3(0.8)	0.713	0.80
0.787	0.6	0.773	4(0.7)	0.673	0.85
0.742	0.7	0.818	4(0.8)	0.633	0.90

large, the classification is better since the recovering helps to the source estimation. We also see that the tight setting of threshold for the parameter ζ gives a high classification result. Especially, we consider the value of ζ for different cascade infection reporting probability θ^c since two parameters are correlated. As a last, we see that the accuracy decreases as the RI reporting probability θ^r increases because the increasing random infected nodes make confusion to estimate the cascade sources, which enables low classification performance.

VII. CONCLUSION

In this paper, we considered the problem that how to distinguish an infected node in the network, whether it is infected from a cascade or randomly when the infection snapshot is partially given. We proposed two approaches for the classification problem such as neighbor-based approach as the use of local infection status information and source-based approach as the global infection status information. In the neighbor-based one, we just count the number of connected infection paths from L-hop distance to the node by using a criteria whereas, we use the information of the location of cascade sources to infer the infection cause of each infected node for the source-based one. We have performed various simulations to obtain performance evaluations of the proposed algorithms. As a result, the method of estimating a global cascade source and comparing the infection probability for each node shows better classification performance than the former one, which uses only the infection information of local neighboring nodes when the sampling rate of cascade infections is sufficient.

REFERENCES

[1] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1852–1872, Oct. 2018.

[2] Y. Huang, N. Feamster, and R. Teixeira, "Practical issues with using network tomography for fault diagnosis," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 5, pp. 53–58, Sep. 2008.

[3] D. Park, S. Kim, Y. An, and J.-Y. Jung, "LiReD: A light-weight real-time fault detection system for edge computing using LSTM recurrent neural networks," *Sensors*, vol. 18, no. 7, p. 2110, Jun. 2018.

[4] S. Babaie, A. Khosrohosseini, and A. Khadem-Zadeh, "A new self-diagnosing approach based on Petri nets and correlation graphs for fault management in wireless sensor networks," *J. Syst. Archit.*, vol. 59, no. 8, pp. 582–600, Sep. 2013.

[5] T. Tošic, N. Thomos, and P. Frossard, "Distributed sensor failure detection in sensor networks," *Signal Process.*, vol. 93, no. 2, pp. 399–410, Feb. 2013.

[6] S. Sundareisan, J. Vreeken, and B. A. Prakash, "Hidden hazards: Finding missing nodes in large graph epidemics," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 415–423.

[7] E. Sadikov, M. J. M. Leskovec, and H. Garcia-Molina, "Correcting for Missing Data in Information Cascades," in *Proc. WSDM*, 2011, pp. 55–64.

[8] S. Lamprier, "A recurrent neural cascade-based model for continuous-time diffusion," in *Proc. ICML*, 2019, pp. 3632–3641.

[9] C. Milling, C. Caramanis, S. Mannor, and S. Shakkottai, "Network forensics: Random infection vs spreading epidemic," in *Proc. ACM SIGMETRICS*, 2012, pp. 223–234.

[10] C. Lo, J. P. Lynch, and M. Liu, "Distributed model-based nonlinear sensor fault diagnosisin wireless sensor networks," *Mech. Syst. Signal Process.*, vol. 66, pp. 470–484, Jan. 2016.

[11] Y. Zhao, N. Yang, T. Lin, and P. S. Yu, "Deep collaborative embedding for information cascade prediction," *Knowl.-Based Syst.*, vol. 193, Apr. 2020, Art. no. 105502.

[12] M. R. Islam, S. Muthiah, B. Adhikari, B. A. Prakash, and N. Ramakrishnan, "DeepDiffuse: Predicting the 'who' and 'when' in cascades," in *Proc. ICDM*, 2018, pp. 1055–1060.

[13] C. Shen, S. Paisitkriangkrai, and J. Zhang, "Efficiently learning a detection cascade with sparse eigenvectors," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 22–35, Jan. 2011.

[14] F. Ducci, M. Kraus, and S. Feuerriegel, "Cascade-LSTM: A tree-structured neural classifier for detecting misinformation cascades," in *Proc. KDD*, 2020, pp. 2666–2676.

[15] D. Shah and T. Zaman, "Detecting sources of computer viruses in networks: Theory and experiment," in *Proc. ACM SIGMETRICS*, 2010, pp. 203–214.

[16] D. Shah and T. Zaman, "Rumor centrality: A universal source detector," in *Proc. ACM SIGMETRICS*, 2012, pp. 199–210.

[17] K. Zhu and L. Ying, "Information source detection in the SIR model: A sample path based approach," in *Proc. IEEE Inf. Theory Appl. Workshop (ITA)*, Feb. 2013, pp. 1–9.

[18] K. Zhu and L. Ying, "A robust information source estimator with sparse observations," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 2211–2219.

[19] W. Luo, W. P. Tay, and M. Leng, "How to identify an infection source with limited observations," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 4, pp. 586–597, Aug. 2014.

[20] J. Choi, S. Moon, J. Woo, K. Son, J. Shin, and Y. Yi, "Information source finding in networks: Querying with budgets," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2271–2284, Oct. 2020.

[21] K. Zhu and L. Ying, "Information source detection in networks: Possibility and impossibility results," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[22] B. A. Prakash, J. Vreeken, and C. Faloutsos, "Efficiently spotting the starting points of an epidemic in a large graph," *Knowl. Inf. Syst.*, vol. 38, no. 1, pp. 35–59, Jan. 2014.

[23] K. Zhu, Z. Chen, and L. Ying, "Catch'em all: Locating multiple diffusion sources in networks with partial observations," in *Proc. AAAI*, 2017, pp. 1676–1682.

[24] Z.-L. Hu, Z. Shen, S. Cao, B. Podobnik, H. Yang, W.-X. Wang, and Y.-C. Lai, "Locating multiple diffusion sources in time varying networks from sparse observations," *Sci. Rep.*, vol. 8, no. 1, pp. 1–9, Dec. 2018.

JAEYOUNG CHOI (Member, IEEE) received the B.S. and M.S. degrees from the Department of Mathematics, Korea University, South Korea, in 2008 and 2013, respectively, and the Ph.D. degree from the Department of Electrical Engineering, KAIST, in 2018. From 2018 to 2020, he was an Assistant Professor with the Department of Automotive Engineering, Honam University, South Korea. Since 2020, he has been an Assistant Professor with the School of Computing, Gachon University, South Korea. His research interests include intersection of applied mathematics and statistical inference, including social networks, wireless vehicular networks, and graphical models.