Diffusion in Networks: Source Localization, History Reconstruction and Real-Time Network Robustification

by

Zhen Chen

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2018 by the
Graduate Supervisory Committee:

Lei Ying, Co-Chair
Hanghang Tong, Co-Chair
Jingrui He
Junshan Zhang

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

Diffusion processes in networks can be used to model many real-world processes, such as the propagation of a rumor on social networks and cascading failures on power networks. Analysis of diffusion processes in networks can help us answer important questions such as the role and the importance of each node in the network for spreading the diffusion and how to top or contain a cascading failure in the network. This dissertation consists of three parts.

In the first part, we study the problem of locating multiple diffusion sources in networks under the Susceptible-Infected-Recovered (SIR) model. Given a complete snapshot of the network, we developed a sample-path-based algorithm, named clustering and localization, and proved that for regular trees, the estimators produced by the proposed algorithm are within a constant distance from the real sources with a high probability. Then, we considered the case in which only a partial snapshot is observed and proposed a new algorithm, named Optimal-Jordan-Cover (OJC). The algorithm first extracts a subgraph using a candidate selection algorithm that selects source candidates based on the number of observed infected nodes in their neighborhoods. Then, in the extracted subgraph, OJC finds a set of nodes that "cover" all observed infected nodes with the minimum radius. The set of nodes is called the Jordan cover, and is regarded as the set of diffusion sources. We proved that OJC can locate all sources with probability one asymptotically with partial observations in the Erdős-Rényi (ER) random graph. Multiple experiments on different networks were done, which show our algorithms outperform others.

In the second part, we tackle the problem of reconstructing the diffusion history from partial observations. We formulated the diffusion history reconstruction problem as a maximum a posteriori (MAP) problem and proved the problem is NP hard. Then we proposed a step-by- step reconstruction algorithm, which can always produce a

diffusion history that is consistent with the partial observations. Our experimental results based on synthetic and real networks show that the algorithm significantly outperforms some existing methods.

In the third part, we consider the problem of improving the robustness of an interdependent network by rewiring a small number of links during a cascading attack. We formulated the problem as a Markov decision process (MDP) problem. While the problem is NP-hard, we developed an effective and efficient algorithm, *R*EAL*W*IRE, to robustify the network and to mitigate the damage during the attack. Extensive experimental results show that our algorithm outperforms other algorithms on most of the robustness metrics.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Diffusion processes in networks can be used to model many real-world phenomena including the spread of an infectious disease, the propagation of a computer virus, the gradual adoption of a new product, etc. Loosely speaking, the research on diffusion processes in networks can be categorized into two groups: prospective analysis which focuses on the structural properties of diffusion processes and networks that lead to epidemic-type outbreaks and algorithms to minimize or maximize network diffusion, and retrospective analysis which focuses on network inference such as identifying the source or underlying network of diffusion.

1.1 Overview

In this dissertation, we first tackle the *information source detection* problem, which is to infer the source(s) of an epidemic diffusion process in a network based on some observations of the diffusion. Possible observed information includes node states (e.g., infected or susceptible) and the timestamps at which nodes changed their states. The solution to this problem has a wide range of applications. In epidemiology, identifying patient zero helps diagnose the cause and the origin of the disease. For cybersecurity, tracing the source of malware is an important step in the investigation of a cyber attack. On online social networks, the trustworthiness of news/information heavily depends on its source. Given a complete snapshot of the network and motivated by the sample-path-based estimator proposed in Zhu and Ying (2013), we first present a clustering and localization algorithm for tree networks, where the number of real sources is assumed to be known. We then prove that on regular trees, the distances

between the estimators given by the algorithm and the real sources are upper bounded by a constant with a high probability. We further present a heuristic algorithm for general networks and an algorithm for estimating the number of sources when the number of sources is unknown. However, when the observation of a complete snapshot is not possible (each node can only report its state with some probability), we propose another novel algorithm, named Optimal-Jordan-Cover (OJC), for locating multiple sources for this case and prove theoretical guarantees on the detection rate for non-tree networks. We further develop a heuristic based on the $K-$means, called Approximate-Jordan-Cover (AJC), to reduce the complexity and generate a similar performance.

Then we go beyond identifying the source of diffusion and study the problem of reconstructing the entire history of a diffusion process, named as *diffusion history reconstruction*, which has been studied only very recently in Sefer and Kingsford (2014). In large-scale networks, due to the cost and privacy concerns, it is almost impossible to monitor the entire network and collect the complete diffusion trace, which makes reconstructing the diffusion history not trivial. For example, when a computer virus propagates among different computer through the network, we cannot track the infection of each computer because of the privacy limitation. And when some fake news goes viral on the Internet, which means thousands of individuals or websites are involved in the diffusion process, it is difficult to obtain the time when each individual or website that propagates the news. We assume that the diffusion process follows the Susceptible-Infected (SI) model, a variant of the popular SIR model first proposed in Kermack and McKendrick (1927), and a *single* snapshot of the network is given, which includes the set of "infected" nodes, and the corresponding infection time. The nodes with known "infection" time can be thought as monitor nodes that were placed in the network. Each monitor node can record the time at which the node is "infected" and report the infection time. We formulate the diffusion

history reconstruction problem as a maximum a posteriori (MAP) estimate problem, and prove that the problem is NP-hard by reducing an arbitrary set cover problem to a diffusion history reconstruction problem. We propose a greedy and step-by-step reconstruction algorithm to reconstruct the most likely network state at time slot τ based on the network state at time slot $\tau - 1$ while guaranteeing the state is consistent with partial observation, and further develop a greedy algorithm for a single-step construction.

Finally, we consider the problem of improving the robustness of an interdependent network under a localized attack. Interdependent networks are widely used to model and analyze many real-world complex systems, such as the Internet, social networks, transportation systems, biochemical reactions, etc Albert and Barabási (2002); Di Muro *et al.* (2016). Interdependent networks consist of nodes and links where nodes represent different components of a complex system and links characterize the interaction between the components. Many researchers have studied interdependent networks to gain insights about features and properties of complex systems, such as their robustness, stability, connectivity and structure Yuan *et al.* (2015); Vespignani (2010); Buldyrev *et al.* (2010); Gao *et al.* (2012); Di Muro *et al.* (2016); Watts and Strogatz (1998); Albert *et al.* (2000); Albert and Barabási (2002); Callaway *et al.* (2000); Albert *et al.* (1999); Newman (2010); Schneider *et al.* (2011); Zeng and Liu (2012). An important topic in this area is to preserve the robustness of interdependent networks under site or link attacks. This problem is important to many real-world networks, such as power networks, transportation networks, fuel distribution networks and communication networks. Many of these networks may have low tolerance to damages on their structures (e.g. the diameter doubled after only 5% of the most connected nodes are removed on the scale-free network Albert *et al.* (2000)). We focus on the development of algorithms that rewire the links of the

interdependent networks in real-time to minimize the impact of the localized attack. In particular, we assume an interdependent network consists of two subnetworks A and B, in which the functioning of each node can depend on a set of nodes from the other layer. We study a localized attack model such that the nodes around attacked nodes are removed hop by hop inspired by Shao *et al.* (2015). We formulate *the interdependent network link rewiring problem* as a Markov decision process (MDP) problem, which is NP-hard. Then, we propose a greedy algorithm to rewire the links during the attack.

1.2 Summary of Contributions

In Chapter 2, we consider the information source detection problem under SIR model with a complete snapshot observed. We propose an algorithm, named clustering and localization (CL), for tree networks. Then we are able to prove for a $(g+1)-$regular tree, where $g+1$ is the degree of each node on the tree, with infinite number of levels, if $gq > 1$ (q is the infection probability) and the distance between any two original sources is greater than some constant, the distance between any estimator and its closest real source is bounded by constant with a high probability. Based the CL algorithm, we develop an algorithm, called clustering and reverse infection (CRI), for general networks. Multiple experiments are done on tree networks and general networks, which shows our algorithm outperforms others.

In Chapter 3, we study the information source detection problem under a more general heterogeneous SIR model, where links have different infection probabilities and nodes have different recovery probabilities, with a partial observation. We propose a novel algorithm for locating multiple sources for such a general model and prove theoretical guarantees on the detection rate for non-tree networks. Firstly, we introduce the concept of Jordan cover, which is an extension of Jordan center.

Loosely speaking, a Jordan cover with size m is a set of m nodes that can reach all observed infected nodes with the minimum hop-distance. We propose Optimal-Jordan-Cover (OJC), which consists of two steps: OJC first selects a subset of nodes as the set of the candidates of the diffusion sources; and then it finds a Jordan cover in the subgraph induced by the candidate nodes and the observed infected nodes. We emphasize that only the hop-distance to the observed infected nodes is considered in computing a Jordan cover. Then we analyze the performance of OJC on the ER random graph, and establish the following performance guarantees. When the infection duration is shorter than $\frac{2}{3}\frac{\log n}{\mu}$, where μ is the average node degree and n is the number of nodes in the network, OJC identifies the sources with probability one asymptotically as n increases. When the infection duration is at least $\left\lceil \frac{\log n}{\log \mu + \log q} \right\rceil + 2$ where q is the minimum infection probability, *under any source location algorithm,* the detection rate diminishes to zero as n increases under the Susceptible-Infected (SI) and Independent-Cascade (IC) models, which are special cases of the SIR model. The computational complexity of OJC is polynomial in n, but exponential in m. We further propose a heuristic based on the K-Means for approximating the Jordan cover, named Approximate-Jordan-Cover (AJC), to reduce the complexity. Our simulations on random graphs and real networks demonstrate that both AJC and OJC significantly outperform other heuristic algorithms.

In Chapter 4, we investigate the problem of reconstructing the entire history of a diffusion process, named as *diffusion history reconstruction*. We assume that the diffusion process starting from one or mulitple sources follows the Susceptible-Infected (SI) model, a variant of the popular SIR model first proposed in Kermack and McKendrick (1927), and a *single* partial snapshot of the network is given, which includes the set of "infected" nodes, and the corresponding infection time of a subset of "infected" nodes. The nodes with known "infection" time can be thought as monitor

nodes that were placed in the network. Each monitor node can record the time at which the node is "infected" and report the infection time. We formulate the diffusion history reconstruction problem as a maximum a posteriori (MAP) estimate problem, and prove that the problem is NP-hard by reducing an arbitrary set cover problem to a diffusion history reconstruction problem. Then we propose a greedy and step-by-step reconstruction algorithm to reconstruct the most likely network state at time slot τ based on the network state at time slot $\tau-1$ while guaranteeing the state is consistent with partial observation, and further develop a greedy algorithm for a single-step construction. The key idea of the single-step construction algorithm is to convert the problem to the weighted set cover problem, for which a well-known greedy algorithm provides a guarantee on the approximation ratio. In this dissertation, we study multi-source diffusion processes, which include single-source diffusion as a special case. The problem is more difficult because the number of initial states is proportional to V_I^N, where V_I is the number of infected nodes and N is the number of sources. It is almost impossible to use the single-source algorithm to reconstruct the diffusion history for multi-source diffusion processes because of the high complexity. Assuming the number of sources, $N(N \geq 1)$, is known, we propose an algorithm to find all possible initial states of the diffusion to reduce the complexity. It is shown that the initial states found by our algorithm are always consistent with the partial observation. We prove that the diffusion history obtained by the step-by-step reconstruction algorithm is always consistent with the partial observation, and the computational complexity of the algorithm is $O(V_I^{N+1}E_I)$, where V_I is the number of infected nodes observed in the snapshot, E_I is the number of edges between the observed infected nodes and N is the number of sources. We evaluate the performance of the algorithm on the Western States Power Grid of the United States Watts and Strogatz (1998) and Internet autonomous systems (IAS) network Leskovec *et al.* (2005), with simulated diffusion

processes following the SI model. We also test our algorithm on the Weibo dataset (Weibo is a famous Chinese microblogging website). In all scenarios, we observe significant improvements of the proposed algorithm compared with other heuristic and existing algorithms.

In Chapter 5, we consider the problem of improving the robustness of an interdependent network under a localized attack. We focus on the development of algorithms that rewire the links of the interdependent networks in real-time to minimize the impact of the localized attack. In particular, we assume an interdependent network consists of two subnetworks A and B, in which the functioning of each node can depend on a set of nodes from the other layer. We study a localized attack model such that the nodes around attacked nodes are removed hop by hop inspired by Shao *et al.* (2015). We formulate *the interdependent network link rewiring problem* as a Markov decision process (MDP) problem, and prove that the problem is NP-hard by reducing the maximum coverage problem to our MDP problem. Then we propose a greedy algorithm to rewire the links during the attack. The key idea of the algorithm is to maximize the objective of the MDP problem in a greedy manner. We compare the performance of our algorithm with the exact solution of the MDP problem on a small network. The results show that the performance is close in terms of the objective of the MDP problem. And finally, we evaluate the performance of the algorithm on interdependent networks formed by the real networks including the air traffic network, the IAS network and the power grid network with simulated localized attacks. In most cases, when a large fraction of nodes in the networks are attacked, our algorithm outperforms others.

1.3 Related Work

1.3.1 Diffusion on Networks

In this section, we review the related work in diffusion process on networks, which can be categorized into two parts: prospective analysis and retrospective analysis.

Prospective Analysis. Many research works in diffusion process Bikhchandani *et al.* (1992); Goldenberg *et al.* (2001a); Kempe *et al.* (2003); Leskovec *et al.* (2007); Gruhl *et al.* (2004); Richardson and Domingos (2002) have been devoted to studying the so-called epidemic threshold, that is, to determine the condition under which an epidemic will break out. While earlier works Hethcote (2000) focus on some specific types of graph structure (e.g., random graphs, power-law graphs, etc), Wang et al. Wang *et al.* (2003) and its follow-up paper by Ganesh et al. Ganesh *et al.* (2005) found that, for the flu-like SIS model, the epidemic threshold for any *arbitrary, real* graph is determined by the leading eigenvalue of the adjacency matrix of the graph. Prakash et. al. Prakash *et al.* (2011) further discovered that the leading eigenvalue (and a model-dependent constant) is the only parameter that determines the epidemic threshold for other virus propagation models. On the algorithmic side, Hayashi et al. Hayashi *et al.* (2003) derived the extinction conditions under random and targeted immunization for the SHIR model (Susceptible, Hidden, Infectious, Recovered). Tong et al. Tong *et al.* (2010) proposed an effective node immunization strategy for the SIS model by approximately minimizing the leading eigenvalue. Briesemeister et al. Briesemeister *et al.* (2003) studied the defending policy in power-law graphs. Prakash et. al. Prakash *et al.* (2010); Valler *et al.* (2011) proposed effective algorithms to perform node immunization on time-varying graphs.

Retrospective Analysis. Earlier work along this line focuses on identifying the source of diffusion Zhu and Ying (2015a); Shah and Zaman (2011, 2012); Zhu *et al.*

(2015); Zhu and Ying (2016); Zhu *et al.* (2017) and inferring the underlying network of diffusion Gomez Rodriguez *et al.* (2010); Myers and Leskovec (2010); Abrahao *et al.* (2013). An even more challenging problem is to reconstruct the diffusion history, which has been received sparse attention so far. Paper Sefer and Kingsford (2014) tried to reconstruct the history by using multiple snapshots of the network at different time under the discrete time SEIRS model and it proposed an algorithm based on submodularity with some provable performance guarantee. The information used to reconstruct the diffusion history in Sefer and Kingsford (2014) is multiple snapshots of the whole network at different time slots, which can be considered as a time domain partial information. In contrast, in this paper, we use a single snapshot with the infection time of partial infected nodes, which is a space domain partial information. A method of finding the most possible diffusion path by using the infection time information of all infected nodes is proposed in Gardner *et al.* (2014), in which the authors considered the diffusion path as a tree. In Fajardo and Gardner (2013), the authors tried to estimate the diffusion path and infection time of some nodes by using the infection time of partial infected nodes. A heuristic algorithm was proposed in Fajardo and Gardner (2013) based on the integer programming problem formulated by the authors. Besides the high complexity, the heuristic algorithm in Fajardo and Gardner (2013) involves iterations between finding the infection path and estimating infection time, while the convergence is not guaranteed. In Zong *et al.* (2012), the authors focused on inferring the diffusion path of the diffusion process based on the independent cascade model by using partial observations. A heuristic algorithm derived from minimum Steiner tree was proposed in Zong *et al.* (2012). Compared with independent cascade model, the Susceptible-Infected model used in this paper goes beyond the assumption that each infected node only has one chance to infect its neighbors. In Rozenshtein *et al.* (2016), with the assumption that shorter paths of

infection are more likely, the authors formulated the problem as a temporal Steiner tree problem, in which they developed a method to recover the diffusion flow by finding a temporal Steiner tree with minimum cost. Song *et al.* (2016) combined the topic detection with the diffusion path reconstruction for reconstructing the diffusion path for different topics on Sina Weibo. Sun et al. Sun *et al.* (2017) proposed a method called Collaborative Inference Model to infer multiple coexisting diffusion processes by using sparse observations.

1.3.2 Network Robustification

Robustness of networks has been studied in previous work. A number of papers investigated the impact of network structure on network robustness Schneider *et al.* (2011); Zeng and Liu (2012); Chan *et al.* (2014). For example, Schneider *et al.* Schneider *et al.* (2011) proposed a metric called node-robustness and developed a greedy algorithm to switch links to improve the node-robustness. Zeng and Liu Zeng and Liu (2012) later defined a related metric called link-robustness and proposed a similar greedy algorithm to improve the link-robustness. Chan *et al.* Chan *et al.* (2014) used the natural connectivity as the robustness metric and proposed algorithms to modify the network structure to maximize the natural connectivity. Another line of research is on minimizing or maximzing information diffusion process in networks Tong *et al.* (2012); Zhang *et al.* (2016). Tong *et al.* Tong *et al.* (2012) proposed algorithms to modify the leading eigenvalue by adding or removing edges to limit or facilitate the information diffusion. Zhang *et al.* Zhang *et al.* (2016) proposed to limit propagation by removing some nodes or edges at a group scale. Chan *et al.* Chan *et al.* (2015) studied the problem of identifying a robust subgraph. There is literature focusing on the analysis of the critical threshold: a threshold such that when the fraction of attacked nodes exceeds it, a gaint component does not exist Cohen *et al.*

(2000); Callaway *et al.* (2000); Yuan *et al.* (2015); Shao *et al.* (2015). Cohen *et al.* Cohen *et al.* (2000) used the percolation theory to calculate the critical threshold of scale-free networks after a random attack. Yuan *et al.* Yuan *et al.* (2015) used theoretical analysis and experimental studies to show the relation between the breadth of the degree distribution and the critical threshold. The robustness of interdependent networks has also been studied. Chen *et al.* Chen *et al.* (2017) developed a near-optimal aglorithm to identify the subset of nodes at the control layer, whose failures would lead to the maximum damage to the target layers. Buldyrev *et al.* Buldyrev *et al.* (2010) proposed a cascading failure model on interdependent networks and developed a framework on analyzing the critical threshold. There is little work on preserving the robustness of interdependent networks during the attack. Di Muro *et al.* Di Muro *et al.* (2016) proposed a method to recover a fraction of attacked nodes during the cascading attacks on interdependent networks to improve network robustness. When nodes in physical infrastructure networks are attacked, it may not be recoverable. In this paper, we focus on modifying edges of healthy nodes to limit cascading attacks and to preserve network robustness.

Chapter 2

MULTIPLE SOURCE DETECTION WITH A COMPLETE SNAPSHOT

Recently, there have been a lot of interests in the problem of detecting information sources in networks. The solutions to this problem have important applications in practice, such as identifying the sources of infectious diseases and finding the sources of leaked confidential information. Shah and Zaman analytically studied this problem under the SI model and developed the rumor centrality estimator Shah and Zaman (2010, 2011, 2012). Detecting multiple information sources using the rumor centrality estimator has been investigated in Luo and Tay (2012); Luo *et al.* (2013), and detecting a single information source with partial observations by using the rumor centrality estimator has been considered in Karamchandani and Franceschetti (2013). In Dong *et al.* (2013), the detection rate of the rumor centrality estimator when a priori distribution of the source node is given has been evaluated.

Besides the SI model, information source detection under the SIR model, in which "susceptible" nodes and "recovered" nodes cannot be distinguished, has also been studied. There are a number of scenarios where it is useful to model "recovered" nodes and assume "susceptible" nodes and "recovered" nodes are indistinguishable. For example, in a blog-network, a user may post a rumor and then subsequently remove the rumor after realizing that it is not the truth. After the post was deleted, from the data crawled from the web, which has been a common methods to collect online social network datasets, it is difficult to distinguish whether the user has never posted the rumor or posted/deleted it. Similarly, classified information may spread in a social network, but people who spread the information may refuse to admit that they know the information and have spread it. This scenario again can be modeled as "recovered"

but indistinguishable from "susceptible". Zhu and Ying developed a sample-path-based estimator in Zhu and Ying (2013) for detecting a single information source under the SIR model. They later proved that the sample path estimator remains to be an effective estimator even with sparse observations Zhu and Ying (2014). The effectiveness of the sample path estimator for the SI model with partial observations and for the SIS model have been investigated in Luo and Tay (2013b) and Luo and Tay (2013a), respectively.

In this chapter, we consider the problem of detecting multiple information sources under the SIR model. It is not uncommon to have multiple information sources. For example, confidential information can be leaked from different sources and an infectious disease can start from multiple locations. We study this multi-source detection problem under the SIR model. Motivated by the sample-path-based estimator proposed in Zhu and Ying (2013), we first present a clustering and localization algorithm for tree networks, where the number of real sources is assumed to be known. We then prove that on g-regular trees, the distances between the estimators given by the algorithm and the real sources are upper bounded by a constant with a high probability. We further present a heuristic algorithm for general networks and an algorithm for estimating the number of sources when the number of sources is unknown.

2.1 Basic Model

In this section, we introduce the SIR model and the multi-source detection problem.

2.1.1 SIR Model

The network is defined to be an undirected graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node in graph G may represent a person, a

computer or a mobile device. An edge represents a communication channel such that information can be transmitted from one node to another if there is an edge between these two nodes.

Define $s(t)$ $(i(t), r(t))$ to be the fraction of nodes that are susceptible (infected, recovered) at time t. In the classical SIR model, with the uniform contact assumption and fully mixed approximation, the dynamic of the SIR system can be expressed by the following set of differential equations:

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\beta si, \quad \frac{\mathrm{d}i}{\mathrm{d}t} = \beta si - \gamma i, \quad \frac{\mathrm{d}r}{\mathrm{d}t} = \gamma i, \tag{2.1}$$

where β is the infection rate and γ is the recovery rate.

Since the network structure is ignored in those equations, these differential equations can only be used for a rough approximation of the state of the network. Therefore, in this chapter, we consider the following multi-source Susceptible-Infected-Recovered (SIR) model for information diffusion in networks. In the SIR model, every node has three states: susceptible (S), infected (I) and recovered (R) such that:

- a susceptible node may be infected by his/her infected neighbors,
- an infected node may recover, and
- a recovered node cannot be infected again.

We consider a time slotted system. At the beginning of each time slot, a susceptible node is infected by each of its infected neighbors with probability q, and each infected node recovers with probability p. Assuming the number of infected neighbors of a susceptible node is n, the probability the node becomes infected is $1 - (1 - q)^n$. Initially, at $t = 0$, all nodes are in the susceptible state except a set of source nodes, denoted by $\mathcal{S}$.

2.1.2 Problem Description

The objective of this chapter is to locate the set of sources $\mathcal{S}$ given a snapshot of the network in which we can distinguish infected nodes from other nodes, but cannot distinguish susceptible nodes and recovered nodes. We say a node is healthy if the node is in either the susceptible state or recovered state.

Define X_v to be the state of node v in the given snapshot and $\mathbf{X} = \{X_v; v \in \mathcal{V}\}$, where

$$X_v = \begin{cases} 1, & \text{if } v \text{ is in the infected state;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Denote by $\mathcal{V}_I$ the set of observed infected nodes in the snapshot. We further assume the number of sources ($S = |\mathcal{S}|$) is known.

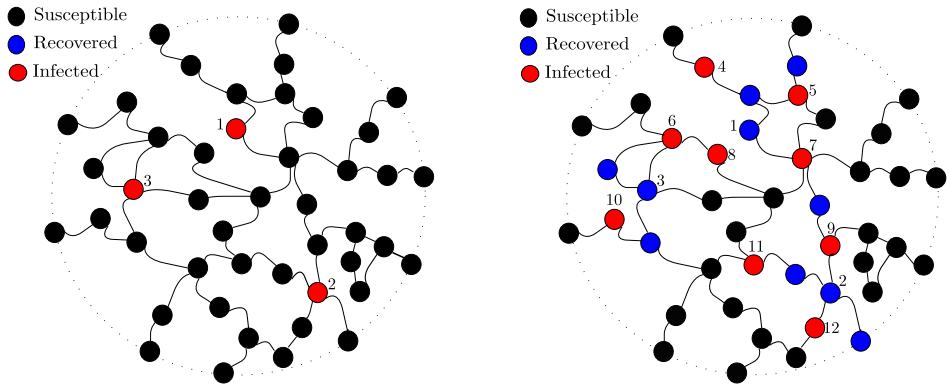
Consider Figure 2.1 as an example. Figure 2.1a is the snapshot at $t = 0$, in which there are three information sources, node 1, node 2 and node 3. When we take a snapshot at some time, the network state may look like Figure 2.1b. Define the set of nodes to be $\mathcal{V}$ and $\mathcal{V}_I = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. Since we cannot distinguish recovered nodes and susceptible nodes, the information we have is

$$\mathbf{X} = \{\forall i \in \mathcal{V}_I, X_i = 1, \text{and } \forall j \in \mathcal{V} \setminus \mathcal{V}_I, X_j = 0\}.$$

Then we need to use $\mathbf{X}$ to identify three nodes in the network as our estimators for the sources.

2.2 Main Results

We summarize the main results in this section and the notations in this chapter is in Table 2.1.



(a) Network state when time $t = 0$.

(b) Network state when the snapshot is taken.

Figure 2.1: An example of multi-source detection.

2.2.1 Multi-Source Detection on Tree Neworks

In this section, we present a multi-source detection algorithm for tree networks, named clustering and localization (CL). The algorithm is presented in Algorithm 2.1.

In the first step of Algorithm 2.1, we select a pair of infected nodes with the maximum distance because most likely these two nodes are associated with two different information sources, where we say an infected node a is "associated" with source s if node a is on the information spreading tree starting from node s. The second step of the algorithm is to select S infected nodes in a greedy fashion to maximize the pairwise distances of these S nodes. These S infected nodes are likely to be associated with different sources, and are likely to be the leaf nodes of the corresponding information spreading trees. The third step divides the set of infected nodes into S sets according to their distances to the selected S nodes. The purpose is to cluster the infected nodes according to their associated sources. The fourth step estimates the maximum distance $r_{\max}$ from a source to any observed infected node associated with the source, which can be used to approximate the depth of the information spreading

Algorithm 2.1 Clustering and Localization (CL)

1: Select two infected nodes e_1 and e_2 with the maximum distance, i.e.,

$$d(e_1, e_2) = \max_{a,b \in \mathcal{V}_I} d(a, b),$$

and let $\mathcal{B} = \{e_1, e_2\}$.

2: Let $i = |\mathcal{B}|$ and select an infected node $e_{i+1} \in \mathcal{V}_I \setminus \mathcal{B}$ such that

$$d(e_{i+1}, \mathcal{B}) = \max_{a \in \mathcal{V}_I \setminus \mathcal{B}} d(a, \mathcal{B}),$$

i.e., selecting an infected node from $\mathcal{V}_I \setminus \mathcal{B}$ that is furthest away from set $\mathcal{B}$. Here $d(a, \mathcal{B}) = \min_{u \in \mathcal{B}} d(a, u)$, where $d(a, v)$ is the distance between node a and v on graph $G(\mathcal{V}, \mathcal{E})$. Repeat this step until $|\mathcal{B}| = \min\{S, |\mathcal{V}_I|\}$.

3: Without loss of generality, assume $|\mathcal{B}| = S$. Partition the set of infected nodes into S sets: $\mathcal{V}_I^{(s)}$ for $s = 1, \cdots, S$. An infected node a is assigned to set $\mathcal{V}_I^{(s)}$ if

$$d(a, e_s) = \min_{j=1,\cdots,S} d(a, e_j).$$

Ties are broken arbitrarily.

4: For each $\mathcal{V}_I^{(s)}$, compute the infection radius

$$r_s = \left\lfloor \max_{a,b \in \mathcal{V}_I^{(s)}} \frac{d(a, b)}{2} \right\rfloor.$$

Furthermore, compute the maximum infection radius

$$r_{\max} = \max_{s=1,\ldots,S} r_s.$$

5: Consider the tree $\mathcal{T}$ formed by the set of nodes in $\mathcal{B}$ and paths between each two nodes in $\mathcal{B}$ on graph G. For each $e_i \in \mathcal{B}$, find a node γ_i on tree $\mathcal{T}$ such that $d(e_i, \gamma_i) = r_{\max}$, and add node γ_i into $\tilde{\mathcal{S}}$.

6: $\tilde{\mathcal{S}}$ is the set of source estimators.

trees. In the final step, a tree $\mathcal{T}$ with nodes in $\mathcal{B}$ as the leaf nodes is constructed; and for each $e_i \in \mathcal{B}$, we select a node $\gamma_i \in \mathcal{T}$ that is $r_{\max}$ hops away from e_i as the corresponding source estimator. Note that γ_i is close to the real source associated with infected node e_i when e_i is close to a leaf node on the corresponding information spreading tree and $r_{\max}$ is close to the depth of the tree.

Next, we present an example to illustrate how our algorithm works. Assuming

$G(\mathcal{V}, \mathcal{E})$	the tree that our detection problem is based on.
$\mathcal{V}_G$	the set of vertices of G.
$\mathcal{V}_S$	the set of actual sources.
S	$\lvert\mathcal{V}_S\rvert$, the number of original sources.
$\mathcal{V}_I$	the set of infected nodes when we take the snapshot.
$d(a, b)$	the distance between node a and b on tree G.
(a, b)	the path between node a and b, and also it represents the set of nodes on that path.
ζ_i $(i = 1, ..., S)$	The actual information sources.
γ_i $(i = 1, ..., S)$	The estimators we find.
$\mathcal{V}_\gamma$	the set of estimators.
$d_{i,j}$	$d_{i,j} = d(\zeta_i, \zeta_j)$.
d	$d = \min_{1 \le i,j \le S} d_{i,j}$.
$K^a_{b,c}$	Node satisfies $(a, c) \cap (a, b) = (a, K^a_{b,c})$.
$(a, b) \subset (c, d)$	path (a, b) is contained in path (c, d).
$a \in (c, d)$	node a is on path (c, d).
$d(a, \mathcal{N})$	$\mathcal{N}$ is a set of nodes and $\min_{b \in \mathcal{N}} d(a, b)$
t^I_a	The time that node a got infected
t^R_a	The time node a recovered

Table 2.1: Notations

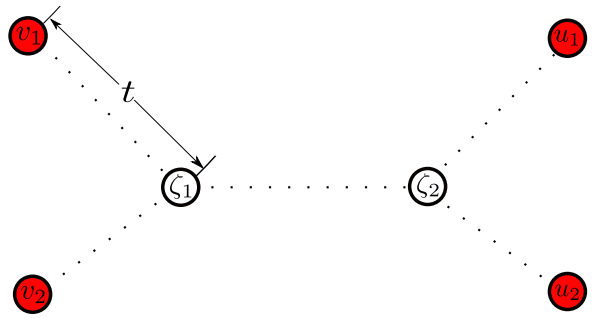


Figure 2.2: An simple example of our algorithm.

the probability of infection $q = 1$ and the probability of recovery $p = 1$, which means each infected node, right after it is infected, will infect all of its susceptible neighbors and then become recovered at the end of the time slot. Given a simple tree structure as in Figure 2.2, we use the dotted lines to represent the paths. There are two original sources, ζ_1 and ζ_2. The snapshot includes four infected nodes, v_1, v_2, u_1 and u_2, and we have $d(\zeta_1, v_1) = d(\zeta_1, v_2) = t$ and $d(\zeta_2, u_1) = d(\zeta_2, u_2) = t$. Under our algorithm, a pair of infected nodes that are farthest away from each other are selected, which could be $\{v_1, u_1\}$, $\{v_1, u_2\}$, $\{v_2, u_1\}$ and $\{v_2, u_2\}$. Without loss of generality, assume the two nodes are v_1 and u_1. Then after step 3, we have $\mathcal{V}_I^{(1)} = \{v_1, v_2\}$ and $\mathcal{V}_I^{(2)} = \{u_1, u_2\}$. Then, we have the maximum infection radius $r_{\max}$ is exactly equal to t and the tree $\mathcal{T}$ formed by the infected nodes and paths between them is the original tree $G(\mathcal{V}, \mathcal{E})$. Therefore, in step 5, the two estimators, say γ_1 and γ_2, on tree $\mathcal{T}$ that satisfy $d(v_1, \gamma_1) = t$ and $d(u_1, \gamma_2) = t$ are the sources ζ_1 and ζ_2.

2.2.2 Performance Analysis

The following theorem shows that for a g-regular tree, the distance between a detected source produced by Algorithm 2.1 and its closest real source is bounded by a constant with a high probability, where the constant is independent of the size of the infected subnetwork.

Theorem 2.1. *Consider a $(g+1)$-regular tree with infinite number of levels where $g > 2$. Assume that $gq > 1$ and the distance between any two sources is larger than C for some large enough constant C. Then given any $\epsilon > 0$, there exists a constant d_ϵ such that the distance between each estimator and its closest real source is upper bounded by d_ϵ with a probability at least $1-\epsilon$, where d_ϵ is independent of the size of the infected subnetwork.* □

The detailed proof is presented in Section A, which consists of the following key steps:

1) We define one-time-slot branching process to be an infection spreading tree such that each infected node on the tree was infected in the immediate next time slot after the infection of the node's parent. A one-time-slot branching process is a subsequence of the infection process where an infected node is included in the one-time-slot branching process if and only if it was infected at the immediate next time slot after her parent was infected. Because of that, the radius of the one-time-slot branching process increases by one in every time step until it terminates. Then for each source ζ_i, we define event $\mathcal{A}_{\zeta_i}$ which includes two cases: Case 1: the source has at least $(S+1)$ one-time-slot branching processes survived after time t_0. This means there exist $(S+1)$ survived one-time-slot branching processes whose roots are nodes that were infected before or at time slot t_0, where a one-time-slot branching process starting from an infected node is said to survive if it never dies out, which occurs with a non-zero probability. Case 2, the infection process from the source terminates at time t_0. We will prove that event $\mathcal{A} = \bigcap_i \mathcal{A}_{\zeta_i}$ occurs with a high probability.

2) The next step is to show that under event $\mathcal{A}$, each estimator produced by the algorithm is within a constant distance to its closest original source. The

analysis includes the following three cases:

(a) Infection processes from all original sources die out at time t_0.

(b) At least two sources have survived $(S+1)$ one-time-slot branching processes.

(c) Only one source have survived $(S+1)$ one-time-slot branching processes after time t_0.

In case (a), since infection processes from all original sources die out at time t_0, the maximum distance between a source and its associated infected nodes is t_0. Since any two sources are sufficiently far away from each other, the infected nodes in each set $\mathcal{V}_I^{(i)}$ are associated with the same source. Then in step 4, $r_{\max} \leq t_0$, which means the distance between each estimator found in step 5 and its closest original source is no larger than $2t_0$, which further means the distance between each estimator and its closest source is bounded by a constant.

For case (b) and case (c), the idea is to study the leaf-nodes of the survived one-time-slot branching processes. The distance between the leaf-nodes of two one-time-slot branching processes from the same source is at least $2t-2t_0$. Note that each survived source has at least $(S+1)$ one-time-slot branching processes. For simplicity, assume the nodes in set $\mathcal{B}$ after step 2 are the leaf-nodes of one-time-slot branching processes (This may not be true in general and we will discuss the general case in the proof). Then after step 3, there are at least two leaf-nodes of one-time-slot branching processes from the same source in $\mathcal{V}_I^{(i)}$, which implies $t - t_0 \leq r_{\max} \leq t$. Then the distance between the estimator and its closest survived source is equal to or smaller than $r_{\max} - (t - t_0) + t_0$, which is smaller than $2t_0$.

We can finally conclude that under event $\mathcal{A}$, the distance between each estimator and its closest original source is bounded by a constant, so Theorem 2.1 holds.

2.2.3 Heuristic for General Network Topologies

Locating multiple information sources in general networks is a much more complicated problem. Algorithm 2.1 is not directly applicable to a general network because after obtaining set $\mathcal{B}$, multiple trees can be constructed with the nodes in $\mathcal{B}$ as leaf nodes. Therefore, we propose the following heuristic algorithm, which use the Jordan infection center defined by $\mathcal{V}_I^{(s)}$ as the estimator associated with e_s.

Algorithm 2.2 Clustering and Reverse Infection (CRI)

1: Step 1 to 3 of Algorithm 2.1.
2: For $\mathcal{V}_I^{(s)}$, use the reverse infection algorithm in Zhu and Ying (2013) to find a Jordan infection center for $\mathcal{V}_I^{(s)}$, named γ_s, and then add γ_s to $\tilde{\mathcal{S}}$. A Jordan infection center γ_s for $\mathcal{V}_I^{(s)}$ is defined to be

$$\gamma_s \in \arg\min_{b\in\mathcal{V}} \left(\max_{a\in\mathcal{V}_I^{(s)}} d(b,a) \right).$$

In Algorithm 2.2, if the distance between any two original sources is sufficiently large, it is likely that the infected nodes in each set $\mathcal{V}_I^{(s)}$ are associated with the same source. The reverse infection algorithm was proposed in Zhu and Ying (2013) to localize the source in the single-source SIR model. If the infected nodes in $\mathcal{V}_I^{(s)}$ are associated with the same source, we can expect the reserve infection algorithm restricted to $\mathcal{V}_I^{(s)}$ to output a good estimator. Therefore, in Algorithm 2.2, we use the reverse infection algorithm in each set $\mathcal{V}_I^{(s)}$ to get our estimators.

2.2.4 Heuristic for Estimating S

Both Algorithms 2.1 and 2.2 require the knowledge of the number of real sources, which may be difficult to know in practice. We further propose the following heuristic for estimating the number of real sources when the number of real sources is unknown.

Algorithm 2.3 An Algorithm for Approximating S

1: Choose a large number $\bar{S}$. For each k such that $1 \leq k \leq \bar{S}$, use the steps 1-4 in Algorithm 2.1 to compute $w_k = r_{\max}$ by assuming the number of real sources is k.
2: Set

$$\tilde{S} = \arg \max_{k:1 \leq k \leq \bar{S}-2} w_k - w_{k+1} - (w_{k+1} - w_{k+2}),$$

and claim $\tilde{S}$ to be the number of real sources.

Assume the information spreading trees never die out. Consider the case where k is smaller than the number of real sources. Then after the clustering step of Algorithm 2.1, there exists a set $\mathcal{V}_I^{(s)}$ which contains infected nodes from at least two different real sources. In such a set, the maximum distance between two nodes will be significantly larger than the distance of two infected nodes that are associated with the same source, assuming the real sources are not close to each other. Then under Algorithm 2.2, we will observe a significant decrease in w_k when the value of k changes from $S-1$ to S. Based on this observation, we use k that maximizes

$$w_k - w_{k+1} - (w_{k+1} - w_{k+2})$$

as the estimator of S.

Theorem 2.1 was established assuming an infinite regular tree network, in which case, the diameter of the network is infinite and the infection process can never reach the "edge" of the network. Therefore, if the time when the snapshot is taken is large enough, we can utilize the survived one-time-slot branching processes to identify the sources. However, in reality, networks are of finite size, which means the

infection processes from different sources may hit the "edge" of the network and get completely "mixed" when the spreading time is large enough. Therefore, in practice, the algorithm may not perform well when the infection time is close to the diameter of the network. In fact, from Table 2.3 and Figure 2.6, the normalized average distance becomes larger as the decrease of the diameter of the small-world network.

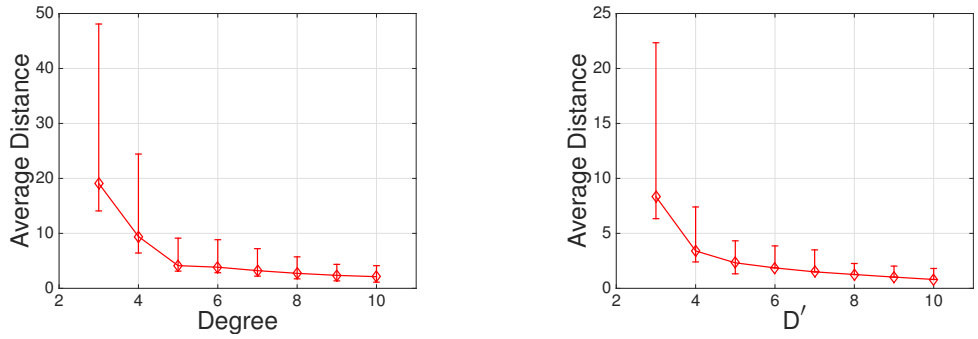
2.3 Performance Evaluation

In this section, we evaluate the performance of our algorithm using simulations.

2.3.1 Tree Networks

In this set of simulations, we assumed the number of real sources is known. We evaluated Algorithm 2.1 on g-regular trees and binomial random trees, in which the number of children of each node follows a binomial distribution with number of trials, D', and success probability β. In this simulation, we choose 4 original sources randomly and set $\beta = 0.6$, the probability of infection, q, is uniformly chosen from $(0, 0.3)$ and the probability of recovery, p, is uniformly chosen from $(0, 0.2)$.

In Figure 2.3a and 2.3b, we plotted the average distance between real sources and the estimators versus the degree of the trees. To calculate the distance between an estimator and its related source, we maintain an estimator list, which contains all estimators, and a source list, which contains all sources. Then, we select the (estimator, source) pair with the smallest distance between them among all possible pairs formed by nodes from these two lists. Then we assign the estimator to the source in the pair we have selected. Next, we remove the source and the corresponding estimator from the source list and estimator list. The previous three steps are repeated until the estimator list or the source list is empty. After that, we can use the distance between the two nodes in each selected pair to calculate the average distance.



(a) Binomial random trees.

(b) Regular trees.

Figure 2.3: The average distance from sources to the estimators with 25 and 75 percentile on tree networks.

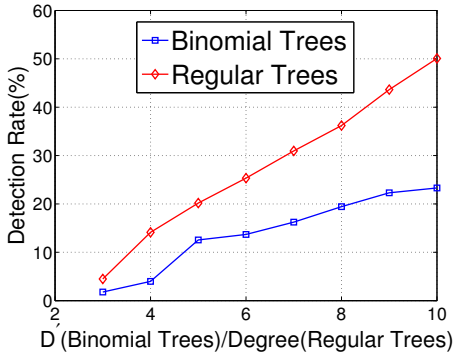


Figure 2.4: True detection rate on binomial trees and regular trees.

From Figure 2.3a and Figure 2.3b, we can see that as the degree of the tree becomes larger, the performance of our algorithm improves. For regular trees, the average distance is smaller than 3 when the degree is 5 or larger; and for binomial random trees, the average distance is smaller than 4 when D' is 5 or larger.

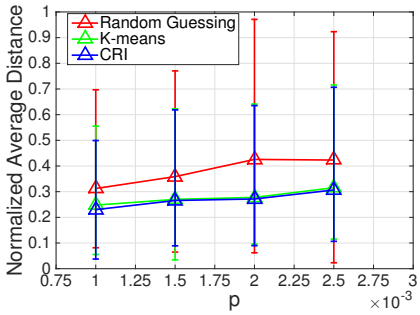
In Figure 2.4, we plotted the detection rate of Algorithm 2.1, which is the fraction of estimators being real sources. As the degree becomes bigger, detection rates of both regular trees and binomial trees improves.

2.3.2 General Networks

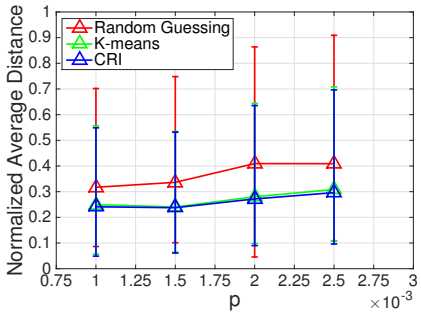
In this set of simulations, we tested the performance of our algorithm on the Erdős-Rényi (ER) model Rényi and Erdős (1959) and the small-world network model proposed in Kleinberg (2000). We compared our algorithm with random guessing and a heuristic algorithm based on k-means clustering. In k-means clustering, the initial centroids are randomly chosen. During the clustering step of each iteration in the k-means heuristic, we used distance centrality to select the centroid of each cluster. We assumed that the number of sources is unknown so we used Algorithm 2.3 to estimate the number of sources.

The ER random graph

The ER random graphs generated in this section contains 2000 nodes with wiring probability p, i.e., every pair of nodes is connected with probability p. We varied p to generate graphs with different diameters to test the algorithms. The diameters of the random graphs with different values of p are listed in Table 2.2. In Figure 2.5, we used the normalized average distance between the estimator and its associated original source, which is the average distance divided by the diameter of the network, to measure the performance. From Figure 2.5a and 2.5b, we can see that both the CRI algorithm and k-means algorithm performs much better than the random guessing algorithm, while the CRI algorithm outperforms k-means. As p becomes larger, which means the number of edges becomes larger, the normalized average distances of all these algorithms increase, which means these algorithms perform better in sparse networks than dense networks in terms of the normalized average distance.

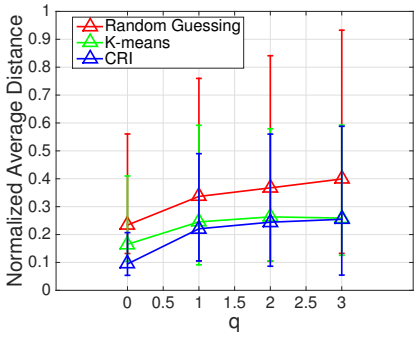


(a) The number of original sources is 4.

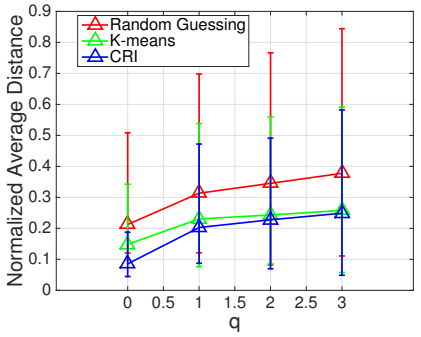


(b) The number of original sources is 5.

Figure 2.5: The average distance between estimators and original sources versus random graph parameter probability p with 25 and 75 percentile.



(a) The number of original sources is 4.



(b) The number of original sources is 5.

Figure 2.6: The average distance between estimators and original sources versus q with 25 and 75 percentile.

p	0.001	0.0015	0.002	0.0025
Diameter	26	17	11	10

Table 2.2: Diameters for random graphs with different p.

q	0	1	2	3
Diameter	98	26	19	15

Table 2.3: Diameters for the small-world networks with different q.

The small-world network

The small-world model proposed in Kleinberg (2000) is based on a two-dimensional $n \times n$ grid, where the nodes are the lattice points. There are three parameters in this model, p, q, and r. For each node in the grid, it has an edge to every other node within lattice distance p. Then for each node u, q edges between u and other nodes are constructed. For example, the i^{th} edge from u has endpoint v with probability proportional to $(d(u,v))^{-r}$. When we generated the network, we chose 50×50 grid with $p = 1$ and $r = 3$, while q was varied from 0 to 3. The diameters of these networks used in section are listed in Table 2.3. The results are shown in Figure 2.6a and 2.6b, we can see that the CRI algorithm outperforms k-means algorithm and random guessing algorithm. As q increases, the normalized average distances increase under all three algorithms, similar to those in the ER random graphs.

2.4 Conclusion

In this chapter, we studied the multi-source detection problem in the SIR model with the observation of states of nodes in the network. We provided an algorithm for tree network to detect multiple information sources when the number of sources

is known. And we also proved that with a fairly general condition, each estimator is within a constant distance to its closest original source for tree networks, which can guarantee our algorithm. Then we proposed another algorithm for general networks and a heuristic algorithm to decide the number of sources, which make our CL and CRI algorithms more general and applicable. The simulation results showed that our algorithm performs well on multi-source detection problem.

Chapter 3

MULTIPLE SOURCE DETECTION WITH PARTIAL OBSERVATION

In this chapter, we consider the information source detection problem in a setting that generalizes the existing ones at several important directions.

- *Multiple sources versus single source:* In this chapter, the diffusion can be originated from multiple nodes simultaneously, instead of from a single source. When the infection duration is sufficiently short, the infected subnetworks from different sources are disconnected components. In such cases, the single-source localization algorithms can be applied to each of the infected subnetwork. We, however, do not make such an assumption, and consider the scenario where the infected subnetworks may overlap with each other, so the single-source localization algorithms cannot be directly applied.

- *A partial snapshot versus a complete snapshot:* In this chapter, we assume a partial snapshot in which each node reports its state with some probability, which is in contrast to a complete snapshot assumed in the literature where all nodes' states are observed. Because of a partial snapshot, the sources may not report their states and be observed as infected nodes; and the observed infected nodes may not form a connected component. Both increase the uncertainty and complexity of the problem. In fact, it turns out to be critical to have a candidate selection algorithm to select source candidates from unobserved nodes but only use observed infected nodes in computing the infection eccentricity. The selection step yields 27$\times$ reduction on the computing time in our simulations while guaranteeing the same the detection rate, and yields 600$\times$ reduction on

the computing time with a slight reduction of the detection rate.

- *Heterogeneous diffusion versus homogeneous diffusion:* Our algorithm applies to the heterogeneous SIR diffusion model where links have different infection probabilities and nodes have different recovery probabilities. The asymptotic guarantees on the detection rate hold for the heterogeneous SIR model.

While some of these extensions have been investigated in the literature individually, our model includes all three extensions. We propose a novel algorithm for locating multiple sources for such a general model and prove theoretical guarantees on the detection rate for non-tree networks. The main results are summarized below.

(1) We introduce the concept of Jordan cover, which is an extension of Jordan center. Loosely speaking, a Jordan cover with size m is a set of m nodes that can reach all *observed* infected nodes with the minimum hop-distance. We propose Optimal-Jordan-Cover (OJC), which consists of two steps: OJC first selects a subset of nodes as the set of the candidates of the diffusion sources; and then it finds a Jordan cover in the subgraph induced by the candidate nodes and the observed infected nodes. We emphasize that only the hop-distance to the observed infected nodes is considered in computing a Jordan cover.

(2) We analyze the performance of OJC on the ER random graph, and establish the following performance guarantees.

   (i) When the infection duration is shorter than $\frac{2}{3}\frac{\log n}{\mu}$, where μ is the average node degree and n is the number of nodes in the network, OJC identifies the sources with probability one asymptotically as n increases.

   (ii) When the infection duration is at least $\left\lceil \frac{\log n}{\log \mu + \log q} \right\rceil + 2$ where q is the minimum infection probability, *under any source location algorithm,* the detec-

tion rate diminishes to zero as n increases under the Susceptible-Infected (SI) and Independent-Cascade (IC) models, which are special cases of the SIR model.

(3) The computational complexity of OJC is polynomial in n, but exponential in m. We further propose a heuristic based on the K-Means for approximating the Jordan cover, named Approximate-Jordan-Cover (AJC). Assuming a constant number of iterations when using the K-Means, the computational complexity of AJC is $O(nE)$, where E is the number of edges. Our simulations on random graphs and real networks demonstrate that both AJC and OJC significantly outperform other heuristic algorithms.

3.1 Problem Formulation

We assume the network is represented by an undirected graph g. Denote by $\mathcal{E}(g)$ the set of edges and $\mathcal{V}(g)$ the set of nodes in graph g. Let n denote the number of nodes and E denote the number of edges. We further assume a heterogeneous SIR model for diffusion. In this model, each node has three possible states: susceptible (S), infected (I) and recovered (R). Time is slotted. At the beginning of each time slot, each infected node (say node u) attempts to infect its neighbor (say node v) with probability q_{uv}, independently across edges. We call q_{uv} the *infection probability* of edge (u, v). At the end of each time slot, each infected node (say node u) recovers with probability r_u, independent of other infected nodes. We call r_u the *recovery probability*. We further assume $q_{uv} \in (0, 1]$ for all edges $(u, v) \in \mathcal{E}(g)$ and $r_v \in [0, 1]$ for all nodes $v \in \mathcal{V}(g)$.

Note that the SIR model includes two important special cases. When the recovery probability is zero, the SIR model becomes the Susceptible-Infected (SI) model

Bailey (1975), where infected nodes cannot recover. When the recovery probability is one, the SIR model becomes the Independent Cascade (IC) model Goldenberg *et al.* (2001b) by regarding both infected nodes and recovered nodes as active nodes and regarding the susceptible nodes as inactive nodes.

We assume the epidemic diffusion starts from m sources in the network. In other words, at time slot 0, m nodes (sources) are in the infected state and all other nodes are in the susceptible state. Denote by $s_1, s_2, \cdots, s_m$ the sources and $\mathcal{S}$ the set of sources, i.e., $\mathcal{S} = \{s_1, s_2, \cdots, s_m\}$. We assume m is *a constant independent of* n.

Finally, we assume that a *partial* snapshot of the network state at time slot t is given, with an unknown observation time t. In the snapshot, each infected or recovered node reports its state with probability $\theta_v \in (0, 1)$, independent of other nodes. If a node reports its state, we call it an observed node. Denote by $\mathcal{I}'$ the set of observed infected and recovered nodes. In this chapter, we call infected nodes and recovered nodes as "infected nodes" unless explicitly clarified.

Based on $\mathcal{I}'$, the source localization problem is to find $\mathcal{S}$ that solves the following maximum likelihood (ML) problem

$$\mathcal{W}^* = \underset{\mathcal{W} \subset \mathcal{V}(g)}{\operatorname{argmax}} \Pr(\mathcal{S} = \mathcal{W} | \mathcal{I}').$$

Even with a single diffusion source, this problem is known to be a difficult problem Shah and Zaman (2010); Zhu and Ying (2013) on non-tree networks. Therefore, instead of solving the ML problem above, we are interested in algorithms with *asymptotic perfect detection*, i.e., finding all sources with probability one as the network size increases. We believe this alternative metric is reasonable because we often need to solve the problem for large-size networks such as online social networks, and an algorithm with asymptotic perfect detection can detect sources with a high probability when the network size is large.

Algorithm 3.1: The Candidate Selection Algorithm

Input: $\mathcal{I}', g, Y$;
Output: g^- (the candidate subgraph)
Set $\mathcal{K}$ to be an empty set.
for $v \in \mathcal{V}(g)$ **do**
  if $|\mathcal{N}(v) \cap \mathcal{I}'| >= Y$ **then**
    Add v to $\mathcal{K}$, where $\mathcal{N}(v)$ is the set of neighbors of node v.
  end
end
Set $\mathcal{K}^+$ to be $\mathcal{K} \cup \mathcal{I}'$.
Set g' to be the graph induced by set $\mathcal{K}^+$.
Find all connected components in g'.
if g' *is connected* **then**
  Set $g^- = g'$.
else
  Randomly select one node in each components of g'. Denote by $\mathcal{R}$ the set of the selected nodes.
  Randomly select one node $v \in \mathcal{R}$.
  for $u \in \mathcal{R} \backslash v$ **do**
    Compute the shortest path P from v to u.
    Set $g' = g' \cup P$.
  end
  Set $g^- = g'$
end
return $g^-, \mathcal{K}$.

In this section, we present OJC and AJC based on the concept of Jordan cover.

Define the hop-distance between a node v and a *node set* $\mathcal{W}$ to be the minimum hop-distance between node v and any node in $\mathcal{W}$, i.e.,

$$d(v, \mathcal{W}) \triangleq \min_{u \in \mathcal{W}} d(v, u).$$

We then define the *infection eccentricity* of node set $\mathcal{W}$ to be the maximum hop-distance from an infected node in $\mathcal{I}'$ to set $\mathcal{W}$, i.e.,

$$e(\mathcal{W}, \mathcal{I}') = \max_{v \in \mathcal{I}'} d(v, \mathcal{W}). \tag{3.1}$$

We further define m-Jordan-cover (m-JC) to be the set $\mathcal{W}^*(\mathcal{K}, \mathcal{I}', m)$ such that

$$\mathcal{W}^*(\mathcal{K}, \mathcal{I}', m) = \underset{\mathcal{W} \in \{\mathcal{W} || \mathcal{W}| = m, \mathcal{W} \subset \mathcal{K}\}}{\operatorname{argmin}} e(\mathcal{W}, \mathcal{I}'). \tag{3.2}$$

where $\mathcal{I}'$ is the set of observed infected nodes that $m-$JC needs to cover and $\mathcal{K}$ is the *candidate set* for the sources. Therefore, m-JC is the set of m nodes in $\mathcal{K}$ with the minimum infection eccentricity.

We now introduce the *optimal Jordan cover* (OJC) algorithm whose asymptotic detection rate will be analyzed in Section 3.3.1.

The Optimal Jordan Cover (OJC) Algorithm

- **Step 1: Candidate Selection:** Let Y be a positive integer. The candidate set $\mathcal{K}$ is the set of nodes with more than Y observed infected neighbors. In addition, define $\mathcal{K}^+ \triangleq \mathcal{K} \cup \mathcal{I}'$. Denote by g^- a connected subgraph of g *induced* by node set $\mathcal{K}^+$. An induced graph is a subset of nodes of a graph with all edges whose endpoints are both in the node subset. If the induced graph is not connected, we select a random node in each component, randomly pick one selected node and add the shortest pathes from this node to all other selected nodes to form a connected g^-. We call Y the *selection threshold*. The pseudo code of the candidate selection algorithm for selecting $\mathcal{K}$ and g^- can be found in Algorithm 3.1.

- **Step 2: Jordan Cover**: For any m combination of nodes in $\mathcal{K}$ in Step 1, we compute the infection eccentricity of the node set as defined in (3.1) on subgraph g^-, and select the combination with the minimum infection eccentricity as the set of sources. Ties are broken by the total distance from the observed infected to the node set, i.e., $\sum_{v \in \mathcal{I}'} d(v, \mathcal{W})$.

With a properly chosen threshold Y, the candidate selection step includes all sources in $\mathcal{K}$ with a high probability and excludes nodes that are more than $t + 1$

hops away from all sources. By limiting the computation on the induced subgraph g^-, the computational complexity is reduced significantly. From simulations, we will see that it results in 27× reduction of the running time without affecting the detection rate. The asymptotic detection rate of OJC will be studied in Theorem 3.2. Under some conditions, OJC identifies all sources with probability one asymptotically.

OJC is a polynomial-time algorithm for given m, but the complexity increases exponentially in m. To further reduce the complexity, we propose Approximate Jordan Cover (AJC), which replaces Step 2 of OJC with the K-Means algorithm Hartigan and Wong (1979). As shown in the simulations, the performance of the AJC algorithm, in terms of both detection rate and the error distance, is close to OJC with much shorter running time. The computational complexity of both algorithms are summarized in the following theorem.

Theorem 3.1. *The computational complexity of OJC is*

$$O\left(|\mathcal{I}'|\left(|\mathcal{E}(g)| + m\binom{|\mathcal{V}(g^-)|}{m}\right)\right),$$

and the computational complexity of AJC is

$$O\left(|\mathcal{I}'|\left(|\mathcal{E}(g)| + H|\mathcal{V}(g^-)|\right)\right),$$

where H is the number of iterations used in the K-Means algorithm in AJC.

Proof. We first analyze the complexity of the OJC algorithm. For simplicity, denote by $V = |\mathcal{V}(g)|$ the number of nodes and $E = |\mathcal{E}(g)|$ the number of edges.

In the candidate selection stage, each node needs to compute its degree. Therefore, each edge is counted twice and each node is processed once. The complexity is $O(V + E)$. Counting the number of the connected components using breadth first search is of complexity $O(V + E)$. When the induced graph is not connected, the complexity to compute the shortest paths from one node to all other nodes in an

unweighted graph is of complexity $O(V+E)$. As a summary, the complexity of the candidate selection algorithm is $O(V+E)$.

The resulting subgraph has $|\mathcal{V}(g^-)|$ nodes and at most E edges. Next, we compute the complexity of the OJC.

We first compute the distances from nodes in g^- to nodes in $\mathcal{I}'$. Note this is equivalent to do a breadth-first search from each node in $\mathcal{I}'$. The complexity of the BFS is $O(|\mathcal{V}(g^-)|+E)$. Therefore, the complexity for this step is $O(|\mathcal{I}'|(|\mathcal{V}(g^-)|+E))$. The results are saved in a two dimensions hashtable so that querying the distance from one observed infected node and another node in graph g^- is of complexity $O(1)$.

After the above precomputation, for each set of nodes with size m, we want to obtain its infection eccentricity. For each observed infected node, we query the hash table to find the minimum distance to the set of nodes with size m. The complexity is $O(m)$. To compute the infection eccentricity of one set is of complexity $O(m|\mathcal{I}'|)$. In addition, there are $\binom{|\mathcal{V}(g^-)|}{m}$ possible node sets. Therefore, the complexity is $O((m|\mathcal{I}'|)\binom{|\mathcal{V}(g^-)|}{m})$

As a summary, the complexity of the OJC algorithm is

$$
\begin{aligned}
&O\left(V+E+|\mathcal{I}'|(|\mathcal{V}(g^-)|+E)+m|\mathcal{I}'|\binom{|\mathcal{V}(g^-)|}{m}\right)\\
=&O\left(|\mathcal{I}'|\left(E+m\binom{|\mathcal{V}(g^-)|}{m}\right)\right)
\end{aligned}
$$

Next, we analyze the complexity of the AJC algorithm. Note the complexity of the candidate selection and the precomputation are the same. The complexity of the Kmeans algorithm is analyzed as follows.

The complexity of the membership assignment phase is $O(m|\mathcal{I}'|)$. For each observed infected node, we only need to query its distance to the preselected m sources and each query is of complexity $O(1)$ based on the results of the precomputation.

For the center update phase, for each cluster, we need to search all $|\mathcal{V}(g^-)|$ nodes to find a center. Therefore, the complexity is $|\mathcal{V}(g^-)||\mathcal{I}'|$.

As a summary, the complexity of one iteration of the Kmeans algorithm is

$$O\left(\left(m+|\mathcal{V}(g^-)|\right)|\mathcal{I}'|\right)$$

Denote by H the number of iterations, the complexity of the AJC algorithm is

$$\begin{aligned}&O\left(V+E+|\mathcal{I}'|(|\mathcal{V}(g^-)|+E)+N\left(\left(m+|\mathcal{V}(g^-)|\right)|\mathcal{I}'|\right)\right)\\ =&O\left(|\mathcal{I}'|\left(E+H|\mathcal{V}(g^-)|\right)\right)\end{aligned}$$

□

3.3 Asymptotic Analysis of OJC

In this section, we present the asymptotic analysis of the detection rate of OJC on the ER random graph. The results include the conditions that guarantee probability one detection and the conditions under which it is impossible to detect the source set with nonzero probability under any source localization algorithm.

3.3.1 Asymptotic Perfect Detection on the ER Random Graph

We first present the positive result that shows that on the ER random graph, OJC identifies the m sources with probability one asymptotically under some conditions. Recall that n is the number of nodes in the graph, and m is the number of sources, which is a constant independent of n. Denote by p the wiring probability of the ER random graph, which is the probability that there exists a link between two nodes. Let $\mu = np$, which is the average node degree. Define $q \triangleq \min_{u,v\in\mathcal{V}(g)} q_{u,v}$, i.e., the minimum infection probability over all edges and $\theta \triangleq \min_{v\in\mathcal{V}(g)} \theta_v$ i.e., the minimum report probability over all nodes.

Theorem 3.2. *OJC identifies all m sources with probability one as $n \to \infty$ when the following conditions hold:*

(c1): $\mu q\theta = \Omega(\log n)$, [1]

(c2): $\limsup_{n\to\infty} \frac{Y}{\mu q\theta} < 1$ *and* $\liminf_{n\to\infty} \frac{Y}{\mu q\theta} > 0$, *and*

(c3): $t = \omega(D)$ *and* $\limsup_{n\to\infty} \frac{t}{\frac{\log n}{\log \mu}} < \frac{2}{3}$.

Proof. In this proof, we will show that one source $s_i \in \mathcal{W}^*(\mathcal{K}, \mathcal{I}', m)$ with a high probability. Then with a union bound, we will show that

$$\mathcal{S} = \mathcal{W}^*(\mathcal{K}, \mathcal{I}', m).$$

with a high probability for sufficiently large n.

Recall that we regard the recovered nodes and infected nodes as "infected" since the recovery process is not related to the proof. Without loss of generality, we consider s_1. Throughout the proof, we consider the BFS tree rooted at s_1. In particular, the level of one node means the level of the node on the BFS tree rooted at s_1.

We first introduce and recall some necessary notations terms.

For an ER random graph g.

- A node v is said to be on level i if $d_{s_1 v} = i$. Denote by $\mathcal{L}_i$ the set of nodes from level 0 to level i and $l_i = |\mathcal{L}_i|$.
- Denote by L'_i the set of nodes on level i. In addition, l'_i is the number of nodes on level i.
- The descendants of node v in a tree are all the nodes in the subtree rooted at v. In addition, v is the ancestor of all its descendants.
- The offsprings of a node on level k (say v) are the nodes which are on level $k+1$ and have edges to v. Denote by $\Phi(v)$ the offspring set of v and $\phi(v) = |\Phi(v)|$.

[1] Throughput this chapter, the asymptotic order notation is defined for $n \to \infty$.

- Denote by p the wiring probability in the ER random graph.
- Denote by n the total number of nodes.
- Denote by $\mu = np$.
- Denote by $\text{Bi}(n, p)$ the binomial distribution with n number of trials and each trial succeeds with probability p.
- Denote by $T^\dagger$ the BFS tree rooted at s_1.
- Denote by $\Phi'(v)$ the set of offsprings of node v on $T^\dagger$ and $\phi'(v) = |\Phi'(v)|$.
- Denote by g_t the subgraph induced by all nodes within t hops from s on the ER graph. The *collision edges* are the edges which are not in $T^\dagger$ but in g_t, i.e., $e \in \mathcal{E}(g_t) \backslash \mathcal{E}(T^\dagger)$.
- A node who is an end node of a collision edge is called a *collision node*. Denote by $\mathcal{R}_k$ the set of collision edges whose end nodes are within level k and $R_k = |\mathcal{R}_k|$.
- Denote by $\mathcal{Z}_i$ the set of nodes which are infected at time i.
- Denote by $\tilde{\mathcal{Z}}_j^i(v)$ the set of nodes that are infected at time slot i, on level j, when s_1 is the only infection source in the graph and the descendants of node v in the BFS tree rooted at s_1 and $\tilde{\mathcal{Z}}_j'^i(v)$ are the observed infected nodes in $\tilde{\mathcal{Z}}_j^i(v)$. $\tilde{Z}_j^i(v)$ and $\tilde{Z}_j'^i(v)$ are defined as the cardinality respectively.
- Denote by $\psi(v)$ the number of observed infected neighbors of node v.
- Denote by $\psi'(v)$ the number of infected offsprings of node v (the offspring is defined based on the BFS tree rooted at node s_1).
- Denote by $\psi''(v)$ the number of *observed* infected offsprings of node v.

To prove $s_1 \in \mathcal{W}^*(\mathcal{K}, \mathcal{I}', m)$, we need to show that any set $\mathcal{W}$ such that $s_1 \notin \mathcal{W}$ has a infection eccentricity larger than t on g^-. We need the following asymptotic high probability events.

- **Offsprings of each node.** Consider the BFS tree rooted at source s_1. Define

$$E_1 = \{\forall v \in \mathcal{L}_{t+D}, \phi'(v) \in ((1-\delta)\mu, (1+\delta)\mu)\}.$$

  E_1, when occurs, provides upper and lower bounds for the number of offsprings of each node in $\mathcal{L}_{t+D}$.

- **Total number collision edges.** Consider the BFS tree rooted at source s_1. We define event E_2 when the following upper bound on the collision edges holds

$$R_j \begin{cases} = 0 & \text{if } 0 < j \leq \lfloor m^- \rfloor, \\ \leq 8\mu & \text{if } \lfloor m^- \rfloor < j < \lceil m^+ \rceil, \\ \leq \frac{4[(1+\delta)\mu]^{2j+1}}{n} & \text{if } \lceil m^+ \rceil \leq j \leq \frac{\log n}{(1+\alpha)\log \mu}. \end{cases}$$

  where $m^+ = \frac{\log n}{2\log[(1+\delta)\mu]}$ and $m^- = \frac{\log n - 2\log\mu - \log 8}{2\log[(1+\delta)\mu]}$. E_2 provides the upper bounds for collision edges at different levels. Note that a subgraph with diameter $\leq m^-$ is a tree with high probability since there is no collision edges.

- **Detailed collision edges in level $> D + t$.** Define E_3 to be the event that

$$\forall v \in \mathcal{L}'_{D+t+1}, \quad \psi(v) < (1-\delta)^3 \mu q \theta.$$

- **Infected nodes.** Define

$$E_4 = \{\forall v \in \cup_{i=0}^{t-1} \mathcal{Z}_i, \quad \psi'(v) > (1-\delta)^2 \mu q.\}$$

  and

$$E_5 = \{\forall v \in \cup_{i=0}^{t-1} \mathcal{Z}_i, \quad \psi''(v) > (1-\delta)^3 \mu q \theta.\}$$

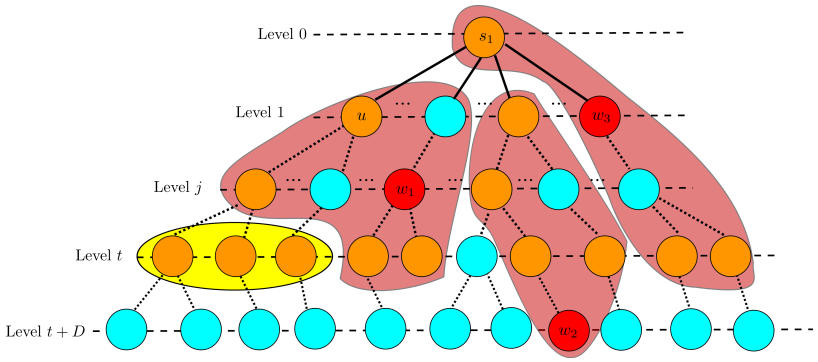


Figure 3.1: A pictorial example for Theorem 2.

- **Infected nodes from source s_1.** Define

$$E_6 = \{\tilde{Z}_1^1 \geq (1-\delta)^2\mu q\} \cap \{\forall v \in \tilde{Z}_1^1, \tilde{Z}_t'^t(v) \geq [(1-\delta)^2\mu q]^{t-1}(1-\delta)\theta\}$$

To prove these event happens with a high probability, we have

$$\begin{aligned}
&\Pr(E_1 \cap E_2 \cap E_3 \cap E_4 \cap E_5 \cap E_6) \\
\geq &\Pr(E_1)(1 - \Pr(\bar{E}_2|E_1) - \Pr(\bar{E}_3|E_1) - \Pr(\bar{E}_6|E_1)) - \Pr(\bar{E}_4 \cup \bar{E}_5) \\
= &\Pr(E_1)(1 - \Pr(\bar{E}_2|E_1) - \Pr(\bar{E}_3|E_1) - \Pr(\bar{E}_6|E_1)) - (1 - \Pr(E_4 \cap E_5)) \\
\geq &1 - \epsilon,
\end{aligned}$$

for sufficiently large n. Based on Lemma B.1, B.2, B.4 and B.5, we have events E_1, E_2, E_3, E_4, E_5, E_6 happen with a high probability as n is large enough. The proofs can be found in the Appendices.

We summarize the important properties of the OJC algorithm based on the above asymptotic events.

(a) Let $Y = (1-\delta)^3\mu q\theta$. In Step 1 of the OJC algorithm, we have $\mathcal{K} \subset \mathcal{L}_{t+D}$, i.e., all nodes on the subgraph g^- are within level $t+D$ of the BFS tree rooted at source s_1. Based on E_3, all nodes on level $t+D+1$ have less than Y observed

infected neighbors. In addition, all nodes one level $l > t+D+1$ have no infected neighbors since all the infected nodes are within level $t + D$.

(b) Based on event E_5, all nodes which are infected at or before time $t - 1$ have at least Y observed infected neighbors. Therefore, we have $\cup_{i=0}^{t-1}\mathcal{Z}_i \subset \mathcal{K}$ which implies $\mathcal{I} \subset \mathcal{K}^+$ and g^- is a connected graph.

Therefore, g^- is a connected graph which contains all the infected nodes and is restricted up to level $t + D$ based on E_3 and E_5.

Next we will show that $s_1 \in \mathcal{W}^*$ by contradiction. Note, we have $e(\mathcal{S}, \mathcal{I}') = t$. Therefore, $e(\mathcal{W}^*, \mathcal{I}') \leq t$. We will show that $e(\mathcal{W}, \mathcal{I}') > t$ for any set of nodes $\mathcal{W}$ where $|\mathcal{W}| = m$, $s_1 \notin \mathcal{W}$. Note, all the infection eccentricity are considered based on the subgraph g^-. Specifically, we will show that no nodes in g^- other than source s_1 can reach all nodes which are infected by source s_1 within t time slot based on events E_1, E_2, E_3, E_4, E_5 and E_6.

Consider the BFS tree rooted at source s_1. We discuss the proof in two cases.

(a) When $t+D \leq \lfloor m^- \rfloor$, according to the event E_2, the $t+D$ hops from s_1 is a tree because there is not collision edges. When event E_6 occur, there are at least $(1-\delta)^2\mu q$ infected nodes at level 1 and $\forall v \in \mathcal{Z}_1^1, Z_t''^t(v) \geq [(1-\delta)^2\mu q]^{t-1}(1-\delta)\theta$ which means there exists at least one observed infected node on level t for each subtree rooted at level 1. Consider a set $\mathcal{W}$ where $s_1 \notin \mathcal{W}$. There exists a node $u \in \tilde{\mathcal{Z}}_1^1$ such that $\mathcal{W} \cap \mathcal{V}(T_u^{-s_1}) \neq \emptyset$ where $T_u^{-s_1}$ is the tree rooted at node u without the branch of node s_1 (the subtree on level 1). Therefore, for one node $w \in \tilde{\mathcal{Z}}_t''^t(u)$, we have $d(w, \mathcal{W}) > t$. Hence for any set $\mathcal{W}$ with size m that does not contain s_1, $e(\mathcal{W}, \mathcal{I}') > t$. Therefore, we have $s_1 \in \mathcal{W}^*$ when $t + D \leq \lfloor m^- \rfloor$.

Figure 3.1 shows a pictorial example when $m = 3$. The red nodes are the nodes in set $\mathcal{W}$. The existence of node u is guaranteed since m nodes are insufficient

to cover all level 1 branches of the BFS tree rooted in s_1. The red areas are the t hop neighborhood of nodes w_1, w_2 and w_3. In this case, the $t+D$ neighborhood of node s_1 is a tree. Therefore, any set $\mathcal{W}$ does not contain s_1 can not reach the yellow area $\tilde{\mathcal{Z}}_t''^t(u)$ within t hops as shown in the figure.

(b) Consider the case when $t+D > \lfloor m^- \rfloor$. Again, based on event E_6, there exists a node $u \in \tilde{\mathcal{Z}}_1^1$ such that $\mathcal{W} \cap \mathcal{V}(T_u^{-s_1}) \neq \emptyset$. The distance between a node in $\mathcal{Z}_t'^t(u)$ and set $\mathcal{W}$ on the BFS tree is larger than t. Therefore, if $\mathcal{W}$ is a Jordan infection cover, the shortest paths between $\mathcal{Z}_t'^t(u)$ and set $\mathcal{W}$ must contain at least one collision nodes. In the rest of the proof, we will show that the number of collision nodes is insufficient to provides the "shortcuts" to all observed infected nodes.

Define H to be the total number of nodes each of which has the shortest path to $\mathcal{W}$ within t hops and containing at least one collision node. If $H < \tilde{Z}_t'^t(u)$, there exists a node $w \in \mathcal{Z}_t^t(u)$ such that $d(w, \mathcal{W}) > t$. Therefore, $\mathcal{W}$ can not be the Jordan infection cover and the theorem is proved.

In the rest of the proof, we will show that $H < \tilde{Z}_t'^t(u)$. We first have the lower bound on $\tilde{Z}_t'^t(u)$ according to E_6,

$$\tilde{Z}_t'^t(u) \geq [(1-\delta)^2 \mu q]^{t-1}(1-\delta)\theta \tag{3.3}$$

For each node w_i in $\mathcal{W}$, define H_i to be the total number of nodes each of which has the shortest path to w_i within t hops and containing at least one collision node. The upper bound of H_i can be obtained based on Lemma 6 in Zhu and Ying (2015b).

$$H_i \leq c[(1+\delta)\mu]^{\frac{3}{4}(t+D)+\frac{1}{2}} + c[(1+\delta)\mu]^{(\frac{5}{4}-\frac{\alpha}{2})(t+D)+2},$$

and we have

$$H \leq \sum_{i=2}^{m} H_i \leq mc[(1+\delta)\mu]^{\frac{3}{4}(t+D)+\frac{1}{2}} + mc[(1+\delta)\mu]^{(\frac{5}{4}-\frac{\alpha}{2})(t+D)+2},$$

Since $\frac{1}{2} < \alpha < 1$, we have $\alpha = \frac{1}{2} + \alpha'$ where $0 < \alpha' < \frac{1}{2}$ is a constant, we have

$$\frac{H}{\tilde{Z}_t'^t(u)} \leq \frac{mc[(1+\delta)\mu]^{\frac{3}{4}(t+D)+\frac{1}{2}}}{[(1-\delta)^2\mu q]^{t-1}(1-\delta)\theta} + \frac{mc[(1+\delta)\mu]^{(\frac{5}{4}-\frac{\alpha}{2})(t+D)+2}}{[(1-\delta)^2\mu q]^{t-1}(1-\delta)\theta} \tag{3.4}$$

$$= \frac{mc}{\mu}\left(\frac{(1+\delta)^{\frac{3}{4}+\left(\frac{3D}{4}+\frac{1}{2}\right)\frac{1}{t}}}{[(1-\delta)^2 q]^{1-\frac{1}{t}}(1-\delta)\theta\mu^{\frac{1}{4}-\left(\frac{3D}{4}+\frac{5}{2}\right)\frac{1}{t}}}\right)^t \tag{3.5}$$

$$+ \frac{mc}{\mu}\left(\frac{(1+\delta)^{1-\frac{\alpha'}{2}+\left[\left(1-\frac{\alpha'}{2}\right)D+2\right]\frac{1}{t}}}{[(1-\delta)^2 q]^{1-\frac{1}{t}}(1-\delta)\theta\mu^{\frac{\alpha'}{2}-\left(4+\left(1-\frac{\alpha'}{2}\right)D\right)\frac{1}{t}}}\right)^t \tag{3.6}$$

Since $t >> D$, we have

$$t \geq \min\left\{3D+2, \left(\frac{2}{\alpha'}-1\right)D + \frac{4}{\alpha'}, \left(\frac{4}{\alpha'}-2\right)D + \frac{16}{\alpha'}\right\}$$

Therefore, Inequality (3.6) becomes

$$\frac{H}{\tilde{Z}_t'^t(u)} \leq \frac{2mc}{\mu}\left(\frac{(1+\delta)}{[(1-\delta)^2 q]\mu^{\frac{\alpha'}{4}}}\right)^t.$$

Since $\mu > \frac{1}{Cq\theta}\log n$ and δ, q, α' are constants, we have

$$\frac{(1+\delta)}{[(1-\delta)^2 q]\mu^{\frac{\alpha'}{4}}} < 1$$

when

$$n > \exp\left(\frac{1}{2}\left(\frac{(1+\delta)}{(1-\delta)^2 q}\right)^{\frac{4}{\alpha'}}\right).$$

Therefore, we have

$$\frac{H}{\tilde{Z}_t'^t(u)} \leq \frac{2mc}{\mu} \leq \epsilon',$$

where $\epsilon' \in (0,1)$ is a constant and the inequality holds for sufficiently large n. There are at least $(1-\epsilon')\tilde{Z}_t'^t(u)$ nodes which cannot be reached from $\mathcal{W}$ with t hops on g^-. Hence we have $e(\mathcal{I}', \mathcal{W}) > t$. Therefore, we proved that $s_1 \in \mathcal{W}^*$ with a high probability when n is sufficiently large, i.e., we have

$$\Pr(s_1 \in \mathcal{W}^*) \geq 1 - \frac{\epsilon}{m}$$

since m is a constant. Then, by applying the union bound, we have, for sufficiently large n,

$$\Pr(s_i \in \mathcal{W}^*, \quad \forall i = 1, \cdots, m) \geq 1 - \epsilon$$

Note, we have $|\mathcal{W}^*| = m$. Therefore, we have

$$\Pr(\mathcal{S} = \mathcal{W}^*) \geq 1 - \epsilon$$

Hence, the Jordan infection cover equals the actual source set with a high probability.

□

We now briefly explain the conditions. Recall that μ is the average node degree, q is the lower bound on the infection probability and θ is the lower bound on the reporting probability, so $\mu q \theta$ is a lower bound on the average number of observed infected neighbors of a node that was infected before time slot t. Therefore, condition (c1) requires that this lower bound is $\Omega(\log n)$, and condition (c2) requires that the threshold used in the candidate selection algorithm is a constant fraction of the average number of observed infected neighors. Applying the Chernoff bound, conditions (c1) and (c2) together yield the following conclusions:

(i) any node who was infected before or at time slot $t-1$ (hence, including the sources) will be selected into the candidate set with a high probability,

(ii) any node that is $t + D + 1$ hops away from the set of sources will not have Y or more observed infected neighbors with a high probability, and

(iii) any node that is more than $t + D + 1$ hops away from the set of sources will not have any observed infected neighbors.

Based the above facts, with a high probability, the candidate set includes all nodes who were infected at $t-1$ or earlier, and any node in g^- is at most $t + D$ hops away from all sources.

Condition (c3) is on the infection duration. We first restrict $t = \omega(D)$ so that the infection subgraphs starting from different sources are likely to overlap and form a connected component. This is a more interesting regime than the one in which infection subgraphs are disconnected from each other. $\limsup_{n\to\infty} \frac{t}{\frac{\log n}{\log \mu}} < \frac{2}{3}$ is a critical condition. The intuition why it is required is explained below. Figure 3.1 provides a pictorial explanation of the proof. The picture illustrates the breadth-first-search (BFS) tree $T^\dagger$ rooted at source s_1 with height $t + D$, where s_1 is one of the m sources. The nodes in orange are the observed infected nodes whose infection was originated from s_1. The blue nodes are unobserved nodes. A node is said to be on level i of the BFS if its hop-distance to s_1 is i. Assume $m = 3$ and consider a set of three nodes who are within $t + D$ hops from s_1 but not includes s_1 (e.g., $\mathcal{W} = \{w_1, w_2, w_3\}$ in Figure 3.1). Suppose the infection eccentricity of $\mathcal{W}$ is $\leq t$. Since s_1 has a sufficient number of neighbors according to the definition of μ, with a high probability, there exists a subtree of $T^\dagger$ starting from an offspring of s_1, which does not include any node in $\mathcal{W}$. Assume u is the root of such a subtree in Figure 3.1. The yellow area in Figure 3.1 includes the level-t observed infected nodes on subtree $T_u^{-s_1}$. Any path from w_1, w_2 or w_3 to the yellow area, formed by edges on $T^\dagger$, must have hop-distance larger than t. Therefore, if the infection eccentricity of $\mathcal{W}$ is at least t, there must exist a path from $\mathcal{W}$ to each of the nodes in the yellow area with hop-distance $\leq t$, and such a path must include at least one edge which is not in $T^\dagger$ (we call these edges *collision edges*). In the detailed analysis, we will prove that with a high probability, the number of nodes within t hops from $\mathcal{W}$ via the collision edges is order-wise smaller than the number of nodes in the yellow area when (c3) holds. Therefore, the Jordan cover has to include s_1. The same argument applies to other sources.

3.3.2 *Impossibility Results*

Theorem 5 in Zhu and Ying (2016) presents the conditions under which it is impossible to identify the single source under the IC diffusion on the ER random graph, which is a special case of the model in this chapter. Assuming SI or IC model, based on Lemma 1 in Zhu and Ying (2016), we have that with a high probability, all nodes of the ER graph become infected when the infection duration is at least

$$t_u \triangleq \left\lceil \frac{\log n}{\log \mu + \log q} \right\rceil + 2.$$

When this occurs, it is impossible to detect the sources since the nodes are indifferentiable.

Theorem 3.3. *Assume the multi-source diffusion follows the IC or SI model. If* $24 \log n < q\mu << \sqrt{n}$ *and* q *is a constant independent of* n*, then*

$$\lim_{n\to\infty} \Pr(\mathcal{I} = \mathcal{V}(g)) = 1$$

when the observation time $t \geq t_u$*. In other words, the entire network is infected after* t_u *with a high probability. In such a case, the probability of finding the sources diminishes to zero as* $n \to \infty$*.*

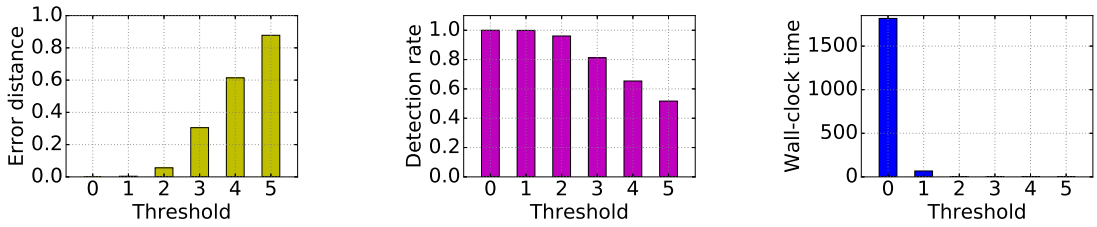
3.4 Performance Evaluation

In this section, we evaluated the performance of our algorithms via simulations. The performance metrics used in this chapter include:

- Error distance: The error distance is defined to be

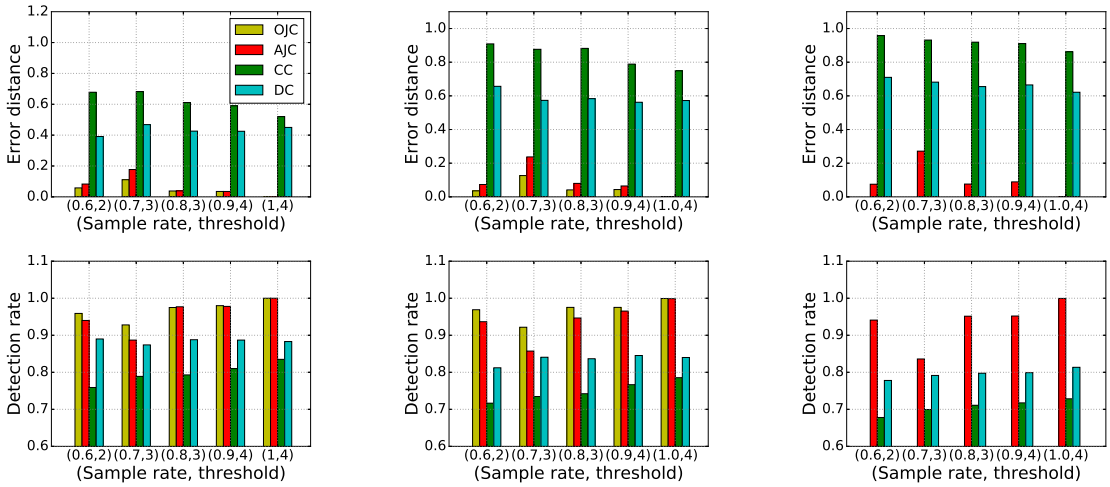
$$\min_{\mathcal{P}\in \text{permutation}(\mathcal{S}')} \sum_{i=1}^{m} \frac{d(s_i, p_i)}{m},$$

  where $s_1, s_2, \ldots, s_m$ are the real sources, $\mathcal{S}'$ is the set of detected sources and $\mathcal{P} = (p_1, p_2, ..., p_m)$ is a permutation of $\mathcal{S}'$.



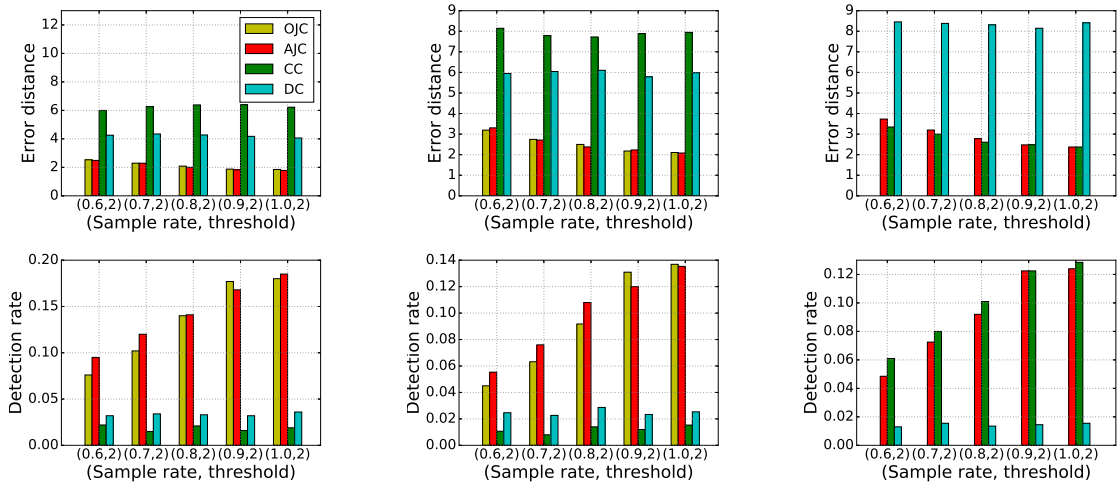
(a) Error distance (b) Detection rate (c) Wall-clock time

Figure 3.2: Performance of OJC with different threshold values on the ER random graph.



(a) Source number: 2, infection size: $100 \sim 300$. (b) Source number: 3, infection size: $200 \sim 400$. (c) Source number: 4, infection size: $300 \sim 500$.

Figure 3.3: The performance of OJC, AJC, CC and DC on the ER random graph with different sample rates and threshold values.



(a) Source number: 2, infection size: $100 \sim 300$.

(b) Source number: 3, infection size: $200 \sim 400$.

(c) Source number: 4, infection size: $300 \sim 500$.

Figure 3.4: The performance of OJC, AJC, CC and DC on the power grid network with different sample rates and threshold values.

- Detection rate: The detection rate is defined as

$$\frac{|\mathcal{S} \cap \mathcal{S}'|}{m}.$$

We compared our algorithms with two heuristic algorithms (DC and CC) based on K-Means, which have been used for comparison in Luo *et al.* (2014). The algorithms proposed in Luo *et al.* (2014) and Chen *et al.* (2014) are the same as AJC without candidate selection. In both DC and CC, the initial centroids are randomly chosen. During the clustering step of each iteration in K-Means, we selected distance centroid of each cluster in DC and selected closeness centroid in CC, where distance centroid is defined as $\arg\min_{v \in \mathcal{C}} \sum_{u \in \mathcal{C} \cap \mathcal{I}'} d(v, u)$, where $\mathcal{C}$ is the set of nodes in the cluster. Closeness centroid is defined as $\arg\max_{v \in \mathcal{C}} \sum_{u \in \mathcal{C} \cap \mathcal{I}', u \neq v} \frac{1}{d(u,v)}$. The following experiments were conducted on an server with 8 Intel Xeon X3450 CPUs and 16G RAM with Linux 64 bit system. All algorithms were implemented with Python 2.7.

3.4.1 OJC with Different Thresholds

In Figure 3.2, we evaluated OJC on the ER random graph. In the experiments, we generated an ER random graph with $5,000$ nodes and wiring probability 0.002. We used the homogeneous SI model for diffusion with infection probability 0.8. In this experiment, we limited the infection network size to be $100 \sim 300$ and the number of sources to be 2 due to the computational complexity of the OJC algorithm. Figure 3.2 shows the performance of OJC with different thresholds. From the results, the detection rate is close to one and the error distance is close to zero under OJC with threshold 0 or 1. However, the running time is 1,817 seconds versus 68 seconds. So the candidate selection algorithm with threshold one results in $27\times$ reduction of the running time. When the threshold increases 2, the running time reduces to 3 seconds, which is a $600\times$ reduction of the running time. Both the detection rate and the error distance became slightly worse in this case. The detection rate in this case is 0.961 and the error distance is 0.056.

3.4.2 OJC, AJC and Other Heuristics

We further evaluated the performance of OJC and AJC on both the power grid network Watts and Strogatz (1998) and ER random graph (size: 5000, wiring probability: 0.002) and compared them with DC and CC heuristics. We used the homogeneous SI model with infection probability 0.8 to generate the diffusion sequences. For AJC/CC/DC, for each diffusion sequence, we repeated the algorithm 100 times from different initial conditions and chose the source set with the smallest the smallest-infection-eccentricity/largest-closeness-centrality/smallest-distance-centrality. In Figure 3.3 and 3.4, the $x-$axis represents the combinations of sample rate and threshold. On the ER random graph, we increased the threshold as the sample rate increased to

control the running time. For the power-grid network, since the average node degree is only 2, we set threshold equal to 2 for experiments for all sample rates. As we can see from the figures that when fixing the threshold, the performance of all algorithms (in terms of both error distance and detection rate) improves as the sample rate increases because we had more information about the diffusion. From Figure 3.3 and 3.4, we can also see that AJC outperforms DC and CC, and has similar performance with OJC. Note that with four sources, OJC became very slow on both the ER random graph and the power grid network because its complexity increases exponentially in the number of sources. So for the cases with four sources, we only simulated AJC.

3.5 Conclusions

In this chapter, we studied the problem of detecting multiple diffusion sources under the heterogeneous SIR model with incomplete observations. We defined a concept called Jordan cover and developed the OJC algorithm based on that. Our theoretical analysis showed that OJC finds the set of sources in the ER random graph with probability one asymptotically under mild conditions. To the best of our knowledge, this is the first theoretic performance guarantee for multiple information sources detection in non-tree networks. Since the computational complexity of OJC is polynomial in n but exponential in m, we proposed a heuristic algorithm — the AJC algorithm. Our simulation results showed that OJC and AJC algorithms have similar performance and both significantly outperform existing algorithms.

Chapter 4

DIFFUSION HISTORY RECONSTRUCTION

In this chapter, we go beyond identifying the source of diffusion and study the problem of reconstructing the entire history of a diffusion process, named as *diffusion history reconstruction*, which has been studied only very recently Sefer and Kingsford (2014). In large-scale networks, due to the cost and privacy concerns, it is almost impossible to monitor the entire network and collect the complete diffusion trace, which makes reconstructing the diffusion history not trivial. For example, when a computer virus propagates among different computer through the network, we cannot track the infection of each computer because of the privacy limitation. And when some fake news goes viral on the Internet, which means thousands of individuals or websites are involved in the diffusion process, it is difficult to obtain the time when each individual or website that propagates the news. We assume that the diffusion process starting from one or mulitple sources follows the Susceptible-Infected (SI) model, a variant of the popular SIR model first proposed in Kermack and McKendrick (1927), and a *single* partial snapshot of the network is given, which includes the set of "infected" nodes, and the corresponding infection time of a subset of "infected" nodes. The nodes with known "infection" time can be thought as monitor nodes that were placed in the network. Each monitor node can record the time at which the node is "infected" and report the infection time. The main contributions of this chapter are summarized below.

- We formulate the diffusion history reconstruction problem as a maximum a posteriori (MAP) estimate problem, and prove that the problem is NP-hard by

reducing an arbitrary set cover problem to a diffusion history reconstruction problem.

- We propose a greedy and step-by-step reconstruction algorithm to reconstruct the most likely network state at time slot τ based on the network state at time slot $\tau - 1$ while guaranteeing the state is consistent with partial observation, and further develop a greedy algorithm for a single-step construction. The key idea of the single-step construction algorithm is to convert the problem to the weighted set cover problem, for which a well-known greedy algorithm provides a guarantee on the approximation ratio.

- The problem of reconstructing the diffusion history for single-source diffusion processes has been studied in Chen *et al.* (2015). In this chapter, we studied multi-source diffusion processes, which include single-source diffusion as a special case. The problem is more difficult because the number of initial states is proportional to V_I^N, where V_I is the number of infected nodes and N is the number of sources. It is almost impossible to use the single-source algorithm to reconstruct the diffusion history for multi-source diffusion processes because of the high complexity. Assuming the number of sources, $N(N \geq 1)$, is known, we propose an algorithm to find all possible initial states of the diffusion to reduce the complexity. It is shown that the initial states found by our algorithm are always consistent with the partial observation.

- We prove that the diffusion history obtained by the step-by-step reconstruction algorithm is always consistent with the partial observation, and the computational complexity of the algorithm is $O(V_I^{N+1}E_I)$, where V_I is the number of infected nodes observed in the snapshot, E_I is the number of edges between the observed infected nodes and N is the number of sources.

- We evaluate the performance of the algorithm on the Western States Power Grid of the United States Watts and Strogatz (1998) and Internet autonomous systems (IAS) network Leskovec *et al.* (2005), with simulated diffusion processes following the SI model. We also test our algorithm on the Weibo dataset (Weibo is a famous Chinese microblogging website). In all scenarios, we observe significant improvements of the proposed algorithm compared with other heuristic and existing algorithms.

4.1 Problem Formulation

In this section, we define the diffusion history reconstruction problem. We assume the diffusion process starting from a set of sources in the network denoted by $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed edges. Node u can infect node v if there is an edge $u \to v$. Node u is called an incoming neighbor of node v, and node v is called an outgoing neighbor of node u. We further assume the number of sources, N, is known to us, the diffusion process starts from time slot 0, and a snapshot of the network is taken at time slot T. The snapshot includes the state of every node in the network at time T, as well as the infection time of a subset of infected nodes. The goal is to infer the complete diffusion history from the partial observation.

In this chapter, we use the capital letter for constants (e.g, T), calligraphic fonts for sets (e.g., $\mathcal{E}$), and bold upper-case for matrices or vectors (e.g., $\mathbf{X}$). We use $\mathbf{X}_t$ to represent the t^{th} column of matrix $\mathbf{X}$. For each set, we use its associated capital letter for the cardinality of the set (e.g., $V = |\mathcal{V}|$). A graph is defined by its node set and edge set, e.g., a graph G with node set $\mathcal{V}$ and edge set $\mathcal{E}$ can be written as $G(\mathcal{V}, \mathcal{E})$. We use tilde to represent the matrix, set or vector to be reconstructed (e.g., $\tilde{\mathbf{X}}$). The key notation used throughout the chapter is summarized in Table 4.1.

Symbol	Definition & Description
$G(\mathcal{V}, \mathcal{E})$	a graph G with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$.
T	the time when the snapshot of the network is taken
$\mathbf{I}$	infected nodes vector, which describes the infected nodes at time T
$\mathbf{T}$	infection timing vector, which saves the observed infection time of nodes
$\mathbf{X}$	the diffusion history matrix
$\tilde{\mathbf{X}}$	the reconstructed diffusion history matrix
$\mathbf{X}_t$ $(0 \leq t \leq T)$	the network state vector at time t
$\tilde{\mathbf{X}}_t$ $(0 \leq t \leq T)$	the reconstructed network state vector at time t
$X_{v,t}$	a binary variable, which is equal to 1 when node v is in the infected state at time t, otherwise, it is 0
p_{uv}	infection probability of edge (u, v)
$\tilde{\mathcal{S}}_t$ $(0 \leq t \leq T)$	the set of susceptible nodes on $G(\mathcal{V}, \mathcal{E})$ which has infected incoming neighbors according to $\mathbf{X}_t$
$\tilde{\mathcal{I}}_n$	the set of infected nodes in $\tilde{\mathbf{X}}_n$
$\mathcal{I}$	the set of infected nodes in the snapshot at time T
$\mathcal{N}_v$	the set of incoming neighbors of node v in network $G(\mathcal{V}, \mathcal{E})$

Table 4.1: Notations

4.1.1 Diffusion Model

We assume the diffusion process follows the discrete-time Susceptible-Infected (SI) model. Each node in the network has two states: susceptible (S) and infected (I). In each time slot, every susceptible node, v, can be infected by each of its infected incoming neighbors, u, with probability p_{uv}. Once the susceptible node gets infected, it will stay at the infected state forever. The diffusion starts at time $t = 0$ from a set of initially infected nodes, which are the sources of diffusion.

4.1.2 Problem Statement

Given a network $G(\mathcal{V}, \mathcal{E})$, we assume the observation we have for reconstructing the diffusion history is a pair $(\mathbf{T}, \mathbf{I})$, such that $\mathbf{I}$, named *infected nodes vector*, is a V-dimensional vector such that $I_v = 1$ if node v is infected and $I_v = 0$ otherwise; and $\mathbf{T}$, named *infection timing vector*, is also a V- dimensional vector such that T_v is infection time of node v if node v's infection time is observed and $T_v = -1$ otherwise. Let $\mathcal{I}$ denote the set of infected nodes. Define a $V \times T$ matrix $\mathbf{X}$ to be the diffusion history such that

$$X_{v,t} = \begin{cases} 1 & \text{Node } v \text{ is in the infected state at time } t, \\ 0 & \text{Node } v \text{ is susceptible at time } t. \end{cases} \tag{4.1}$$

Note that $\mathbf{X}$ defines the entire history of the diffusion process and under the SI model, $X_{v,t} = 1$ if $X_{v,\tau} = 1$ for $\tau < t$.

We can use the column vector $\mathbf{X}_t$ of diffusion history $\mathbf{X}$ to represent the network state at time t. Then the diffusion history matrix $\mathbf{X}$ can be written as

$$\mathbf{X} = (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_T).$$

The *diffusion history reconstruction* problem can be defined as follows:

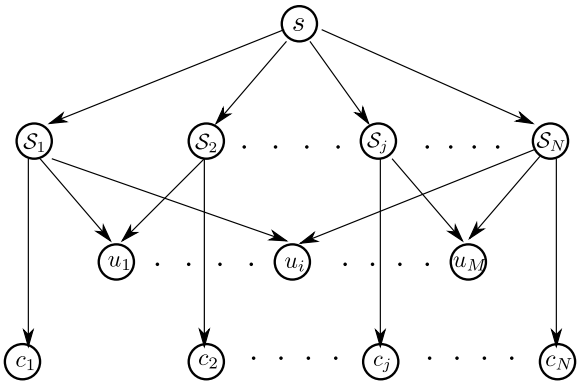


Figure 4.1: The graph built in the proof of Theorem 4.1.

Diffusion History Reconstruction Problem

Input: The underlying network for the diffusion, $G(\mathcal{V}, \mathcal{E})$, the time when the snapshot is taken, T, the infected nodes vector $\mathbf{I}$ and the infection timing vector $\mathbf{T}$.

Output: A diffusion history $\tilde{\mathbf{X}}$ such that

$$\tilde{\mathbf{X}} = \underset{\mathbf{X}}{\operatorname{argmax}} \Pr(\mathbf{X}|\mathbf{I}, \mathbf{T}). \tag{4.2}$$

$\diamond$

In order to find the $\tilde{\mathbf{X}}$, the exclusive search needs to calculate $\Pr(\mathbf{X}|\mathbf{I}, \mathbf{T})$ for all possible diffusion history $\mathbf{X}$, which increases exponentially as the number of infected nodes increases. The problem is NP-hard and the proof is presented in the appendix.

Theorem 4.1. *The diffusion history reconstruction problem defined in (4.2) is NP-hard.*

Proof. By reduction from the set cover problem. In the set cover, we are given a set of M elements $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and a set $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N\}$ of N sets whose union equals to set $\mathcal{U}$. The set cover problem is to identify the smallest subset of $\mathcal{S}$ whose union equals to the universe. We consider each element in set $\mathcal{U}$ or set $\mathcal{S}$ as a vertex on a graph and think there exists an directed edge from $\mathcal{S}_j$ $(0 \leq j \leq N)$ to u_i

$(0 \leq j \leq M\ )$ if $u_i \in \mathcal{S}_j$. Then for each $\mathcal{S}_j \in \mathcal{S}$, we add a node c_j to the graph and create a directed edge from $\mathcal{S}_j$ to c_j. After that, we add a node s to the graph and create a directed edge from s to $\mathcal{S}_j$ for each $\mathcal{S}_j \in \mathcal{S}$. An example of the graph we built is provided in Figure 4.1. We assume there is a diffusion process based on the graph in Figure 4.1 and SI model. Assume the infection probability on each edge $(s, \mathcal{S}_j)$ $(0 \leq j \leq N)$ is p_1, the infection probability on each edge between $\mathcal{S}_j$ $(0 \leq j \leq N)$ and u_i $(0 \leq i \leq M)$ is 1, and the infection probability on each edge $(\mathcal{S}_j, c_j)$ $(0 \leq j \leq N)$ is p_2 $(p_2 > p_1)$. The snapshot was taken at time $T = 3$ and all nodes are in the infected state at time 3. We observed that the infection time of node u_i $(0 \leq i \leq M)$ is 2, the infection time of node c_j $(0 \leq j \leq N)$ is 3, and the infection time of node s is 0. Thus, we have known the observation $(\mathbf{I}, \mathbf{T})$. To determine the diffusion history, we need to decide when node $\mathcal{S}_j$ $(0 \leq j \leq N)$ is infected. We try to find a diffusion history by solving (4.2). Since c_j $(0 \leq j \leq N)$ is infected at time 3, node $\mathcal{S}_j$ can only be infected at time 1 or 2. If $\mathcal{S}_j$ is infected at time 1, the probability brought by path $s \to \mathcal{S}_j \to c_j$ is

$$p_1(1 - p_2)p_2.$$

Otherwise, the the probability brought by path $s \to \mathcal{S}_j \to c_j$ is

$$p_1(1 - p_1)p_2.$$

Since $p_2 > p_1$, we have

$$p_1(1 - p_2)p_2 < p_1(1 - p_1)p_2,$$

which means node $\mathcal{S}_j$ prefers to be infected at time 2. However, since u_i $(0 \leq i \leq M)$ is infected at time 2, for each u_i, there must exist a node $\mathcal{S}_k$ infected at time 1 such that $u_i \in \mathcal{S}_k$, which means u_i can be infected at time 2. Therefore, the problem described in (4.2) can be converted to a set cover problem of finding the smallest subset $\mathcal{S}'$ of $\mathcal{S}$ to cover $\mathcal{U}$. Then we may think the nodes in $\mathcal{S}'$ are infected at time 1,

while others in $\mathcal{S}$ are infected at time 2. Since

$$\Pr(\mathbf{X}|\mathbf{I},\mathbf{T}) \propto \Pr(\mathbf{X})\Pr(\mathbf{I},\mathbf{T}|\mathbf{X}) \tag{4.3}$$

and the value of $\Pr(\mathbf{I},\mathbf{T}|\mathbf{X})$ can only be 1 or 0, this reconstructed diffusion history $\tilde{\mathbf{X}}$ has the maximum probability $\Pr(\tilde{\mathbf{X}})$ while $\tilde{\mathbf{X}}$ is consistent with $(\mathbf{I},\mathbf{T})$, $\Pr(\mathbf{I},\mathbf{T}|\mathbf{X}) = \mathbf{1}$. Thus, the set cover problem can be reduced to a special case of our diffusion history reconstruction problem, which means the diffusion history reconstruction problem is NP-hard. □

4.2 A Step-by-Step Reconstruction Algorithm

Since the diffusion history reconstruction problem defined in (4.2) is difficult to solve, we propose a heuristic algorithm with polynomial complexity in this section. According to Bayes' theorem, we have

$$\begin{aligned}\Pr(\mathbf{X}|\mathbf{I},\mathbf{T}) &= \frac{\Pr(\mathbf{X},\mathbf{I},\mathbf{T})}{\Pr(\mathbf{I},\mathbf{T})} \\ &\propto \Pr(\mathbf{X},\mathbf{I},\mathbf{T}) \\ &= \Pr(\mathbf{X})\Pr(\mathbf{I},\mathbf{T}|\mathbf{X})\end{aligned} \tag{4.4}$$

When the diffusion history $\mathbf{X}$ is known, the infection time of nodes and the set of infected nodes are fixed. Thus, in (4.4), the value of $\Pr(\mathbf{I},\mathbf{T}|\mathbf{X})$ is either 0 or 1 :

$$\Pr(\mathbf{I},\mathbf{T}|\mathbf{X}) = \begin{cases} 1 & \mathbf{X} \text{ is consistent with } (\mathbf{I},\mathbf{T}), \\ 0 & \mathbf{X} \text{ is inconsistent with } (\mathbf{I},\mathbf{T}). \end{cases} \tag{4.5}$$

where we say that the diffusion history $\mathbf{X}$ is **consistent** with observation $(\mathbf{I},\mathbf{T})$ if the following two conditions hold:

H1 $X_{v,T} = 1$ when node v is an infected node according to $\mathbf{I}$ and $X_{v,T} = 0$ otherwise, and

H2 $X_{v,t} = 1$ for $t \geq T_v$ if $T_v \neq -1$.

We further define the network state at time τ ($\mathbf{X}_\tau$) to be **consistent** with $(\mathbf{I}, \mathbf{T})$ if the following conditions hold:

S1 If $I_v = 0$, then $X_{v,\tau} = 0$. In other words, node v should be in the susceptible state if it is in the susceptible state at time T.

S2 If $I_v = 1$ and $0 \leq T_v \leq \tau$, then $X_{v,\tau} = 1$. In other words, node v should be in the infected state at time τ if it was infected at or before time slot τ.

S3 If $I_v = 1$ and $T_v > \tau$, then $X_{v,\tau} = 0$, and one of the following two conditions must hold

c1 There exists node u with $T_u > \tau$ and

$$d(u, v) \leq T_v - T_u.$$

c2 There exists node u with $X_{u,\tau} = 1$ and

$$d(u, v) \leq T_v - \tau.$$

Here $d(u, v)$ is defined to be the length of the shortest *Infection-Time-Free path* (or ITF-path) between node u and node v, where an ITF-path is a path that includes only infected nodes such that $X_{w,\tau} = 0$ except the two end nodes. The condition (c1) means node u, who was infected at time slot T_u, can infect node v via an ITF-path at T_v. The condition (c2) means node u, who has already been infected at time slot τ, can infect node v via an ITF-path at T_v.

S4 If $I_v = 1$ and $T_v = -1$, then either $X_{v,\tau} = 1$, or $X_{v,\tau} = 0$ and one of the following two conditions must hold

c1 There exists node u with $T_u > \tau$ and

$$d(u, v) \leq T - T_u.$$

c2 There exists node u with $X_{u,\tau} = 1$ and

$$d(u, v) \leq T - \tau.$$

Here the condition (c1) means node u, who was infected at time slot T_u, can infected node v via an ITF-path before or at time T. The condition (c2) means node u, who has already been infected at time slot τ, can infect node v via an ITF-path before or at time T.

According to the discussion above, the problem of

$$\max_{\mathbf{X}} \Pr(\mathbf{X}|\mathbf{I}, \mathbf{T})$$

is equivalent to

$$\begin{aligned} &\max_{\mathbf{X}} \Pr(\mathbf{X}) \\ &\text{subject to: } \mathbf{X} \text{ is consistent with } (\mathbf{I}, \mathbf{T}). \end{aligned} \tag{4.6}$$

Since $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_T$ form a Markov chain under the SI model, we have

$$\begin{aligned} \Pr(\mathbf{X}) &= \Pr(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_T) \\ &= \Pr(\mathbf{X}_0) \Pr(\mathbf{X}_1|\mathbf{X}_0) \ldots \Pr(\mathbf{X}_T|\mathbf{X}_{T-1}) \\ &= \Pr(\mathbf{X}_0) \prod_{\tau=1}^{T} \Pr(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}). \end{aligned}$$

Now to solve (4.6), our greedy approach is to recursively solve the following *single-step reconstruction* problem with a given $\mathbf{X}_{\tau-1}$

$$\begin{aligned} &\max_{\mathbf{X}_\tau} \Pr(\mathbf{X}_\tau|\mathbf{X}_{\tau-1}) \\ &\text{subject to: } \mathbf{X}_\tau \text{ is consistent with } (\mathbf{I}, \mathbf{T}). \end{aligned} \tag{4.7}$$

The first step of this greedy algorithm needs input $\mathbf{X}_0$, i.e., the identify of the sources. Define $\mathbf{1}^{(\mathcal{F})}$ to be a V-dimensional vector, where $\mathcal{F} \subset \mathcal{V}$, such that $\mathbf{1}_v^{(\mathcal{F})} = 1$ for $v \in \mathcal{F}$ and $\mathbf{1}_m^{(\mathcal{F})} = 0$ for $m \notin \mathcal{F}$. The algorithm will set $\tilde{\mathbf{X}}_0^{(\mathcal{F})} = \mathbf{1}^{(\mathcal{F})}$ for each set of possible sources $\mathcal{F}$ and then calculates $\tilde{\mathbf{X}}_\tau^{(\mathcal{F})}$ by recursively solving (4.7) (again with a greedy algorithm which will be presented in the next subsection). Then the diffusion history is set to be the most likely $\tilde{\mathbf{X}}^{(\mathcal{F})}$. The pseudo code is presented in Algorithm 4.1.

Algorithm 4.1: The Step-by-Step Reconstruction Algorithm (SSR)

for *each possible set of sources, i.e.,* $\mathcal{F} \subset \mathcal{V}_s$ **do**
  Set $\tilde{\mathbf{X}}_0^{(\mathcal{F})} = \mathbf{1}^{(\mathcal{F})}$;
  for $1 \leq \tau \leq T$ **do**
    Set $\tilde{\mathbf{X}}_\tau^{(\mathcal{F})}$ to be solution of problem (4.7) with $\mathbf{X}_{\tau-1} = \tilde{\mathbf{X}}_{\tau-1}^{(\mathcal{F})}$.
  Set $\gamma_{\mathcal{F}} = \prod_{1\leq\tau\leq T} \Pr\left(\tilde{\mathbf{X}}_\tau^{(\mathcal{F})} \middle| \tilde{\mathbf{X}}_{\tau-1}^{(\mathcal{F})}\right)$.
Set $\mathcal{F}^* \in \arg\max_{\mathcal{F}\in\mathcal{V}_f} \gamma_{\mathcal{F}}$.
return $\tilde{\mathbf{X}}^{(\mathcal{F}^*)}$

Note that $\mathcal{V}_s$ is defined to be the set of feasible source combinations with each element represents a set of sources that can generate the observation. $\mathcal{V}_s$, can be determined by Algorithm 4.3, which is introduced later.

4.2.1 Single-Step Reconstruction

We now focus on solving (4.7) when $\tilde{\mathbf{X}}_{\tau-1}$ is given. In the following discussion, we assume $\tilde{\mathbf{X}}_{\tau-1}$ is given. Note that node v can become an infected node at time τ if $I_v = 1$, $\tilde{X}_{v,\tau-1} = 0$ and it has at least an infected incoming neighbor at time $\tau - 1$. Denote by $\tilde{\mathcal{S}}_{\tau-1}$ the set of susceptible nodes with infected incoming neighbors according to $\tilde{\mathbf{X}}_{\tau-1}$, $\mathcal{N}_v$ the set of incoming neighbors of node v, and $\tilde{\mathcal{I}}_{\tau-1}$ the set of infected nodes at time $\tau - 1$. For each node $v \in \tilde{\mathcal{S}}_{\tau-1}$, the probability that v is not

infected by its incoming infected neighbors at time τ is

$$\prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv}).$$

Furthermore, the objective in (4.7) can be written as

$$\begin{aligned} \Pr(\mathbf{X}_\tau | \tilde{\mathbf{X}}_{\tau-1}) = \prod_{v \in \tilde{\mathcal{S}}_{\tau-1}} & \underbrace{\big(1 - \prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})\big)^{X_{v,\tau}}}_{(a)} \\ & \underbrace{\big(\prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})\big)^{1 - X_{v,\tau}}}_{(b)}, \end{aligned} \tag{4.8}$$

where term (a) is the probability that v gets infected at time τ while term (b) represents the probability that v stays susceptible at time τ. The log-likelihood can be written as

$$\begin{aligned} & \log \Pr(\mathbf{X}_\tau | \tilde{\mathbf{X}}_{\tau-1}) \\ = & \sum_{v \in \tilde{\mathcal{S}}_{\tau-1}} \Big(X_{v,\tau} \log \big(1 - \prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})\big) \\ & \qquad + (1 - X_{v,\tau}) \log \prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv}) \Big) \\ = & \sum_{v \in \tilde{\mathcal{S}}_{\tau-1}} X_{v,\tau} \log \frac{1 - \prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})}{\prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})} \\ & \qquad + \log \prod_{u \in \mathcal{N}_v \cap \tilde{I}_{\tau-1}} (1 - p_{uv}) \\ = & \sum_{v \in \tilde{\mathcal{S}}_{\tau-1}} \alpha_v^\tau X_{v,\tau} + \beta_v^\tau, \end{aligned} \tag{4.9}$$

where

$$\alpha_v^\tau = \log \frac{1 - \prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})}{\prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})} \tag{4.10}$$

and

$$\beta_v^\tau = \log \prod_{u \in \mathcal{N}_v \cap \tilde{\mathcal{I}}_{\tau-1}} (1 - p_{uv})$$

are two constants whose values depend only on $\tilde{\mathbf{X}}_{\tau-1}$. Therefore, optimization problem described in (4.7) can be written as

$$\begin{aligned} \max_{\mathbf{X}_\tau} \quad & \sum_{v \in \tilde{\mathcal{S}}_{\tau-1}} \alpha_v^\tau X_{v,\tau} + \beta_v^\tau \\ \text{subject to} \quad & \mathbf{X}_\tau \text{ is consistent with } \mathbf{I} \text{ and } \mathbf{T}. \end{aligned} \tag{4.11}$$

Note that only nodes in $\tilde{\mathcal{S}}_{\tau-1} \cap \mathcal{I}$ can change from susceptible to infected at time slot τ. The problem is combinatoric in nature since $X_{v,\tau} \in \{0, 1\}$. Next we reduce the problem above to a weighted set cover problem. Note that the consistency conditions are to guarantee all infected nodes can be infected at the observed infection time or by the time at which the snapshot was taken. It is not difficult to see that we only need to check the consistency of nodes in $\mathcal{I} \setminus \tilde{\mathcal{I}}_{\tau-1}$ for $\mathbf{X}_\tau$ because nodes in $\tilde{\mathcal{I}}_{\tau-1}$ have been successfully infected by time $\tau - 1$ and nodes in $\mathcal{V} \setminus \mathcal{I}$ should always stay as susceptible. Problem (4.11) can be converted to a weighted set cover problem with the following steps:

(1) Define the *universe* to be

$$\mathcal{U} = \mathcal{I} \setminus \tilde{\mathcal{I}}_{\tau-1},$$

i.e., the set of nodes whose consistency needs to be verified in $\mathbf{X}_\tau$.

(2) Set $\mathcal{S} = \tilde{\mathcal{S}}_{\tau-1} \cap \mathcal{I}$. For each node in $\mathcal{S}$, say node u, initiate a set $\mathcal{S}_u = \emptyset$.

(3) For each node in the universe (say node v), construct a modified-breadth-first-search (MBFS) tree as follows: Reverse all edges of the infected subgraph. On the reversed graph, starting from the node, we conduct the breadth-first search (BFS). When BFS hits an infected node in $\tilde{\mathcal{I}}_{\tau-1}$ or with observed infection time, BFS at this node stops, i.e., the node becomes a leaf-node of the MBFS tree. Note that a path from the root to any leaf-node of the MBFS-tree is an ITF-path.

– If one of leaf-nodes on the MBFS-tree, which is not in $\mathcal{S}$, satisfies the consistency condition **S3** or **S4** for root node v, then node v is removed from the universe, i.e.,

$$\mathcal{U} = \mathcal{U} \setminus \{v\}$$

because node v is consistent in $\mathbf{X}_\tau$ regardless of the states of nodes in $\mathcal{S}$.

– Otherwise, for each node (say node u) in $\mathcal{S}$ with $T_u > \tau$, remove u from $\mathcal{S}$, i.e.,

$$\mathcal{S} = \mathcal{S} \setminus \{u\}.$$

For each node u in $\mathcal{S}$:

* If $T_v \neq -1$, check whether the depth of node u on the MBFS-tree is $\leq T_v - \tau$. If it is the case, node v is added to $\mathcal{S}_u$, i.e.,

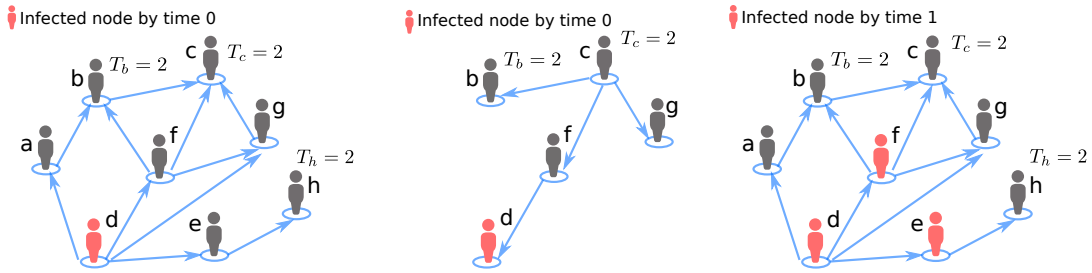
$$\mathcal{S}_u = \mathcal{S}_u \cup \{v\}.$$

* If $T_v = -1$, check whether the depth of node u on the MBFS-tree is $\leq T - \tau$. If it is the case, node v is added to $\mathcal{S}_u$.

(4) According to (4.10), we calculate the weight of set $\mathcal{S}_u$: $w_u = -\alpha_u^\tau$ for $u \in \mathcal{S}$. For each $u \in \mathcal{S}$ with $w_u < 0$, set $X_{u,\tau} = 1$, change the universe $\mathcal{U}$ to $\mathcal{U} \setminus \mathcal{S}_u$, and remove u from $\mathcal{S}$.

Note that $v \in \mathcal{S}_u$ implies that node v's consistency is guaranteed if node u becomes infected at time slot τ. The problem (4.11), therefore, is equivalent to identifying a set of $\mathcal{S}_u$ ($u \in \mathcal{S}$) with the smallest summation of weights to cover the universe $\mathcal{U}$.

Consider a simple network in Figure 4.2a. Assume the infection probability of any edge is 0.3, the snapshot is taken at $T = 4$, and all nodes are in the infected state at time 4. Furthermore, assume the infection time of nodes b, c and h is known ($T_b = 2$,



(a) The network when the time slot is 1.

(b) The MBFS-tree when node c is the root.

(c) The reconstructed network state for time 1.

Figure 4.2: The example of diffusion network state single-step reconstruction.

$T_c = 2$ and $T_h = 2$). We next outline the key steps to solve (4.11) by starting from node d, i.e., assuming $\tilde{\mathbf{X}}_0 = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^{tr}$ where tr means the transpose. So

$$\tilde{\mathcal{I}}_0 = \{d\}$$

and

$$\mathcal{U} = \mathcal{I} \setminus \tilde{\mathcal{I}}_0 = \{a, b, c, e, f, g, h\}.$$

Furthermore,

$$\tilde{\mathcal{S}}_0 = \{a, f, g, e\}.$$

Now consider the MBFS rooted at node c. The algorithm first explores the outgoing neighbors of c, which are nodes b, f and g. The edges (c, b), (c, f) and (c, g) are added to the MBFS tree. Since b's infection time is known, the outgoing neighbors of b should not be explored in the next step. In the next step, the MBFS checks the outgoing neighbors of f and g. Since node d is an outgoing neighbor of node f, edge (f, d) is added to the MBFS-tree. And the outgoing neighbors of node g, which are f and d, are already explored by the MBFS, the MBFS at g stops. Since node d does not have any outgoing neighbors, the MBFS stops and the MBFS-tree at root c is in Figure 4.2b.

On the MBFS-tree in Figure 4.2b, nodes b and d do not satisfy **S3.c1** or **S3.c2** for node c, so node c cannot be removed from the universe. We then check the ITF-paths to nodes f and g on the MBFS-tree, and both ITF-paths have a length $\leq T_c - 1$. So node c is added to $\mathcal{S}_f$ and $\mathcal{S}_g$. After doing a similar procedure for other MBFS-trees, we have $\mathcal{U} = \{b, c, f, g, h\}$, $\mathcal{S}_a = \{b\}$, $\mathcal{S}_e = \{e, h\}$, $\mathcal{S}_f = \{c, b\}$, and $\mathcal{S}_g = \{c\}$. According to (4.10), the weights are $w_a = -\alpha_a^1 = 0.847$, $w_f = -\alpha_f^1 = 0.847$, $w_g = -\alpha_g^1 = 0.847$, and $w_e = -\alpha_e^1 = 0.847$. For this example, the solution to the weighted set cover problem is easy to find, which are $\mathcal{S}_f$ and $\mathcal{S}_e$, i.e., nodes f and e should become infected at time slot 1. The reconstructed network state at time slot 1 is shown in Figure 4.2c.

In the discussion above, we need the MBFS-tree of every remaining-infected-node in $\tilde{\mathbf{X}}_{\tau-1}$, while the MBFS-tree depends on the state of $\tilde{\mathbf{X}}_{\tau-1}$. Running the MBFS at every single step is very time-consuming. Instead, we run the MBFS starting from every infected-node at the beginning of the algorithm and save the MBFS-trees. Then in each iteration, when a remaining-infected-node is selected to be an already-infected-node (say node u is selected), we prune the subtree starting from node u from all MBFS-trees but keep node u. The reason the subtree starting from node u can be pruned is because for any node on the subtree, say node y, we have $d(v, y) > d(v, u)$, where v is the root of the MBFS-tree, so node v can be infected by node y via an ITF-path after τ only if it can be infected by node u via an ITF-path.

An single-step reconstruction algorithm based on weighted set cover is stated in Algorithm 4.2, which is formed by four steps:

1. Prune each MBFS-tree rooted at a susceptible node v in previous network state with $I_v = 1$;

2. Convert the problem to a weighted set cover problem by following the procedure

in Page 4;

3. Solve the weighted set cover problem by using greedy set cover algorithm;

4. Obtain the network state according to the result of greedy set cover algorithm and calculate the objective in (4.11).

As described in Algorithm 4.1, by using Algorithm 4.2 recursively, we can get a diffusion history, which is consistent with the observation.

In general, the weighted set cover problem is NP-hard. However, we can use the greedy set cover algorithm in Young (2008) to find a feasible solution with performance guarantee. Define O to be the objective value of Algorithm 4.2 for equation (4.11) and $O^\star$ to be the objective value of the optimal solution. Let k denote the largest set size for the set cover problem and $H_k = \sum_{i=1}^{k} 1/i$.

Lemma 4.1. *By using Algorithm 4.2, we have* $O \geq H_k O^\star$.

Proof. The proof is straight forward according to Theorem 1 in Young (2008). □

Define $G_I(\mathcal{V}_I, \mathcal{E}_I)$ to be the infected subgraph of $G(\mathcal{V}, \mathcal{E})$, which is formed by the infected nodes observed at T. Then we have $V_I = |\mathcal{V}_I|$. The next lemma shows the feasibility of the SSR algorithm for generating a consistent diffusion history.

Lemma 4.2. $\tilde{\mathbf{X}}_\tau$ *output by Algorithm 4.2 is consistent with the observation* $(\mathbf{I}, \mathbf{T})$ *if* $\tilde{\mathbf{X}}_{\tau-1}$ *is consistent with the observation.*

Proof. According to our previous definition, the reconstruction state $\tilde{\mathbf{X}}_\tau$ is consistent with the observation $(\mathbf{I}, \mathbf{T})$ if for each node, one of the conditions **S1**, **S2**, **S3** and **S4** holds.

In the Single-Step Reconstruction algorithm, only the observed infected nodes can get infected in the reconstruction. Therefore, each observed susceptible node satisfies

Algorithm 4.2: Single-Step Reconstruction

Input : Network $G(\mathcal{V}, \mathcal{E})$, the previous reconstructed network state $\tilde{\mathbf{X}}_{\tau-1}$, the observed information $(\mathbf{I}, \mathbf{T})$, current time τ and the MBFS-tree $\mathbb{T}_v$ $(v \in \mathcal{I} \setminus \tilde{\mathcal{I}}_{\tau-1})$ rooted at node v from previous step.

Output: The network state $\tilde{\mathbf{X}}_\tau$, the value of objective in (4.11), o, and the MBFS-trees after pruning.

$o \leftarrow 0$; $\mathcal{U} \leftarrow \mathcal{I} \setminus \tilde{\mathcal{I}}_{\tau-1}$; $\tilde{\mathbf{X}}_\tau \leftarrow \tilde{\mathbf{X}}_{\tau-1}$; $\mathcal{S} \leftarrow \tilde{\mathcal{S}}_{\tau-1} \cap \mathcal{I}$;
let $\mathcal{S}_v \leftarrow \emptyset$ for each $v \in \mathcal{S}$; $\mathcal{M} \leftarrow \{v | v \in \mathcal{I}, T_v \neq -1\}$;
for $v \in \mathcal{I} \setminus \tilde{\mathcal{I}}_{\tau-1}$ **do**
  let $t_v \leftarrow T_v$ if $v \in \mathcal{M}$. Otherwise, $t_v \leftarrow T$;
  for $u \in \{$*the nodes on* $\mathbb{T}_v\}$ **do**
    if $u \in \tilde{\mathcal{I}}_{\tau-1}$ **then** remove the subtree rooted at u on $\mathbb{T}_v$ except u;
  if *there exists a node* $u \in \{$*the nodes on* $\mathbb{T}_v\} \cap (\mathcal{M} \setminus \tilde{\mathcal{I}}_{\tau-1})$ *such that the depth of* u *on* $\mathbb{T}_v$ *is* $\leq t_v - T_u$ *and* $T_u > \tau$ **then** remove v from $\mathcal{U}$;
  else if *there exists a node* $u \in \{$*the nodes on* $\mathbb{T}_v\} \cap \tilde{\mathcal{I}}_{\tau-1}$ *such that the depth of* u *on* $\mathbb{T}_v$ *is* $\leq t_v - \tau$ **then** remove v from $\mathcal{U}$;
  else
    for $u \in \mathcal{S}$ **do**
      if $u \in \mathcal{M}$ *and* $T_u > \tau$ **then continue**;
      else if $u \in \{$*the nodes on* $\mathbb{T}_v\}$ *and the depth of* u *on tree* $\mathbb{T}_v$ *is* $\leq t_v - \tau$ **then** $\mathcal{S}_u \leftarrow \mathcal{S}_u \cup \{v\}$;
remove any node $v \in \mathcal{S}$ with $T_v > \tau$;
if *the union of* $\mathcal{S}_v$ *for any* $v \in \mathcal{S}$ *is not equal to* $\mathcal{U}$ **then** $o \leftarrow -\infty$;
else
  $w_v \leftarrow -\alpha_v^\tau$ for any $v \in \mathcal{S}$;
  for $v \in \mathcal{S}$ **do**
    if $w_v < 0$ *or* $T_v = \tau$ **then**
      $X_{v,\tau} \leftarrow 1$; $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_v$; $\mathcal{S} \leftarrow \mathcal{S} \setminus \{v\}$;
  let $\mathcal{R}$ to be the result of the greedy set cover algorithm Young (2008) on the subset $\mathcal{S}_v$ for any $v \in \mathcal{S}$ and the universe $\mathcal{U}$;
if $o \neq -\infty$ **then**
  set $X_{v,\tau} \leftarrow 1$ for any $v \in \mathcal{S}$ with $\mathcal{S}_v \in \mathcal{R}$;
  calculate the objective value in (4.11), o;
return $\tilde{\mathbf{X}}_\tau$, o and $\mathbb{T}_v$ for $v \in \mathcal{V}$;

Algorithm 4.3: Build Feasible Initial States

Input : Network $G(\mathcal{V}, \mathcal{E})$, the observed information $(\mathbf{I}, \mathbf{T})$.
Output: The set of feasible initial states $\mathcal{V}_s$.

$\mathcal{M} \leftarrow \{v | v \in \mathcal{I}, T_v \neq -1\}$;
$\mathcal{P} \leftarrow \{v | v \in \mathcal{I}, T_v = 0\}$;
let $t_v \leftarrow T$ for $v \in \mathcal{I}$ with $T_v = -1$ and $t_v \leftarrow T_v$ for $v \in \mathcal{I}$ with $T_v \neq -1$;
for $v \in \mathcal{I} \setminus (\mathcal{M} \setminus \mathcal{P})$ **do**
  for *any* $u \in \mathcal{I} \setminus \{v\}$ **do**
    build the MBFS tree $\mathbb{T}_u$;
  let $\mathcal{I}_v = \emptyset$;
  for $u \in \mathcal{M}$ **do**
    if *there exists a node* $w \in \{$*the nodes on* $\mathbb{T}_u\} \cap \mathcal{F}$ *such that the depth of* w *on* $\mathbb{T}_u$ *is* $\leq t_u$ **then** $\mathcal{I}_v = \mathcal{I}_v \cup \{u\}$;
  for $u \in \mathcal{I} \setminus \mathcal{I}_v$ **do**
    if *there exists a node* $w \in \{$*the nodes on* $\mathbb{T}_u\} \cap \mathcal{F}$ *such that the depth of* w *on* $\mathbb{T}_u$ *is* $\leq t_u$ **then** $\mathcal{I}_v = \mathcal{I}_v \cup \{u\}$;
    else if *there exists a node* $w \in \{$*the nodes on* $\mathbb{T}_u\} \cap \mathcal{M}$ *such that* $w \in \mathcal{I}_v$ *and the depth of* w *on* $\mathbb{T}_u$ *is* $\leq t_u - t_w$ **then** $\mathcal{I}_v = \mathcal{I}_v \cup \{u\}$;
for *any* $\mathcal{F} \subset \mathcal{I} \setminus (\mathcal{M} \setminus \mathcal{P})$ *with* $|\mathcal{F}| = N$ *and* $\mathcal{P} \subset \mathcal{F}$ **do**
  if $\cup_{v \in \mathcal{F}} \mathcal{I}_v = \mathcal{I}$ **then** $\mathcal{V}_s = \mathcal{V}_s \cup \{\mathcal{F}\}$;
return $\mathcal{V}_s$;

condition **S1** in $\tilde{\mathbf{X}}_\tau$. Since $\tilde{\mathbf{X}}_{\tau-1}$ is consistent with the observation according to the assumption, for each node $v \in \mathcal{V}$ with $I_v = 1$ and $\tilde{X}_{v,\tau-1} = 1$, the condition **S2** holds. Thus, we only need to discuss the $v \in \mathcal{I}$ with $\tilde{X}_{v,\tau-1} = 0$. For each of those nodes, v, the consistency requirement can be converted to the following three conditions:

C1 There exists $u \in \mathcal{I}$ with $T_u \neq -1$, $\tilde{X}_{u,\tau-1} = 0$ and $T_u \geq \tau$ such that

$$d(u, v) \leq t_v - T_u;$$

C2 There exists $u \in \mathcal{V}$ with $\tilde{X}_{u,\tau-1} = 1$ and

$$d(u, v) \leq t_v - \tau;$$

C3 There exists $u \in \tilde{\mathcal{S}}_{\tau-1} \cap \mathcal{I}$ with $T_u = -1$ and

$$d(u, v) = t_v - \tau.$$

Here $d(u, v)$ represents the length of the shortest ITF-path between node u and v. And $t_v = T_v$ if $T_v \neq -1$, otherwise, $t_v = T$.

For any node $v \in \mathcal{I}$ with $\tilde{X}_{v,\tau-1} = 0$, assume condition **C1** holds at $\tau - 1$, which means for node v, there exists some $u \in \mathcal{I}$ with $T_u \neq -1$, $\tilde{X}_{u,\tau-2} = 0$, and

$$d(u, v) \leq t_v - T_u.$$

If $\tilde{X}_{u,\tau-1} = 0$, condition **C1** is satisfied at τ. Otherwise, if $\tilde{X}_{u,\tau-1} = 1$, which means $T_u = \tau - 1$, we have

$$d(u, v) \leq t_v - T_u = t_v - t + 1. \tag{4.12}$$

Thus, there exists a node w with either $\tilde{X}_{w,\tau-1} = 1$ or $\tilde{X}_{w,\tau-1} = 0$ on the shortest ITF-path between u and v such that $d(w, v) \leq t_v - t$, which means either **C2** or **C3** is satisfied at τ for node v.

Assume **C2** holds for node v at $\tau - 1$, which means there exists some node u with $\tilde{X}_{u,\tau-2} = 1$ and

$$d(u, v) \leq t_v - \tau + 1.$$

Then there exists a node w, which is the neighbor of u on the shortest ITF-path between u and v such that

$$d(w, v) \leq t_v - \tau.$$

Thus, either **C2** or **C3** is satisfied at τ.

If **C1** or **C2** holds for node v at time $\tau - 1$, then we have node v is consistent with the observation at τ from the previous discussion. Now, we assume only **C3** holds at $\tau - 1$ for node v. Then, there exists some set $\mathcal{D} \subset \tilde{\mathcal{S}}_{\tau-2} \cap \mathcal{I}$ such that for any $u \in \mathcal{D}$,

$$d(u, v) = t_v - \tau + 1$$

and $T_u = -1$. Then, according to Algorithm 4.2, at least one node, $u \in \mathcal{D}$ needs to be infected at $\tau - 1$, which means $\tilde{X}_{u,\tau-1} = 1$. Then there exists a susceptible node w

which is the neighbor of u on the ITF-path between u and v such that $d(w, v) = t_v - \tau$, which means **C3** holds at τ for node v.

Therefore, from the above, we can see if $\tilde{\mathbf{X}}_{\tau-1}$ is consistent, $\tilde{\mathbf{X}}_{\tau}$ output by Algorithm 4.2 is consistent. □

4.2.2 Feasible Source Combinations

For infected node v with $T_v = -1$ or $T_v = 0$, define $\mathbf{1}^{(v)}$ to be a V-dimensional vector such that $\mathbf{1}^{(v)}_v = 1$ and $\mathbf{1}^{(v)}_m = 0$ for $m \neq v$. Define $\mathcal{P} = \{v | v \in \mathcal{I}, T_v = 0\}$. Then, we use $\mathbf{1}^{(v)}$ as the initial state to run the Single-Step Reconstruction algorithm, but instead of generating the network state at time slot 1, we find a set of infected nodes, $\mathcal{I}_v$, such that for each $u \in \mathcal{I}_v$, there exists a path from v to u such that node u can be infected at its observed infection time. For any combination of N infected nodes with unknown infection time or infection time 0, $\mathcal{F}$, we define $\mathcal{V}_s$ such that $\mathbf{1}^{\mathcal{F}} \in \mathcal{V}_s$ if only if $\cup_{v \in \mathcal{F}} \mathcal{I}_v = \mathcal{I}$ and $\mathcal{P} \subset \mathcal{F}$. Here $\mathcal{V}_s$ is a set of initial states. The pseudocode can be found in Algorithm 4.3.

Lemma 4.3. *$\tilde{\mathbf{X}}_0 \in \mathcal{V}_s$ if only if $\tilde{\mathbf{X}}_0$ is consistent with the observation.*

Proof. Assume $\tilde{\mathbf{X}}_0 = \mathbf{1}^{(\mathcal{F})}$, where $|\mathcal{F}| = N$ and $\mathcal{F} \subset \mathcal{V}$.

- $\tilde{\mathbf{X}}_0 \in \mathcal{V}_s$ implies $\tilde{\mathbf{X}}_0$ is consistent:

  This can be shown by contradiction. Suppose $\tilde{\mathbf{X}}_0$ is not consistent with the observation.

  - If **S1** does not hold for $\tilde{\mathbf{X}}_0$, which means for some node v with $v \notin \mathcal{I}$, we have $\tilde{X}_{v,0} = 1$. According to the construction of $\mathcal{V}_s$, only the observed infected nodes can be the sources. Thus, $\tilde{\mathbf{X}}_0 \notin \mathcal{V}_s$.
  - If **S2** does not hold for $\tilde{\mathbf{X}}_0$, which means for some node v with $T_v = 0$,

$X_{v,0} = 0$. Since the observed sources are included in each possible source combination, we have $\tilde{\mathbf{X}}_0 \notin \mathcal{V}_s$.

- Assume for some node v with $T_v > 0$, **S3** does not hold.
  - Suppose $\tilde{X}_{v,0} = 1$. According to the algorithm to construct $\mathcal{V}_s$, only infected nodes without infection time observed can be set as initial infected nodes, which means $\tilde{\mathbf{X}}_0 \notin \mathcal{V}_s$.
  - Suppose $\tilde{X}_{v,0} = 0$ while both **c1** and **c2** do not hold, which means there does not exist a path to make sure node v can be infected at time T_v. Thus, we have $v \notin \cup_{u\in\mathcal{F}}\mathcal{I}_u$, which means $\tilde{\mathbf{X}}_0 \notin \mathcal{V}_s$.
- Suppose for some node v with $T_v = -1$, **S4** does not hold. By using a similar argument we use for **S3**, we have $\tilde{\mathbf{X}}_0 \notin \mathcal{V}_s$.

- $\tilde{\mathbf{X}}_0$ is consistent implies $\tilde{\mathbf{X}}_0 \in \mathcal{V}_s$:

  Suppose $\tilde{\mathbf{X}}_0$ is consistent, we need to show $\cup_{v\in\mathcal{F}}\mathcal{I}_v = \mathcal{I}$. For any $u \in \mathcal{I}$, $\tilde{\mathbf{X}}_0$ is consistent means there exists a path from a node $w \in \mathcal{F}$ to u such that node u can be infected at T_u (if $T_u \neq -1$) or by time T (if $T_u = -1$). Then, according to the algorithm to construct $\mathcal{V}_s$, we have $u \in \mathcal{I}_w$. Thus, we have $\cup_{v\in\mathcal{F}}\mathcal{I}_v = \mathcal{I}$.

□

Theorem 4.2. *From the initial state $\tilde{\mathbf{X}}_0$, we can build a diffusion history consistent with the observation if only if the $\tilde{\mathbf{X}}_0 \in \mathcal{V}_s$.*

Proof.

- $\tilde{\mathbf{X}}_0 \in \mathcal{V}_s$ implies $\tilde{\mathbf{X}}$ is consistent:

  According Lemma 4.3, $\tilde{\mathbf{X}}_0$ is an initial state which is consistent with the observation. Then, by using Lemma 4.2, we can reconstruct a series of network state at different times, $\tilde{X}_\tau$ $(1 \leq \tau \leq T)$ which are consistent with the observation. Thus, we can reconstruct a consistent diffusion history.

- $\tilde{\mathbf{X}}$ is consistent implies $\tilde{\mathbf{X}}_0 \in \mathcal{V}_s$:

  This can be shown by contradiction. If $\tilde{\mathbf{X}}_0 \notin \mathcal{V}_s$, then according to Lemma 4.3, $\tilde{\mathbf{X}}_0$ is not consistent with the observation. Thus, according to our definition, the reconstructed diffusion history cannot be consistent with the observation.

□

Theorem 4.3. *Algorithm 4.1 outputs a diffusion history consistent with the observation with a worst-case computational complexity of* $O(V_I^{N+1}E_I)$*, where* N *is the number of sources.*

Proof. The first half of the theorem holds according to Theorem 4.2. We next analyze the complexity of our algorithm. Define $G_I(\mathcal{V}_I, \mathcal{E}_I)$ to be the infected subgraph of $G(\mathcal{V}, \mathcal{E})$.

- At first, we need to run Algorithm 4.3 to generate the set of feasible initial states. In Algorithm 4.3, the complexity of building MBFS trees is $O(V_I^2 E_I)$. Then, for each combinations of N infected nodes without infection time observed, we need to check whether it is a feasible source set. These operations have a complexity of $O(V_I^{N+1})$. Thus, the complexity of Algorithm 4.3 is $O(V_I^2 E_I + V_I^{N+1})$.

- Then for each feasible source set, we build the MBFS trees, whose complexity is $O(V_I E_I)$. In each single step reconstruction, we need to prune the tree first. For each MBFS-tree, after pruning, the nodes removed from the MBFS-tree at the current step will not appear in the MBFS-tree for the future single step reconstructions. Thus, for a specific MBFS-tree, the worse case complexity for pruning in the diffusion history reconstruction is $O(V_I)$. Since there are at most V_I MBFS-trees, the worst case complexity of pruning the MBFS-trees for a specific initial state is $O(V_I^2)$. In a similar way, the worst case complexity of

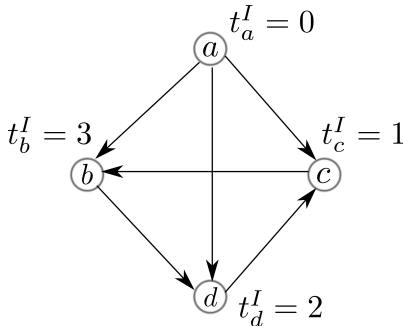


Figure 4.3: An example of diffusion. For $v \in \{a, b, c, d\}$, t_v^I is the infection time of node v. In this example, a is the source.

the greedy set cover is also $O(V_I^2)$ for a specific initial state. Thus, for each feasible source set, the complexity of Algorithm 4.2 in SSR is $O(V_I E_I + V_I^2)$. Since the number of feasible initial states is V_I^N in worst case, the complexity of Algorithm 4.2 in Algorithm 4.1 is $O(V_I^{N+1} E_I)$.

Therefore, the complexity of Algorithm 4.1 is $O(V_I^2 E_I + V_I^{N+1} + V_I^{N+1} E_I)$. Since $N \geq 1$, we have $V_I^{N+1} E_I \geq V_I^2 E_I$ and $V_I^{N+1} E_I \geq V_I^{N+1}$, which means the complexity is $O(V_I^{N+1} E_I)$.

□

4.3 Performance Evaluation

In this section, we compare the performance of SSR with other heuristics using following three performance measures.

- **Kendall's τ_b coefficient:** Since the diffusion history includes the infection time of each infected node in the network, we compare the infection order of the obtained diffusion history with the true infection order. Since there could be ties in the infection order because more than one nodes may be infected in the same time slot, we use Kendall's τ_b statistic Agresti (2010), which takes ties

into consideration. The value of τ_b varies from -1 to 1, where $\tau_b = 1$ means the two orders are in perfect agreement and $\tau_b = -1$ means the two orders are perfect inversion to each other.

- **Edge precision:** From a diffusion history, we can further infer the set of edges involved in the diffusion process. We call edge $u \to v$ a diffusion edge if node v was infected by node u in information diffusion. Under the SI model, given a diffusion history, each edge $u \to v$ satisfying $t_u^I < t_v^I$ is a possible diffusion edge. Define $\mathcal{E}_d$ to be the set of possible diffusion edges based on the true diffusion history $\mathbf{X}$ and $\tilde{\mathcal{E}}_d$ to be the set of possible diffusion edges based on the reconstructed diffusion $\tilde{\mathbf{X}}$. We consider the following performance metric:

$$P = \frac{|\mathcal{E}_d \cap \tilde{\mathcal{E}}_d|}{|\mathcal{E}_d|}.$$

There is few work on using partial infection time information to reconstruct the diffusion history. The only one in the literature which can be used in our setting is A_ILP developed in Fajardo and Gardner (2013). Therefore, we compare our algorithm with A_ILP, and two other heuristics: the breadth-first-search (BFS) heuristic and the infection-simulation (IS) heuristic.

- BFS: On the infected subgraph, we construct the breadth-first search (BFS) tree from each source combination and set the infection time of a node to its distance to the root. Then we consider the set of infected nodes with observed infection time, and compare its infection order on the breadth-first search tree with the actual infection order using Kendall's τ_b coefficient. The BFS tree with the largest τ_b is chosen to be the diffusion history.

- IS: For each source combination, we generate an infection sequence using the SI model on the original network. The diffusion stops when the diffusion process

"infects" all observed infected nodes. We again extract the infection order of the nodes with observed infection time and compare the order of it with the true infection order using Kendall's τ_b coefficient. The infection sequence with the largest τ_b is chosen as the diffusion history.

In Fajardo and Gardner (2013), the source is assumed to be known. However, in our setting, the source of the diffusion process is unknown. Therefore, when we implement the A_ILP algorithm, we try to reconstruct the diffusion path for each possible source and then choose the reconstructed diffusion path with the largest value of optimization objective derived in Fajardo and Gardner (2013) as the result of A_ILP.

We tested our algorithm on both synthetic diffusion data and real data. The networks used in generating the synthetic diffusion data include

- The power network: This network is used to represent the topology of the Western States Power Grid of the United States, which contains 4941 nodes and 6594 edges Watts and Strogatz (1998).

- The BA network: This is a network generated by using the Barabási-Albert model Barabasi and Albert (1999) with 300 nodes. Each new node is connected to 3 existing nodes.

- The IAS network: This is the Internet Autonomous Systems network Leskovec *et al.* (2005), which contains 10670 nodes and 22002 edges. This is a small-world network.

In our experiment, we first generated a diffusion sequence by using the discrete time SI model with an equal infection probability p for each edge and a set of randomly chosen sources. At time T, we took a snapshot of the network. Define s_{rate} to be the fraction of infected nodes with infection time observed. For example, if $s_{rate} = 20\%$, we randomly choose 20% of infected nodes and reveal their infection time.

We further evaluated the performance of our algorithm on the Weibo dataset provided by the WISE 2012 challenge [1] , which contains the data of Sina Weibo [2] , a famous microblogging website in China. The dataset consists of two parts: the friendship graph and a set of tweets.

Since each tweet in the dataset contains the post time, user id, retweet path and message id, we extracted the tweets for a specific message and considered the post time of each tweet as the infection time of that user.

We pre-processed the dataset as follows:

1. We added links used in the retweet path into the friendship graph to form the network of the diffusion.

2. We removed the nodes whose infection time is not consistent with the network.

3. We selected the weakly connected component formed by all the nodes with infection time, and the first infected node on this component is viewed as the source.

4. We calculated the average infection time according to

$$\bar{t} = \frac{\sum_{(u,v)\in\mathcal{E}, t_u^I < t_v^I} (t_v^I - t_u^I)}{m},$$

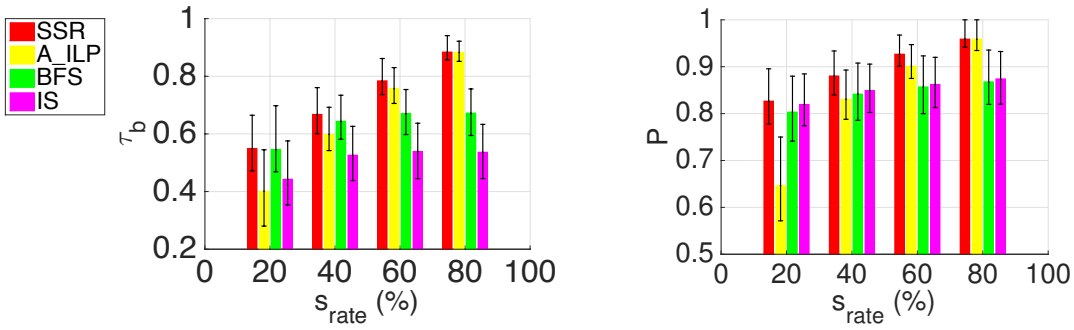
   where $\mathcal{E}$ is defined to be the set of edges on the weakly connected component and m is the number of directed edge (u, v) such that $t_u^I < t_v^I$. Here t_v^I is defined to be the infection time of node v.

5. We discretized the infection time according to

$$t'_u = \lceil \frac{t_u^I - t_s^I}{\bar{t}} \rceil,$$

[1] `http://www.wise2012.cs.ucy.ac.cy/challenge.html`

[2] `http://www.weibo.com`



(a) Average τ_b for different s_{rate} with 25-75 percentile

(b) Average P for different s_{rate} with 25-75 percentile.

Figure 4.4: Power Network with a single source, $p = 0.3$ and $T = 10$.

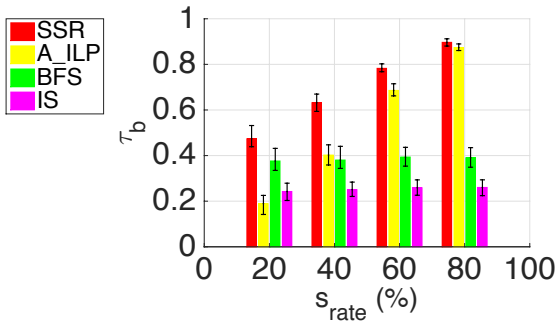
where s is the first infected node of the component.

6. We deleted the node whose adjusted infection time is not consistent with the network structure.

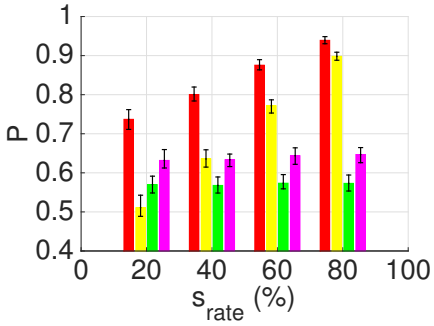
After these steps, we obtained 357 diffusion traces generated by a single source with an average size of 81.82 nodes/trace.

4.3.1 Performance Evaluation with Synthetic Diffusion Traces

Figure 4.4, Figure 4.5 and Figure 4.6 show the performance of Algorithm 4.1 and other algorithms based on the synthetic diffusion traces on different real-world networks with a single source. A_ILP has to solve a linear integer programming multiple times, and becomes very time-consuming on large-size networks such as the IAS network. So the performance of A_ILP is not included in Figure 4.6. In Power Network, BA Network and IAS Network, our algorithm has the best performance under most of the sample rates in terms of all the metrics, which proves that our algorithm is prominent under tree-like network and small-world network.

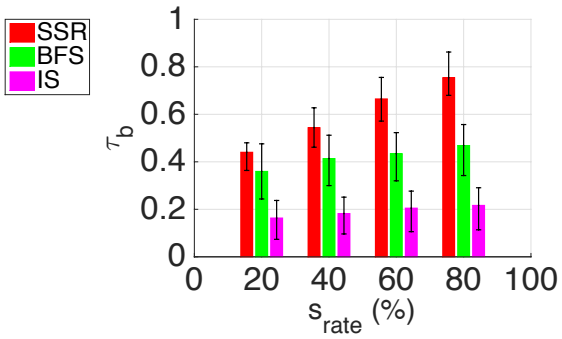


(a) Average τ_b for different s_{rate} with 25-75 percentile.

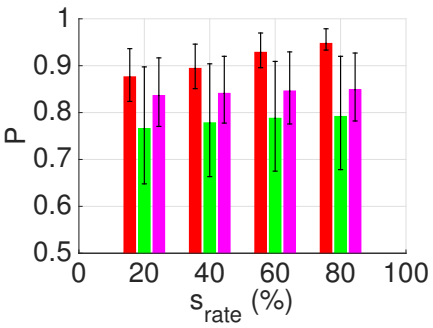


(b) Average P for different s_{rate} with 25-75 percentile.

Figure 4.5: BA Network with a single source, $p = 0.3$ and $T = 10$.



(a) Average τ_b for different s_{rate} with 25-75 percentile.



(b) Average P for different s_{rate} with 25-75 percentile.

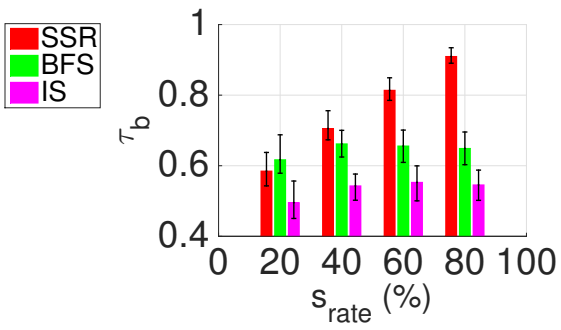
Figure 4.6: IAS Network with a single source, $p = 0.04$ and $T = 4$.

Furthermore, we test our algorithm for diffusion traces generated by multiple sources. For these diffusion traces, if the infected subnetwork generated by each source is not connected with each other, we can simply run a single source SSR algorithm on each component. Thus, we only used the diffusion traces where the infected subnetwork is a single connected component, in other words, the infected subnetworks from different sources are mixed. Figure 4.7 and Figure 4.8 provide the performance of SSR and other algorithms when the number of sources is 2. Similar

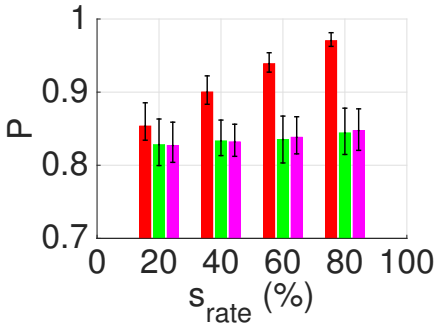
to the results from the single source case, our algorithm still outperforms the others under most of the sample rates. Figure 4.9 and Figure 4.10 compares the performance of SSR on the diffusion traces generated by 2 sources and 3 sources. We can see that, the performance of SSR when the number of sources is 3 is close to the case when the number of sources is 2, which means the performance of our algorithm is robust with different number of sources. In the other algorithms, without determining the feasible source combinations first, we need to test much more combinations of N nodes as the sources compared to our algorithm, which makes the other algorithms very time-consuming as N increases. Therefore, we did not include the results of the other algorithms when the number of sources is 3. Algorithm 4.3 is to find the all the feasible initial states that could generate a valid diffusion history. Without using Algorithm 4.3, we can still use Algorithm 4.2 to build a diffusion history for each source combination. However, the number of source combinations is proportional to V_I^N, which makes the algorithm almost impossible to run for multi-source diffusion processes. Table 4.2 and Table 4.3 shows the number of initial states with or without using Algorithm 4.3 for Power Network with $N = 2$ and 3. From the tables, we can see Algorithm 4.3 can reduce the number of initial states dramatically.

4.3.2 Performance Evaluation with the Weibo Dataset

In the experiment based on the Weibo dataset, since we do not have the infection probability, we set the infection probability to be 0.8 for each edge. The performance of our algorithm as well as other algorithms is shown in Figure 4.11. From Figure 4.11, we can see that our algorithm has the best performance under most sample rates in terms of τ_b and P.

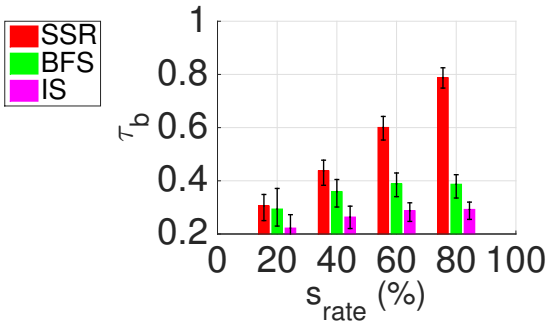


(a) Average τ_b for different s_{rate} with 25-75 percentile.

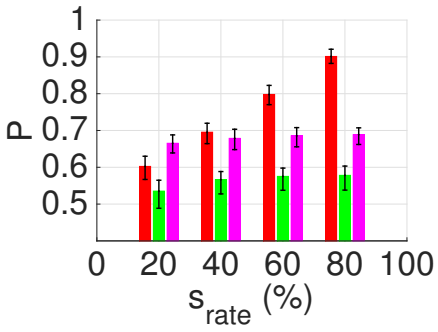


(b) Average P for different s_{rate} with 25-75 percentile.

Figure 4.7: Power Network with two sources, $p = 0.4$ and $T = 10$.



(a) Average τ_b for different s_{rate} with 25-75 percentile.

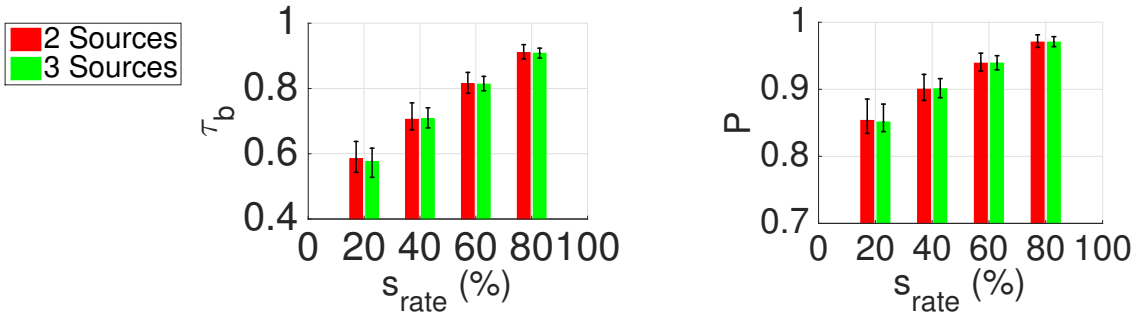


(b) Average P for different s_{rate} with 25-75 percentile.

Figure 4.8: BA Network with two sources, $p = 0.1$ and $T = 10$.

4.3.3 Optimality of the Single-Step Reconstruction

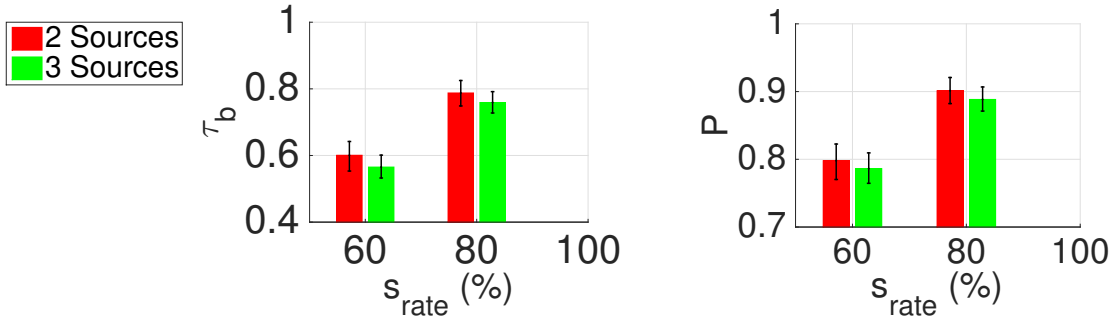
In the single-step reconstruction, we converted the problem to a weighted set cover problem, which is well-known to be NP-hard. Thus, we adopted the greedy set cover algorithm in Young (2008), which provides a worst-case approximation ratio guarantee. In this set of simulations, we compared SSR using the greedy set cover solution for single-step reconstruction with SSR using the optimal solution for single-



(a) Average τ_b for different s_{rate} with 25-75 percentile.

(b) Average P for different s_{rate} with 25-75 percentile.

Figure 4.9: Power Network $p = 0.4$ and $T = 10$.

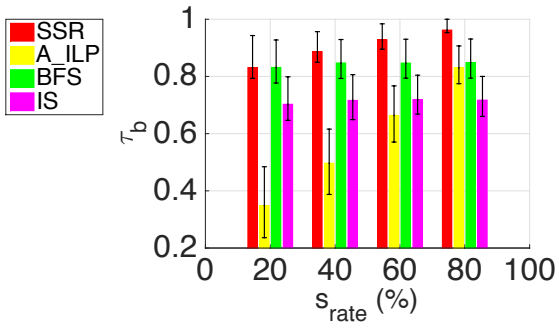


(a) Average τ_b for different s_{rate} with 25-75 percentile.

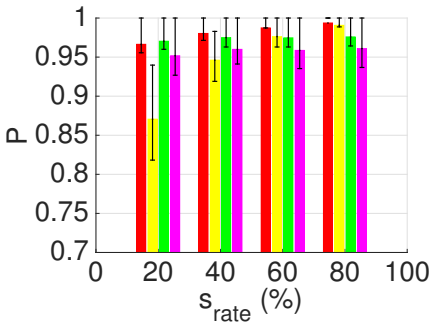
(b) Average P for different s_{rate} with 25-75 percentile.

Figure 4.10: BA Network with $p = 0.1$ and $T = 10$.

step reconstruction. Here, we used a small-size network, Zachary's Karate Club Network Zachary (1977), which has 34 nodes and 78 edges. Figure 4.12 shows the results of the comparison between the two algorithm. We can see that the results are almost identical, which shows that the greedy solution performs reasonably well, at least for small size networks.



(a) Average τ_b for different s_{rate} with 25-75 percentile.



(b) Average P_1 for different s_{rate} with 25-75 percentile.

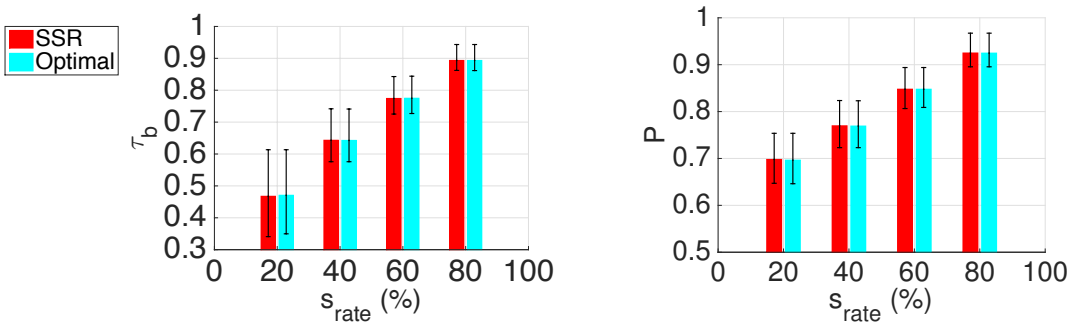
Figure 4.11: Weibo dataset.

Sample Rate	Without Algorithm 4.3	With Algorithm 4.3
20%	15855.78	266.86
40%	9066.33	48.97
60%	3951.05	11.91
89%	1060.31	3.30

Table 4.2: The average number of initial states with or without Algorithm 4.3 on Power Network with $N = 2$.

Sample Rate	Without Algorithm 4.3	With Algorithm 4.3
20%	4439775.98	7285.78
40%	1884810.87	528.19
60%	567404.51	64.69
89%	73913.57	9.19

Table 4.3: The average number of initial states with or without Algorithm 4.3 on Power Network with $N = 3$.



(a) Average τ_b for different s_{rate} with 25-75 percentile.

(b) Average P_1 for different s_{rate}.

Figure 4.12: Comparison with optimal on Zachery's Karate Club Network with a single source, $p = 0.3$ and $T = 10$.

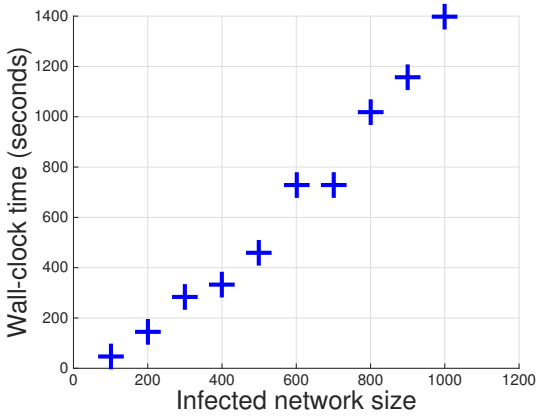
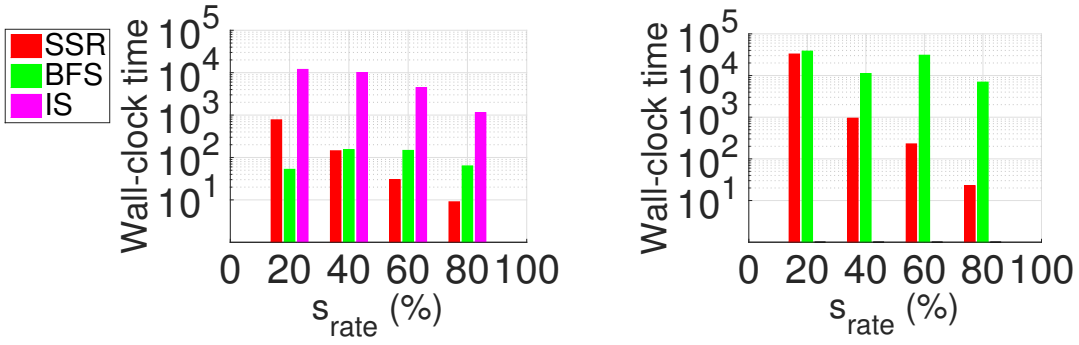


Figure 4.13: Wall-clock time vs infected network size.



(a) Wall-clock time when the number of sources is 2 for different s_{rate}.

(b) Wall-clock time when the number of sources is 3 for different s_{rate}.

Figure 4.14: Power Network with $p = 0.4$ and $T = 10$.

4.3.4 Efficiency Results

Figure 4.13 shows the wall-clock time of our algorithm versus the network size. In order to obtain Figure 4.13, at first, many diffusion traces were generated based on the power network Watts and Strogatz (1998) with a single source and infection time T varying from 5 to 90. Then we used the sample rate 40%. We tested the wall-clock time of our algorithm on these diffusion traces and classified the wall-clock time by the number infected nodes. For example, a diffusion trace with 150 infected nodes was included into the calculation of size 200. Finally, Figure 4.13 was obtained by calculating the average wall-clock time of each cluster and plotting the figure of wall-clock time versus the cluster size. From Figure 4.13, we can see that the running time increases in a near-linear trend with the size of infected subnetwork. Note that the x-axis is the size of the infected subnetwork, which is much smaller than the size of the power network Watts and Strogatz (1998).

Figure 4.14 provides the result of the wall-clock time versus the sample rate when the number of sources is 2 and 3. From the figure, we can see that as the sample rate increases, the running time of our algorithm decreases significantly and becomes

much faster than the other heursitics.

4.4 Conclusion

In this chapter, we studied the problem of diffusion history reconstruction. We formulated the problem as an optimization problem and developed a step-by-step reconstruction algorithm, in which the single-step reconstruction can be converted to a weight set cover problem. Our simulation results show the superior performance over heuristic and existing algorithms.

Chapter 5

INTERDEPENDENT NETWORK ROBUSTIFICATION: REAL-TIME REWIRING

Interdependent networks are widely used to model and analyze many real-world complex systems, such as the Internet, social networks, transportation systems, biochemical reactions, etc Albert and Barabási (2002); Di Muro *et al.* (2016). Interdependent networks consist of nodes and links where nodes represent different components of a complex system and links characterize the interaction between the components. Many researchers have studied interdependent networks to gain insights about features and properties of complex systems, such as their robustness, stability, connectivity and structure Yuan *et al.* (2015); Vespignani (2010); Buldyrev *et al.* (2010); Gao *et al.* (2012); Di Muro *et al.* (2016); Watts and Strogatz (1998); Albert *et al.* (2000); Albert and Barabási (2002); Callaway *et al.* (2000); Albert *et al.* (1999); Newman (2010); Schneider *et al.* (2011); Zeng and Liu (2012). An important topic in this area is to preserve the robustness of interdependent networks under site or link attacks. This problem is important to many real-world networks, such as power networks, transportation networks, fuel distribution networks and communication networks. Many of these networks may have low tolerance to damages on their structures (e.g. the diameter doubled after only 5% of the most connected nodes are removed on the scale-free network Albert *et al.* (2000)).

In an interdependent network, different networks (layers) interact with each other. For example, the nodes in the communication network need the power from the power stations of the power network, while the power stations need to communicate via the communication network.

We consider the problem of improving the robustness of an interdependent network under a localized attack. We focus on the development of algorithms that rewire the links of the interdependent networks in real-time to minimize the impact of the localized attack. In particular, we assume an interdependent network consists of two subnetworks A and B, in which the functioning of each node can depend on a set of nodes from the other layer. We study a localized attack model such that the nodes around attacked nodes are removed hop by hop inspired by Shao *et al.* (2015). The main contributions of this chapter are summarized below.

- **Problem Formulation:** We formulate *the interdependent network link rewiring problem* as a Markov decision process (MDP) problem, and prove that the problem is NP-hard by reducing the maximum coverage problem to our MDP problem.

- **Algorithm:** We propose a greedy algorithm to rewire the links during the attack. The key idea of the algorithm is to maximize the objective of the MDP problem in a greedy manner.

- **Analysis:** We compare the performance of our algorithm with the exact solution of the MDP problem on a small network. The results show that the performance is close in terms of the objective of the MDP problem.

- **Empirical Evaluations:** We evaluate the performance of the algorithm on interdependent networks formed by the real networks including the air traffic network, the IAS network and the power grid network with simulated localized attacks. In most cases, when a large fraction of nodes in the networks are attacked, our algorithm outperforms others.

5.1 Problem Formulation

5.1.1 Interdependent Networks

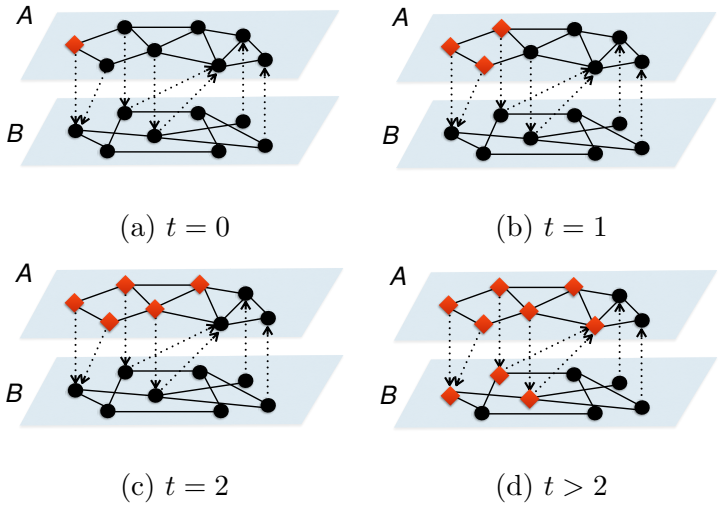
The chapter adopts the interdependent network model inspired by Buldyrev *et al.* (2010), which consists two networks, network A and network B, and a set of directed dependency links. Each dependency link is a directed link starting from a node in network A (B) and ending at a node in network B (A), which represents the influence of network A (B) on network B (A). Let $G_a(\mathcal{V}_a, \mathcal{E}_a)$ ($G_b(\mathcal{V}_b, \mathcal{E}_b)$) represent the topological structure of network A (B), where $\mathcal{V}_a$ ($\mathcal{V}_b$) is the set of vertices and $\mathcal{E}_a$ ($\mathcal{E}_b$) is the set of edges of graph G_a (G_b). Define $\mathcal{D}$ to be the set of dependency links that connect graph G_a and graph G_b. The interdependent network can be represented by the tuple $(G_a(\mathcal{V}_a, \mathcal{E}_a), G_b(\mathcal{V}_b, \mathcal{E}_b), \mathcal{D})$. For each vertex $v \in \mathcal{V}_b$ ($\mathcal{V}_a$) we can define its supporting node set to be

$$\mathcal{R}_v = \{u | \forall (u, v) \in \mathcal{D}\}. \tag{5.1}$$

Here $\mathcal{R}_v$ is a set of nodes from graph G_a (G_b), whose failures will impact the functioning of node v from G_b (G_a). In this chapter, we consider the case where a node v will stop functioning if all its supporting nodes in $\mathcal{R}_v$ from the other layer fail. Note that if $\mathcal{R}_v$ is empty, the functioning of node v does not depend on any other node from the other layer.

5.1.2 Localized Attack Model

We consider the localized attack model inspired by Yuan *et al.* (2015). The attack occurs and spreads on network A. In particular, at the beginning, a single node from network A is attacked and fails. Then, at each time slot, each attacked node at network A can attack and cause the failures of all its neighbors on network A. We assume the attack stops with probability p at each time slot before time T and stops



(a) $t = 0$ (b) $t = 1$

(c) $t = 2$ (d) $t > 2$

Figure 5.1: An example of the localized attack with $p = 0$ and $T = 2$. The red diamond represents failures. At time $t = 0$, a node in network A is attacked and fails. Then, the attack starts to propagate. At time $t = 1$, the neighbors of the previous attacked nodes in network A get attacked and fail. The similar procedure goes on at time $t = 2$. Then, the attack stops and there are 4 more node failures caused by the failures of their supporting nodes.

at time slot T if it does not stop before that. There are two kinds of failures during this attack:

- The failure directly resulted from attacks: A node stops functioning because it is attacked.
- The failures caused by failures: A node stops functioning because of the failures of all nodes from its supporting node set.

We assume failures directly due to the attack propagates much faster than the failures caused by other failures. Thus, the attack propagates on network A first and the second type failures happen after the attack stops as shown in the example in Figure 5.1. Consider an interdependent network where the communication network is the

first layer and the power grid is the second layer. In general, the cyber attack occurring on the communication network spreads very fast, while the failures ocurring in the power grid may take some time. For example, in the sequence of events that led to the cascading failures of the power grid in India 2012, the 2nd event ocurred one hour and twenty minutes after the 1st event and the 3rd event was 58 minutes after the 2nd event Bakshi *et al.* (2012). Motivated by this, we assume the propagation of failures due to the attack is much faster than the failures due to the loss of supporting nodes.

5.1.3 Markov Decision Process (MDP) Formulation

Markov decision processes (MDPs) have been widely used to model the control of stochastic systems. At each time step, the process in some state s moves to a new state s' given action a, which results in a certain reward. Now we try to explain the intuition of our algorithm by using the Markov decision process (MDP). Denote the interdependent network by the tuple

$$(G_a(\mathcal{V}_a, \mathcal{E}_a), G_b(\mathcal{V}_b, \mathcal{E}_b), \mathcal{D}).$$

Define $\mathcal{C}_t$ to be the set of nodes attacked and failed at time slot t on G_a, $\mathcal{F}_t = \cup_{i=0}^{t} \mathcal{C}_i$, and $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$ $(G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t))$ to be the topological graph of network A (B) before action is taken at time slot t. Thus, we have $G_a^0 = G_a$ and $G_b^0 = G_b$. Define the state at time slot t as a tuple

$$s_t = (G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t), G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t), \mathcal{C}_t, \mathcal{F}_t, \mathcal{D}).$$

The action a_t on state s_t can be considered as edge rewiring happened on G_a^t, where a number of

$$n_t = \min(\lfloor r||\mathcal{W}_t|| \rfloor, z) \tag{5.2}$$

edges are rewired, where

$$\mathcal{W}_t = \{(v, u) \in \mathcal{E}_a^t | v \in \mathcal{F}_t \text{ and } u \in \mathcal{V}_a^t \setminus \mathcal{F}_t\}$$

is the set of edges that connect the current attacked nodes with unattacked nodes in network A, and r, z are two constants while $0 < r < 1$ and $z \in \mathbb{Z}$. At each time slot, the number of rewired links is the smaller value of a constant z and the number of links connecting current attacked nodes and healthy nodes multiplying by a constant r $(0 < r < 1)$. We remark under the assumption above, the attacked nodes cannot be isolated from the rest of the network by cutting all the edges between attacked nodes and unattacked nodes. If it is possible, then a simple solution is to isolate the attacked nodes immediately. Denote by T the attack propagation time. We further assume at each time slot t $(t \geq 0)$, the diffusion of the attack stops with probability p.

At time slot 0, we have

$$s_0 = (G_a(\mathcal{V}_a, \mathcal{E}_a), G_b(\mathcal{V}_b, \mathcal{E}_b), \{source\}, \{source\}, \mathcal{D}),$$

where *source* represents the single attacked node at time slot 0. Given state

$$s_t = (G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t), G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t), \mathcal{C}_t, \mathcal{F}_t, \mathcal{D}) \ (t \geq 0),$$

and action a_t, define $G_a^{t\prime}(\mathcal{V}_a^{t\prime}, \mathcal{E}_a^{t\prime})$ to be the graph at time t after action a_t has been taken. Then, the state transitions can be described as the following:

1. If $\mathcal{C}_t = \emptyset$, we have $\mathcal{C}_{t+1} = \emptyset$, $\mathcal{F}_{t+1} = \mathcal{F}_t$, $G_a^{t+1} = G_a^{t\prime}$ and $G_b^{t+1} = G_b^t$.

2. If $\mathcal{C}_t \neq \emptyset$, we have $G_a^{t+1} = G_a^{t\prime}$ and $G_b^{t+1} = G_b^t$. Since with probability p, the attack stops at time slot t, and with probability $1 - p$, the attack continues, we have

$$\mathcal{C}_{t+1} = \begin{cases} \emptyset, \text{ with probability } p, \\ \{u | \forall u \in \mathcal{V}_a^{t\prime} \setminus \mathcal{F}_t, (v, u) \in \mathcal{E}_a^{t\prime}, v \in \mathcal{F}_t\}, \\ \text{with probability } 1 - p, \end{cases}$$

and $\mathcal{F}_{t+1} = \mathcal{F}_t \cup \mathcal{C}_{t+1}$.

Then we use $\mathcal{S}$ to represent the set of possible states and $\mathcal{A}$ to represent the set of actions. Define a reward function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that:

1. If $\mathcal{C}_t \neq \emptyset$, define

$$\mathcal{P}_t = \{u | \forall u \in \mathcal{V}_a^{t\prime} \setminus \mathcal{F}_t, (v, u) \in \mathcal{E}_a^{t\prime}, v \in \mathcal{F}_t\},$$

   where $\mathcal{P}_t$ represents the set of nodes that can be attacked in the next time slot. Define $\mathcal{O}_a^t$ to be the set of failed nodes caused by the failures of $\mathcal{F}_t \cup \mathcal{P}_t$ on network A and $\mathcal{O}_b^t$ to be the set failed nodes caused by the failures of $\mathcal{F}_t \cup \mathcal{P}_t$ on network B. Denote f_a^t as the size of the largest connected component of $G_a^{t\prime}$ after removing the nodes $\mathcal{F}_t \cup \mathcal{P}_t \cup \mathcal{O}_a^t$. Similarly, denote f_b^t to be the size of the largest connected component of G_b^t after removing nodes in $\mathcal{O}_b^t$. When $\mathcal{C}_t \neq \emptyset$, the reward function returns $f_a^t + f_b^t$.

2. Otherwise, it returns 0.

We remark that here we use the size of the largest connected component as the robustness metric. The value function at time t is

$$V_t(s_t) = \max_{a_t}(f(s_t, a_t) + \gamma \sum_{s_{t+1}} \mathbb{P}(s_{t+1}|s_t, a_t) V_{t+1}(s_{t+1}(s_t, a_t)). \tag{5.3}$$

In our case, we may set the discount factor $\gamma = 1$. Thus, we have

$$V_t(s_t) = \max_{a_t}(f(s_t, a_t) + \sum_{s_{t+1}} \mathbb{P}(s_{t+1}|s_t, a_t) V_{t+1}(s_{t+1}(s_t, a_t)). \tag{5.4}$$

Calculating the value function $V_0(s_0)$ is equivalent to solving the following problem Powell (2007):

Problem 1 (MDP problem)**.**

$$\max_{\pi} \mathbb{E}(\sum_{t=0}^{T} f(s_t, a_t)|s_0) \tag{5.5}$$

where π is a policy that decides which action to take at each state.

5.2 A Low-Complexity Algorithm

5.2.1 Challenges - NP-hardness

To solve the MDP problem, we have the following difficulties:

1. The cardinality of the state set and the action set are very large, which makes it impossible to solve the problem by using dynamical programming.

2. There is no closed form expression of function f.

Theorem 5.1. *The MDP problem defined in Problem 1 is NP-hard.*

Proof. The proof can be done by reducing the maximum coverage problem to Problem 1. Detailed proof can be found in Appendix C. □

5.2.2 Proposed Algorithm - ***REALWIRE***

Since the MDP problem is hard to solve, we propose a heuristic algorithm, **REAL WIRE**, based on the expected number of link failures. Define $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$ to be a directed subgraph of G_a^t at time t such that

$$\tilde{\mathcal{V}}_a^t = \mathcal{V}_a^t \setminus \mathcal{F}_{t-1}$$

and

$$\tilde{\mathcal{E}}_a^t = \cup_{v\in\mathcal{C}_t, u\in\mathcal{V}_a^t\setminus\mathcal{F}_t}\{(x,y)|\ (x,y)\in\mathcal{E}_a^t, \text{ and } (x,y) \text{ is on the shortest path between node } v \text{ and } u \text{ on graph } G_a^t.\}$$

Then, for $\forall v \in \tilde{\mathcal{V}}_a^t$, we define *the expected number of link failures* of node v at time t, f_v^t to be

$$f_v^t = \sum_{u\in\mathcal{N}_v^t} \frac{f_u^t}{d_{in}^t(u)} + (1-p)^{l_v^t}(d_{out}^t(v) + w_v^t + \frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}). \tag{5.6}$$

Here p is the stopping probability for the localized attack at each time slot, $\mathcal{N}_v^t$ is the set of successors of node v on graph g_a^t, $d_{in}^t(v)$ is the incoming degree of node v on g_a^t, $d_{out}^t(v)$ is the outgoing degree of node v on g_a^t, $d_a^t(v)$ is the degree of node v on subgraph of G_a^t with node set $\tilde{\mathcal{V}}_a^t$, and l_v^t is the level of node v on graph g_a^t, where

$$l_v^t = \min_{u \in \mathcal{C}_t} dist^t(u, v),$$

and $dist^t(u, v)$ is the distance from node u to v on graph g_a^t. In the definition of *the expected number of link failures*:

- $d_{out}^t(v)$ represents the link failures of outgoing links brought by the attack of node v on graph g_a^t.
- $\frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}$ is the number of link failures caused by the attack of node v on the edges with endpoints from the same level l_v^t on graph G_a^t. Here we divide it by 2 because we attribute half to each endpoint for each failed link.
- w_v^t represents the links failures caused by the second kind of node failures, which can be calculated according to Algorithm 5.4.
- $\frac{f_u^t}{d_{in}^t(u)}$ is the expected link failures brought by node u which is from level $l_v^t + 1$ whose shortest paths from the current attack nodes pass through node v.
- $(1-p)^{l_v^t}$ is the probability that the attack continues at level l_v^t.

In ***R*EAL*W*IRE**, we use *the total expected number of link failures*,

$$F_{t+1}(s_t, a_t) = \sum_{v \in \mathcal{V}_a^t \setminus \mathcal{F}_t} (1-p)^{l_v^t} (d_{out}^t(v) + w_v^t + \frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}) \tag{5.7}$$

to replace $V_{t+1}(S_{t+1}(S_t, a_t))$ in the equation (5.4), and the number of link failures brought by the current attack,

$$f_t(s_t, a_t) = \sum_{v \in \mathcal{C}_t} (d_{out}^t(v) + w_v^t + 1/2 \sum_{v \in \mathcal{C}_t} (d_a^t(v) - d_{out}^t(v)) \tag{5.8}$$

to replace $f(s_t, a_t(n_t))$. Thus, equation (5.4) becomes

$$\begin{aligned} F_t(s_t) &= \min_{a_t} f_t(s_t, a_t) + F_{t+1}(s_t, a_t) \\ &= \min_{a_t} \sum_{v \in \mathcal{V}_a^t \setminus \mathcal{F}_{t-1}} (1-p)^{l_v^t} (d_{out}^t(v) + w_v^t + \frac{d_a^t(v) - d_{in}^t(v) - d_{out}^t(v)}{2}) \end{aligned} \tag{5.9}$$

REALWIRE can be divided into four steps:

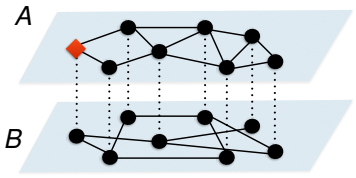
1. At each time slot t, we build the graph $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$ by using Algorithm 5.1. Figure 5.2 provides an example of how the algorithm works. Figure 5.2.a shows the network at time 0, while there is only one attacked node in network A. Then, the result of $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$ can be found in Figure 5.2.b.

2. We calculate the expected number of link failures for any node $v \in \tilde{\mathcal{V}}_a^t$ according to Equation 5.6. Figure 5.2.c shows the calculations of the expected number of link failures for the example.

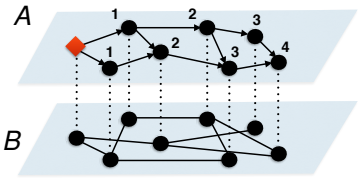
3. For each edge $(u, v) \in \tilde{\mathcal{E}}_a^t$, we assign a score $w_{u,v} = \frac{f_v^t}{d_{in}^t(v)}$. The result of link scores for the example is presented in Figure 5.2.d.

4. Pick the top n links with the highest scores. For each one of those links (u, v), reattach node v to another node on g_a^t on the highest level with a probability proportional to its degree according to Algorithm 5.2. Finally, the network after rewiring is in Figure 5.2.e.

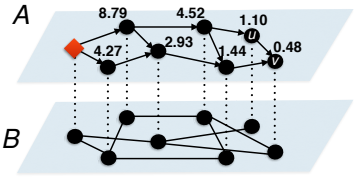
The pseudo code can be found in Algorithm 5.3. Higher the score an edge has, more shortest paths will be affected by removing the edge. Thus, in Algorithm 5.3, by choosing the link with the highest score to remove, ***REALWIRE*** aims at increasing the distances between the current attacked nodes and unattacked nodes. Then, we reattach the node to the lowest level in g_a^t so that the levels, l_v^t, for the nodes whose shortest paths will be impacted by the edge can be increased the most.



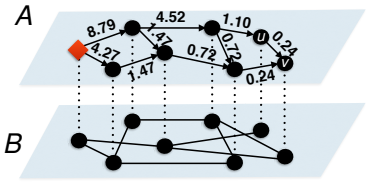
(a) In this example, assume at current time slot, in each subnetwork, there is only one attacked node, which is marked by red diamond. The dash lines between two layers are bidirectional dependency links.



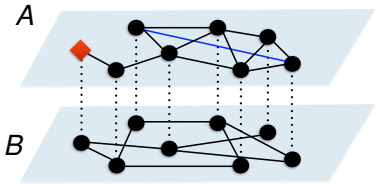
(b) In the first step, we run a breadth-first search from the current attacked node in subnetwork A. The number around each healthy node in subnetwork A represents the distance from current attack.



(c) Calculate the expected number of link failures for each healthy node in subnetwork A according to (5.6). For example, with assumption of $p = 0.3$, for node v, we have $f_v^t = 0.7^4 \times 2 = 0.48$. Then, we have $f_u^t = f_v^t/2 + 0.7^3 \times 2.5 = 1.10$.



(d) We assign the expected number of link failures of each node equally to the incoming links of these nodes.



(e) Finally, we choose links with highest scores to cut and reattach the endpoints to the nodes with the largest level number. In this example, assume we only can reattach one link. The link with score 8.79 is cut and the link in blue is added.

Figure 5.2: An example of our algorithm.

Thus, according to equation (5.9), our algorithm aims to minimize $F_t(s_t)$ in a greedy manner.

5.2.3 Complexity Analysis

Lemma 5.1. *The complexity of our algorithm is* $O(T|\mathcal{V}_a||\mathcal{D}| + T|\mathcal{E}_a|)$.

Proof. At each time slot, the complexity comes from three parts:

1. Calculating w_v^t for each node v, which contributes a complexity $O(|\mathcal{V}_a||\mathcal{D}|)$.
2. Finding the BFS subnetwork, which has a complexity $O(|\mathcal{V}_a| + |\mathcal{E}_a|)$.
3. Selecting the top n_t links with highest scores. This step has a complexity $O(|\mathcal{E}_a| \log n_t)$.

Since n_t is upper bounded by a constant, the complexity becomes $O(|\mathcal{V}_a||\mathcal{D}| + |\mathcal{E}_a|)$. Assume T is the duration of the attack, then the complexity of the algorithm is $O(T|\mathcal{V}_a||\mathcal{D}| + T|\mathcal{E}_a|)$.

□

Figure 5.3 plots the wall-clock time vs network size for the BA-BA networks with attack duration $T = 4$. From Figure 5.3, we can see the complexity has a near linear trend.

5.3 Performance Evaluation

5.3.1 Performance Measures

We use the following performance metrics to measure the performance:

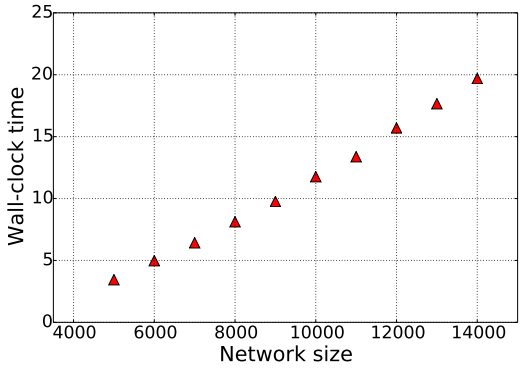


Figure 5.3: The wall-clock time vs network size for BA networks when attack time duration is 4.

Algorithm 5.1: BFS subnetwork

Input : Graph $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$, current attack $\mathcal{C}_t$, $\mathcal{F}_{t-1}$,
Output: $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$, level l_v^t, $\forall v \in \mathcal{V}_a^t$

$l_v^t \leftarrow 0,\ \forall v \in \mathcal{C}_t$;
$Current \leftarrow \mathcal{C}_t,\ Total \leftarrow \mathcal{C}_t$;
Let g_a^t be a null graph;
Add nodes $\mathcal{V}_a^t \setminus \mathcal{F}_{t-1}$ to graph g_a^t;
while $Current \neq \emptyset$ **do**
    $Next \leftarrow \emptyset$;
    for $v \in Current$ **do**
        for u *in neighbors of* v *on* G_a^t **do**
            if $u \notin Total$ *and* g_a^t *has node* u **then**
                Add directed link $v \rightarrow u$ to graph g_a^t;
                Add u to the set $Next$;
                $l_u^t \leftarrow l_v^t + 1$;
    $Current \leftarrow Next$;
    $Total \leftarrow Total \cup Current$;

Algorithm 5.2: Probabilistic Reattach

Input : Node v, graph $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$, $G_a^t(\mathcal{V}_a^t, \mathcal{V}_a^t)$, $G_b^t(\mathcal{V}_b^t, \mathcal{V}_b^t)$, l_u^t $\forall u \in \tilde{\mathcal{V}}_a^t$.
Output: $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$.

$ReattachSet \leftarrow \{u | u \in \tilde{\mathcal{V}}_a^t, l_u^t = \max_{u \in \tilde{\mathcal{V}}_a^t} l_u^t\} \setminus \{u | (u,v) \in \mathcal{E}_a^t\}$;
$d \leftarrow \sum_{u \in ReattachSet} d_a^t(u) + \sum_{w \in \mathcal{V}_b^t, \text{ where } u \in \mathcal{R}_w^t} \frac{d_b(w)}{||\mathcal{R}_w^t||}$;
Randomly pick a node u from $ReattachSet$ with probability $(d_a^t(u) + \sum_{w \in \mathcal{V}_b^t, \text{ where } u \in \mathcal{R}_w^t} \frac{d_b(w)}{||\mathcal{R}_w^t||})/d$ and add an edge between v and u;

Algorithm 5.3: ***REALWIRE***

Input : Graph $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$, $G_b^t(\mathcal{V}_b^t, \mathcal{E}_b^t)$, current attack $\mathcal{C}_t$, number of rewired links n, stopping probability p.
Output: $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$.

Construct graph $g_a^t(\tilde{\mathcal{V}}_a^t, \tilde{\mathcal{E}}_a^t)$ by using Algorithm 5.1;
Calculate f_v^t $\forall v \in \tilde{\mathcal{V}}_a^t$ according to Equation (5.6);
$w_{u,v} \leftarrow f_v^t / d_{in}^t(v)$ for $\forall (u,v) \in g_a^t$;
$W \leftarrow$ the n edges with highest $w_{u,v}$;
$i \leftarrow 1$;
for $1 \leq i \leq n$ **do**
    Remove edge (u,v) on G_a^t with $(u,v) = W[i]$;
    Reattach node v by using Algorithm 5.2;

Algorithm 5.4: Calculate w_v^t

Input : Graph $G_a^t(\mathcal{V}_a^t, \mathcal{E}_a^t)$, current attack $\mathcal{C}_t$, attacked nodes $\mathcal{F}_t$, node v, the set $VisitedEdges$, the score dictionary D
Output: A score

if $v \in \mathcal{V}_a^t$ **then** $D_v \leftarrow d_a^t(v)$;
else $D_v \leftarrow d_b(v)$;
for $(v,w) \in \mathcal{D}$ **do**
    if $(v,w) \notin VisitedEdges$ *and* $(w,v) \notin VisitedEdges$*;* **then**
        $VisitedEdges$.add$((v,w))$;
        if D *does not contain* D_w **then**
            Calculate D_w using Algorithm 5.4 based current $VisitedEdges$ and D;
        $D_v \leftarrow D_v + \frac{D_w}{||\{(u,w) | \forall (u,w) \in \mathcal{D}\}||}$;
Return D_v;

1. Largest connected component fraction: The size of the largest connected component of graph G_a (G_b) after removing all failed nodes divided by the original size of the graph.

2. Natural connectivity: It is defined in Jun *et al.* (2010) as

$$\bar{\lambda} = \ln(\frac{1}{N}\sum_{i=1}^{N} e^{\lambda_i}), \tag{5.10}$$

where λ_i is the i^{th} largest eigenvalue of the adjacency matrix of a graph. It can be used to measure the number of closed walks on the graph. Since in the interdependent networks, there are two networks G_a and G_b, after removing the failed nodes, we calculate the natural connectivity for each network and add them together.

3. Spectral radius: The spectral radius is defined as the largest absolution value of the eigenvalues of the adjacency matrix.

4. Spectral gap: The spectral gap of a graph is the difference between the largest and the second largest eigenvalues of the adjacency matrix. The spectral gap is closely related to the expansion properties of the graph. It has been used as a robustness metric in Malliaros *et al.* (2012).

5.3.2 Optimality on the Small Networks

For a small size network, we can calculate the optimal solution for the MDP problem defined in Problem 1. In this experiment, we used the Florentine families graph, which only contains 15 nodes. Since there are two networks, A and B, involved in the interdependent network model, we used the Florentine families graph to represent A (B) and randomly assign the name $0, 1, \ldots, 14$ to each node. Then, the functioning of one node in B depends on the node with the same name in A. Then, we let the

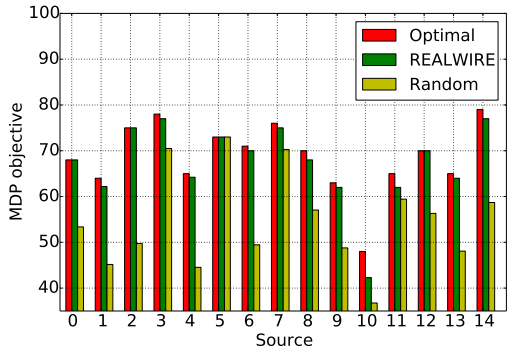


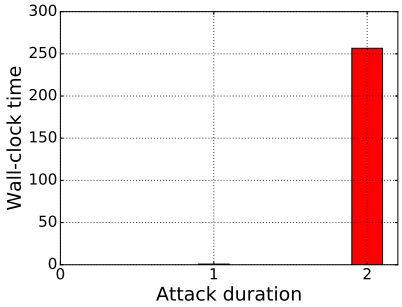
Figure 5.4: Florentine families network.

attack start from each node in the network A and propagate for 2 time slots. The result is listed in Figure 5.4. The bars of the optimal solution are generated based on the maximum objective value defined in (5.5) over all possible rewiring traces. For the other algorithms, we run 500 times for each source and take the average. Based on Figure 5.4, we can see that the value of the MDP objective generated by our algorithm is close to the optimal solution.

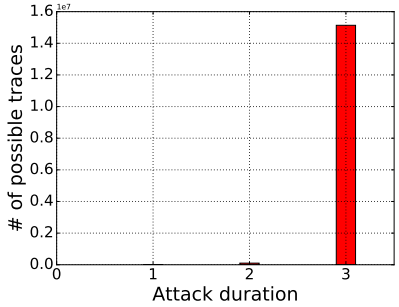
To calculate the optimal solution of the MDP problem, we need to consider every possible rewiring of the networks, which means as the increase of the propagation time , T, the computation time grows exponentially. For example, the average number of possible rewiring combinations over all possible attack sources is 312 when $T = 1$, 103342 when $T = 2$, and 13718030 when $T = 3$.The average of the wall-clock time or the number of possible rewirings versus attack duration is shown in Figure 5.5. From Figure 5.5, we can see the computation time grows exponentially as the increase of T, which makes the calculation of the optimal solution for large networks unpractical.

5.3.3 General Networks

We tested our algorithm on the following networks:



(a) Wall-clock time vs attack duration.



(b) Number of possible rewiring traces vs attack duration.

Figure 5.5: The complexity of the optimal algorithm on the Florentine families network.

1. BA network: The network is generated by using the Barabasi-Albert preferential attachment model, in which the number of edges to attach from a new node to the existing nodes is 3.

2. Air traffic network: The network is built based on one year (2016) of interval USA air traffic data [1] composed of 1243 airports connected by 16106 links.

3. IAS network: The IAS network is based on the Internet Autonomous Systems peering information inferred from oregon route-views Leskovec *et al.* (2005). The network contains 6474 nodes and 13895 edges [2] .

4. Power grid network:This network is used to represent the topology of the Western States Power Grid of the United States, which contains 4941 nodes and 6594 edges Watts and Strogatz (1998).

From each of the network mentioned above, we built the following interdependent

[1] `https://www.transtats.bts.gov/TRAFFIC/`

[2] `http://snap.stanford.edu/data/as.htmls`

networks.

- The BA-BA network: Both the first layer and second layer are BA networks with 2000 nodes generated randomly. We generated a one-to-one mapping between nodes from both layers as the dependency links.

- The IAS-PG network: For each node v in the IAS network or the power grid network, we uniformly picked an integer number d from 0 to 2 and randomly chose d nodes from the other layer as the supporting nodes of node v.

- The IAS-Air network: Since the communication network can impact the air traffic network and not the other way around, we generated a single direction interdependent network in this case. For each node v in the air traffic network, we randomly picked $0 \sim 2$ nodes from the other layer as the supporting node of node v.

We compared our algorithm with the following two heuristics:

- Single step rewiring: Instead of rewiring during the attack, it rewires the links before the attack. This algorithm aims to balance the degree of two endpoints connected by each edge.

  - For each link (u, v) on G_a, we define the degree difference to be $|d_a(u) - d_a(v)|$. We rank all the links according to their degree differences.

  - We pick the top n links with largest degree differences and remove those links.

  - For each removed link, we rewire the endpoint with the smaller degree to another node in the network with a small degree.

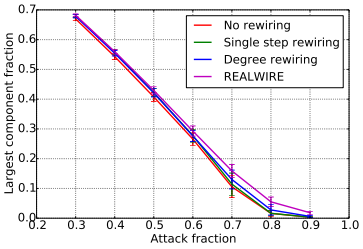
- Degree rewiring: This algorithm consists of the following steps:

	REALWIRE	Degree rewiring	Single step rewiring
BA-BA	126.81	130.34	299.55
IAS-PG	131.68	126.10	694.75
IAS-Air	134.78	119.63	694.74

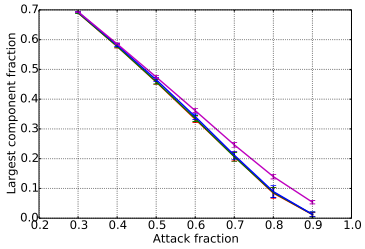
Table 5.1: The average number of rewired links.

- At each time slot t, we only rank all the links that connect attacked nodes and unattacked nodes and rank them according to a score based on the degree of their unattacked endpoints. For example, assume there is an edge $(u, v) \in G_a^t$, while $u \in \mathcal{F}_t$ and $v \in \mathcal{V}_a^t \setminus \mathcal{F}_t$. Then the score of edge (u, v) is $d_a^t(v) + \sum_{\text{while } v \in \mathcal{R}_w} \frac{d_b(w)}{|\mathcal{R}_w|}$, where $d_b(w)$ is the degree of node w on G_b, and $d_a^t(v)$ is the degree of node v on the subgraph of G_a^t with node set $\mathcal{V}_a^t \setminus \mathcal{F}_{t-1}$.
- We pick n_t links with highest scores to remove and then reattach the unattacked endpoint of each link to another unattacked with a higher score. Here n_t is defined in Equation (5.2).

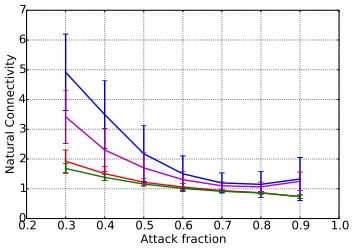
We evaluated our algorithm on a more general SI model, where at each time slot each attacked node in network A can attack its unattacked neighbors with certain probability p_a. In the experiments for the BA-BA network, $p_a = 1$, while for the IAS-Air network and the IAS-PG network, $p_a = 0.7$. The attack stops after a certain fraction of nodes gets attacked. For the number of rewired links at each time slot, we set $r = 0.5$ and $z = 20$ in Equation (5.2). For the single step rewiring algorithm, we chose to rewire 5% of the total number of links on network A. Table 5.1 shows the average number rewired links where for **REALWIRE** and degree rewiring, the numbers were calculated when the attack fraction is 0.9. From Table 5.1, the number



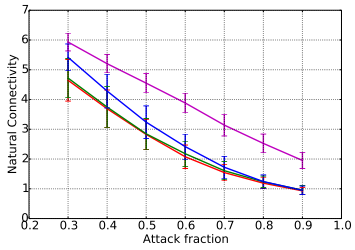
(a) Largest component fraction of network A.



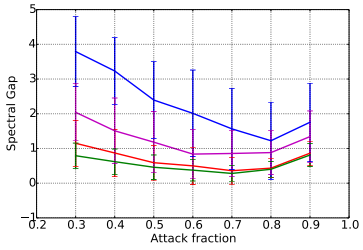
(b) Largest component fraction of network B.



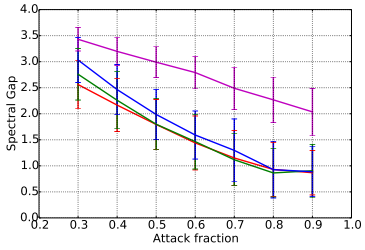
(c) Natural connectivity of network A.



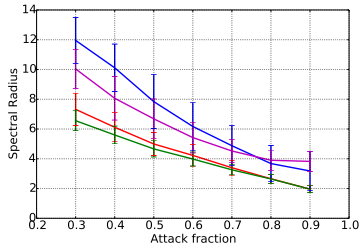
(d) Natural connectivity of network B.



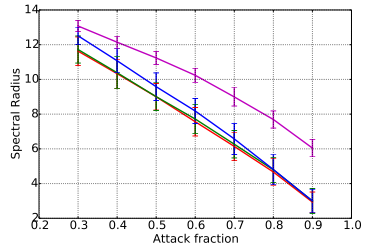
(e) Spectral gap of network A.



(f) Spectral gap of network B.

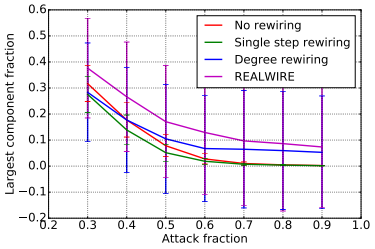


(g) Spectral radius of network A.

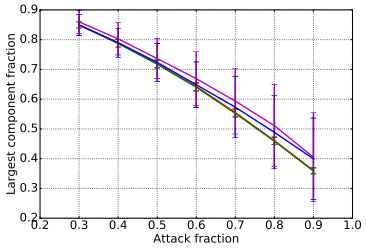


(h) Spectral radius of network B.

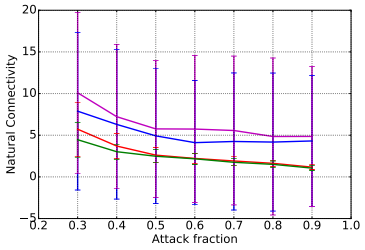
Figure 5.6: The BA-BA network.



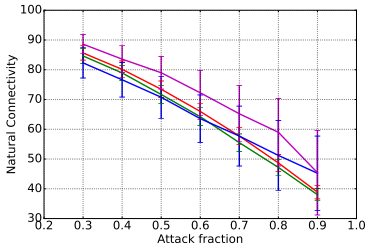
(a) Largest component fraction of network A.



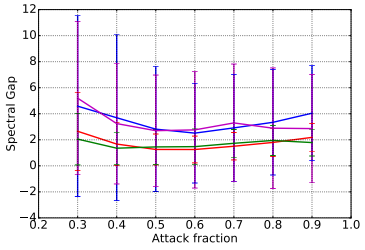
(b) Largest component fraction of network B.



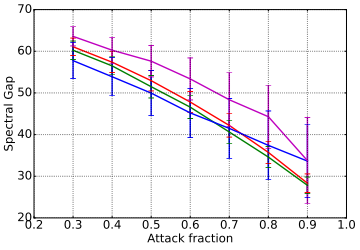
(c) Natural connectivity of network A.



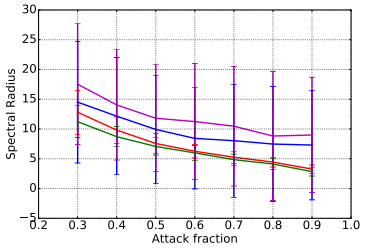
(d) Natural connectivity of network B.



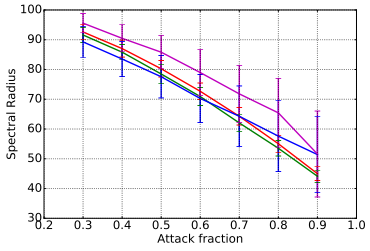
(e) Spectral gap of network A.



(f) Spectral gap of network B.

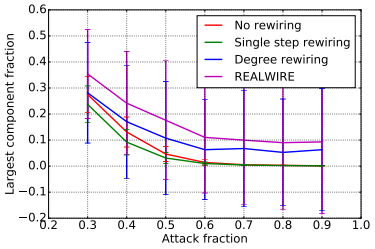


(g) Spectral radius of network A.

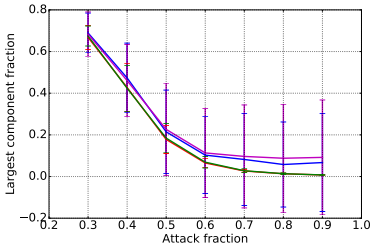


(h) Spectral radius of network B.

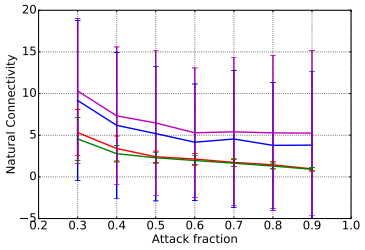
Figure 5.7: The IAS-Air network.



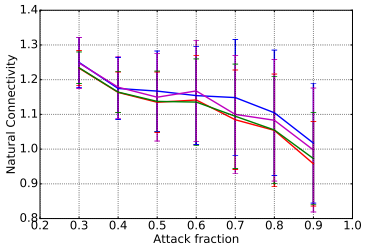
(a) Largest component fraction of network A.



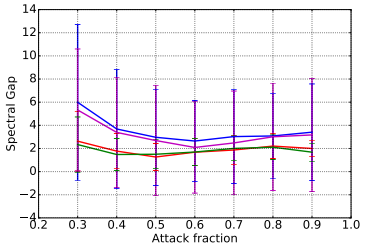
(b) Largest component fraction of network B.



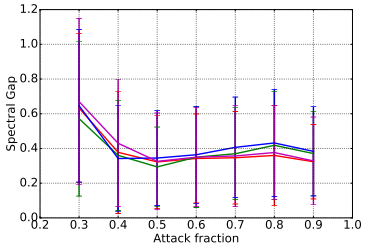
(c) Natural connectivity of network A.



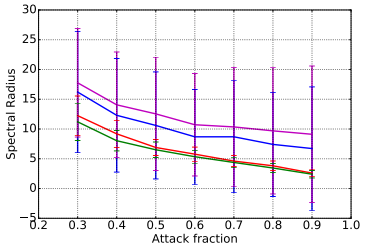
(d) Natural connectivity of network B.



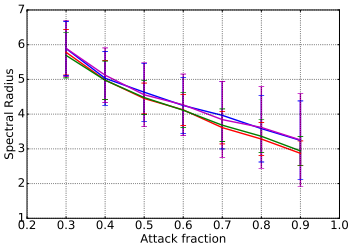
(e) Spectral gap of network A.



(f) Spectral gap of network B.

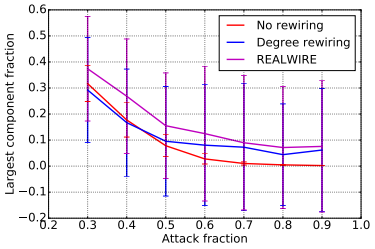


(g) Spectral radius of network A.

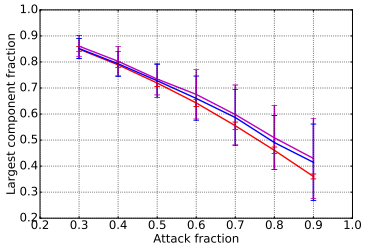


(h) Spectral radius of network B.

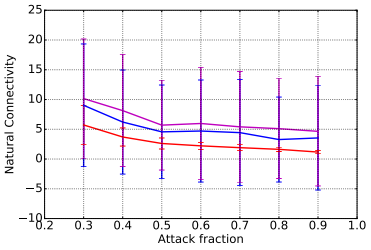
Figure 5.8: The IAS-PG network.



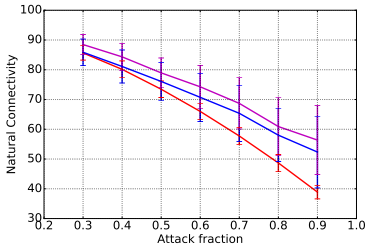
(a) Largest component fraction of network A.



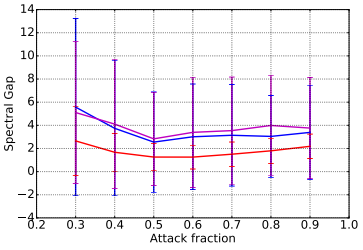
(b) Largest component fraction of network B.



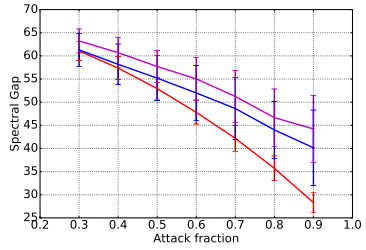
(c) Natural connectivity of network A.



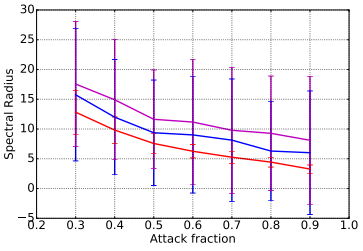
(d) Natural connectivity of network B.



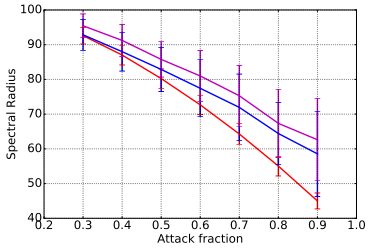
(e) Spectral gap of network A.



(f) Spectral gap of network B.



(g) Spectral radius of network A.



(h) Spectral radius of network B.

Figure 5.9: The IAS-Air network with backup links.

of rewired links resulted by $\boldsymbol{R}\mathbf{EAL}\boldsymbol{W}\mathbf{IRE}$ is close to the number of rewired links from the degree rewiring, which is much smaller than the number of links rewired by the single step rewiring algorithm. From Figure 5.3, we can see that the wall-clock time is almost linear to the size of the network. From Figure 5.6,5.7,5.8, we can see that in terms of the largest connected component fraction, our algorithm performs the best. For example, in the IAS-Air network, when the attack fraction is 0.3, the largest connected component fraction under $\boldsymbol{R}\mathbf{EAL}\boldsymbol{W}\mathbf{IRE}$ is 0.39, which is much higher than that under the degree rewiring, 0.28. In the BA-BA network, our algorithm outperforms others in terms on network B, while in network A, as the attack fraction increases, our algorithm starts to perform better than the degree heuristic. In the IAS-Air network, our algorithm outperforms the others for most metrics. In the IAS-PG network, the performance of our algorithm and the degree heuristic is close.

In reality, there exist circumstances, in which we cannot add a link between any two nodes in the network. This means we may not be able to rewire the links according to our algorithm. Thus, we considered another scenario, in which we assume each node has some backup links we can activate during the attack. In this experiment, for each node we randomly generated a number of backup links equal to 20% multiplying by its degree. So during the attack, for each link we cut, we can activate a backup link of one of its endpoints. The other parameters are the same as the rewiring case. From Figure 5.9, we can see our algorithm still outperforms the others.

5.4 Conclusion

In this chapter, we studied the problem of improving robustness of interdependent networks against the localized attack. We proposed a novel algorithm, named $\boldsymbol{R}\mathbf{EAL}$ $\boldsymbol{W}\mathbf{IRE}$, to improve the robustness of the interdependent networks and to limit the

impact of the attack by rewiring the links of the networks in real-time. We formulated the problem as an MDP problem, which has been proved to be NP-hard, and then proposed a greedy algorithm. The simulation results showed the performance of $\boldsymbol{REALWIRE}$ is close to the exact solution of the MDP problem in a small network and $\boldsymbol{REALWIRE}$ outperforms the others in different networks when the attack fraction is high.

Chapter 6

CONCLUSION

In this dissertation, we covered three problems related to information diffusion processes on networks: information source detection, diffusion history reconstruction, and network robustification with real-time rewiring.

In Chapter 2 and Chapter 3, we studied the information source detection problem. In Chapter 2, we investigated the information source detection problem with a complete snapshot under SIR model. We presented an algorithm, named clustering and localization, for tree networks. We proved that for a regular tree, the distance between any estimator and its closest real source is bounded by a constant with a high probability under some conditions. Then we extended our algorithm to general networks. In Chapter 3, we considered the information source detection problem under a more general setting, including a partial observation instead of the complete snapshot and the heterogeneous SIR model instead of the homogeneous one. We proposed a novel algorithm, named Optimal-Jordan-Cover (OJC), for locating multiple sources and showed theoretical guarantees on the detection rate for non-tree networks. Furthermore, we developed a heuristic based on the $K-$means, called Approximate-Jordan-Cover (AJC), to reduce the complexity of OJC.

In Chapter 4, we studied the problem of reconstructing the history of a diffusion process. Under SI model as the diffusion model and a partial snapshot as observation, we formulated the diffusion history reconstruction problem as a maximum a posteriori (MAP) estimate problem, and showed its NP-hardness. We proposed a greedy and step-by-step reconstruction algorithm to reconstruct the most likely network state at time slot based on the network state at time slot while guaranteeing the state

is consistent with the partial observation, and further developed a greedy algorithm for a single-step construction. We proved that the diffusion history obtained by the step-by-step reconstruction algorithm is always consistent with the observation.

In Chapter 5, we considered the problem of improving the robustness of an interdependent network under a localized attack. We formulated the interdependent network link rewiring problem as a Markov decision process (MDP) problem, which is NP-hard. Then, we proposed a greedy algorithm to rewire the links during the attack.

For each algorithm we proposed, an extensive amount of experiments on both synthetic networks and real-world networks have been done, which shows our algorithms outperform other algorithms.

Abrahao, B., F. Chierichetti, R. Kleinberg and A. Panconesi, "Trace complexity of network inference", in "Proc. of the 19th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining", pp. 491–499 (Chicago, USA, 2013).

Agresti, A., *Analysis of ordinal categorical data*, vol. 656 (John Wiley & Sons, 2010).

Albert, R. and A.-L. Barabási, "Statistical mechanics of complex networks", Rev. Mod. Phys. **74**, 47–97 (2002).

Albert, R., H. Jeong and A.-L. Barabási, "Internet: Diameter of the world-wide web", Nature **401**, 130–131 (1999).

Albert, R., H. Jeong and A.-L. Barabasi, "Attack and error tolerance of complex networks", Nature **406**, 378–382 (2000).

Bailey, N. T. J., *The mathematical theory of infectious diseases and its applications* (Hafner Press, 1975).

Bakshi, A., A. Velayutham, S. Srivastava, K. Agrawal, R. Nayak, S. Soonee and B. Singh, "Report of the enquiry committee on grid disturbance in northern region on 30th july 2012 and in northern, eastern & north-eastern region on 31st july 2012", New Delhi, India (2012).

Barabasi, A.-L. and R. Albert, "Emergence of scaling in random networks", Science **286**, 5439, 509–512 (1999).

Bikhchandani, S., D. Hirshleifer and I. Welch, "A theory of fads, fashion, custom, and cultural change in informational cascades", J. of Political Economy **100**, 5, 992–1026 (1992).

Briesemeister, L., P. Lincoln and P. Porras, "Epidemic profiles and defense of scale-free networks", in "Proc. of the 2003 ACM Workshop on Rapid Malcode", pp. 67–75 (Washington DC, USA, 2003).

Buldyrev, S. V., R. Parshani, G. Paul, H. E. Stanley and S. Havlin, "Catastrophic cascade of failures in interdependent networks", Nature **464**, 7291, 1025–1028 (2010).

Callaway, D. S., M. E. J. Newmann, S. H. Strogatz and D. J.Watts, "Network robustness and fragility: Percolation on random graphs", Phys. Rev. Lett. **85**, 5468–5471 (2000).

Chan, H., L. Akoglu and H. Tong, "Make it or break it: Manipulating robustness in large networks", in "Proceedings of the 2014 SIAM International Conference on Data Mining", pp. 325–333 (SIAM, 2014).

Chan, H., S. Han and L. Akoglu, "Where graph topology matters: the robust subgraph problem", in "Proceedings of the 2015 SIAM International Conference on Data Mining", pp. 10–18 (SIAM, 2015).

Chen, C., J. He, N. Bliss and H. Tong, "Towards optimal connectivity on multi-layered networks", IEEE Transactions on Knowledge and Data Engineering **29**, 10, 2332–2346 (2017).

Chen, Z., H. Tong and L. Ying, "Full diffusion history reconstruction in networks", in "2015 IEEE Int. Conf. on Big Data (Big Data)", pp. 707–716 (IEEE, 2015).

Chen, Z., H. Tong and L. Ying, "Robustification on-the-fly: Real-time rewiring on interdependent networks", Tech. rep., Arizona State University, available at `http://www.public.asu.edu/~zchen113/Publications/TechnicalReport.pdf` (2018).

Chen, Z., K. Zhu and L. Ying, "Detecting multiple information sources in networks under the SIR model", in "Proc. IEEE Conf. Information Sciences and Systems (CISS)", (Princeton, NJ, 2014).

Cohen, R., K. Erez, D. Ben-Avraham and S. Havlin, "Resilience of the internet to random breakdown", Phys. Rev. Lett. **85**, 4626–4628 (2000).

Di Muro, M., C. La Rocca, H. Stanley, S. Havlin and L. Braunstein, "Recovery of interdependent networks", Scientific reports **6** (2016).

Dong, W., W. Zhang and C. W. Tan, "Rooting out the rumor culprit from suspects", in "Proc. IEEE Int. Symp. Information Theory (ISIT)", pp. 2671–2675 (Istanbul, Turkey, 2013).

Fajardo, D. and L. M. Gardner, "Inferring contagion patterns in social contact networks with limited infection data", Networks and Spatial Econ. **13**, 4, 399–426 (2013).

Ganesh, A., E. Massouli and D. Towsley, "The effect of network topology on the spread of epidemics", in "Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)", pp. 1455–1466 (2005).

Gao, J., S. V. Buldyrev, H. E. Stanley and S. Havlin, "Networks formed from interdependent networks", Nature physics **8**, 1, 40–48 (2012).

Gardner, L. M., D. Fajardo and S. Travis Waller, "Inferring contagion patterns in social contact networks using a maximum likelihood approach", Natural Hazards Review **15**, 3 (2014).

Goldenberg, J., B. Libai and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth", Marketing Lett. **12**, 3, 211–223 (2001a).

Goldenberg, J., B. Libai and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth", Marketing Letters **12**, 3, 211–223 (2001b).

Gomez Rodriguez, M., J. Leskovec and A. Krause, "Inferring networks of diffusion and influence", in "Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining", pp. 1019–1028 (Washington DC, USA, 2010).

Gruhl, D., R. Guha, D. Liben-Nowell and A. Tomkins, "Information diffusion through blogspace", in "Proc. of the 13th Int. Conf. on World Wide Web", pp. 491–501 (New York, USA, 2004).

Hartigan, J. A. and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm", Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**, 1, 100–108 (1979).

Hayashi, Y., M. Minoura and J. Matsukubo, "Recoverable prevalence in growing scale-free networks and the effective immunization", arXiv:cond-mat/0305549 v2 (2003).

Hethcote, H. W., "The mathematics of infectious diseases", SIAM Review **42**, 4, 599–653 (2000).

Jun, W., M. Barahona, T. Yue-Jin and D. Hong-Zhong, "Natural connectivity of complex networks", Chinese Phys. Lett. **27**, 7, 078902 (2010).

Karamchandani, N. and M. Franceschetti, "Rumor source detection under probabilistic sampling", in "Proc. IEEE Int. Symp. Information Theory (ISIT)", (Istanbul, Turkey, 2013).

Kempe, D., J. Kleinberg and E. Tardos, "Maximizing the spread of influence through a social network", in "Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining", pp. 137–146 (Washington DC, USA, 2003).

Kermack, W. O. and A. G. McKendrick, "A contribution to the mathematical theory of epidemics", in "Proc. of the Royal Soc. of London A: Math., Physical and Eng. Sci.", vol. 115, pp. 700–721 (1927).

Kleinberg, J., "The small-world phenomenon: an algorithm perspective", in "Proceedings of the thirty-second annual ACM symposium on Theory of computing", p. 170 (ACM, 2000).

Leskovec, J., L. A. Adamic and B. A. Huberman, "The dynamics of viral marketing", ACM Trans. Web (TWEB) **1**, 1, 5 (2007).

Leskovec, J., J. Kleinberg and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations", in "Proc. of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining", pp. 177–187 (ACM, 2005).

Luo, W. and W. P. Tay, "Identifying multiple infection sources in a network", in "Proc. Asilomar Conf. Signals, Systems and Computers", (2012).

Luo, W. and W. P. Tay, "Estimating infection sources in a network with incomplete observations", in "Proc. IEEE Global Conference on Signal and Information Processing (GlobalSIP)", pp. 301–304 (Austin, TX, 2013a).

Luo, W. and W. P. Tay, "Finding an infection source under the SIS model", in "Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)", (Vancouver, BC, 2013b).

Luo, W., W. P. Tay and M. Leng, "Identifying infection sources and regions in large networks", IEEE Trans. Signal Process. **61**, 2850–2865 (2013).

Luo, W., W. P. Tay and M. Leng, "On the universality of jordan centers for estimating infection sources in tree networks", CoRR **abs/1411.2370**, URL `http://arxiv.org/abs/1411.2370` (2014).

Malliaros, F. D., V. Megalooikonomou and C. Faloutsos, "Fast robustness estimation in large social graphs: Communities and anomaly detection", in "Proc. of the 2012 SIAM International Conference on Data Mining", pp. 942–953 (SIAM, 2012).

Myers, S. and J. Leskovec, "On the convexity of latent social network inference", in "Advances in Neural Inform. Process. Syst.", pp. 1741–1749 (2010).

Newman, M., *Networks: An Introduction* (Oxford University Press, Inc., 2010).

Powell, W. B., *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703 (John Wiley & Sons, 2007).

Prakash, B. A., D. Chakrabarti, M. Faloutsos, N. Valler and C. Faloutsos, "Threshold conditions for arbitrary cascade models on arbitrary networks", in "IEEE 11th Int. Conf. on Data Mining (ICDM)", pp. 537–546 (Vancouver, Canada, 2011).

Prakash, B. A., H. Tong, N. Valler, M. Faloutsos and C. Faloutsos, "Virus propagation on time-varying networks: Theory and immunization algorithms", in "Mach. Learning and Knowledge Discovery in Databases", pp. 99–114 (2010).

Rényi, A. and P. Erdős, "On random graphs", Publicationes Mathematicae **6**, 290-297, 5 (1959).

Richardson, M. and P. Domingos, "Mining knowledge-sharing sites for viral marketing", in "Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining", pp. 61–70 (Edmonton, Canada, 2002).

Rozenshtein, P., A. Gionis, B. A. Prakash and J. Vreeken, "Reconstructing an epidemic over time.", in "KDD", pp. 1835–1844 (2016).

Schneider, C. M., A. A. Moreira, J. S. Andrade, S. Havlin and H. J. Herrmann, "Mitigation of malicious attacks on networks", Proc. the National Academy of Sciences **108**, 10, 3838–3841 (2011).

Sefer, E. and C. Kingsford, "Diffusion archaeology for diffusion progression history reconstruction", in "IEEE 14th Int. Conf. on Data Mining (ICDM)", pp. 530–539 (Shenzhen, China, 2014).

Shah, D. and T. Zaman, "Detecting sources of computer viruses in networks: Theory and experiment", in "Proc. Ann. ACM SIGMETRICS Conf.", pp. 203–214 (New York, NY, 2010).

Shah, D. and T. Zaman, "Rumors in a network: Who's the culprit?", IEEE Trans. Inf. Theory **57**, 5163–5181 (2011).

Shah, D. and T. Zaman, "Rumor centrality: a universal source detector", in "Proc. Ann. ACM SIGMETRICS Conf.", pp. 199–210 (London, England, UK, 2012).

Shao, S., X. Huang, H. E. Stanley and S. Havlin, "Percolation of localized attack on complex networks", New Journal of Physics **17**, 2, 023049 (2015).

Song, Y., A. Li, J. Huang, Y. Quan and L. Deng, "History path reconstruction analysis of topic diffusion on microblog", in "Recent Developments in Intelligent Systems and Interactive Applications", pp. 150–157 (Springer, 2016).

Sun, Y., C. Qian, N. Yang and P. S. Yu, "Collaborative inference of coexisting information diffusions", Arxiv preprint arXiv:1708.06890 (2017).

Tong, H., B. A. Prakash, T. Eliassi-Rad, M. Faloutsos and C. Faloutsos, "Gelling, and melting, large graphs by edge manipulation", in "Proceedings of the 21st ACM international conference on Information and knowledge management", pp. 245–254 (ACM, 2012).

Tong, H., B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos and D. H. Chau, "On the vulnerability of large graphs", in "IEEE 10th Int. Conf. on Data Mining (ICDM)", pp. 1091–1096 (Sydney, Australia, 2010).

Valler, N., B. A. Prakash, H. Tong, M. Faloutsos and C. Faloutsos, "Epidemic spread in mobile ad hoc networks: Determining the tipping point", in "Networking 2011", pp. 266–280 (2011).

Vespignani, A., "Complex networks: The fragility of interdependency", Nature **464**, 7291, 984–985 (2010).

Wang, Y., D. Chakrabarti, C. Wang and C. Faloutsos, "Epidemic spreading in real networks: An eigenvalue viewpoint", in "IEEE Proc. 22nd Int. Symposium on Reliable Distributed Syst.", pp. 25–34 (Florence, Italy, 2003).

Watts, D. J. and S. H. Strogatz, "Collective dynamics of 'small-world' networks", Nature **393**, 6684, 440–442 (1998).

Young, N. E., "Greedy set-cover algorithms (1974-1979, chvátal, johnson, lovász, stein)", Encyclopedia of Algorithms pp. 379–381 (2008).

Yuan, X., S. Shao, H. E. Stanley and S. Havlin, "How breadth of degree distribution influences network robustness: Comparing localized and random attacks", Phys. Rev. E **92**, 3, 032122 (2015).

Zachary, W. W., "An information flow model for conflict and fission in small groups", J. of Anthropological Research pp. 452–473 (1977).

Zeng, A. and W. Liu, "Enhancing network robustness against malicious attacks", Phys. Rev. E **85**, 6, 066130 (2012).

Zhang, Y., A. Adiga, S. Saha, A. Vullikanti and B. A. Prakash, "Near-optimal algorithms for controlling propagation at group scale on networks", IEEE Transactions on Knowledge and Data Engineering **28**, 12, 3339–3352 (2016).

Zhu, K., Z. Chen and L. Ying, "Locating the contagion source in networks with partial timestamps", Data Mining and Knowledge Discovery pp. 1–32 (2015).

Zhu, K., Z. Chen and L. Ying, "CatchEm All: Locating multiple diffusion sources in networks with partial observations", in "AAAI Conference on Artificial Intelligence", (2017).

Zhu, K. and L. Ying, "Information source detection in the SIR model: A sample path based approach", in "Proc. Information Theory and Applications Workshop (ITA)", (2013).

Zhu, K. and L. Ying, "A robust information source estimator with sparse observations", in "Proc. IEEE Int. Conf. Computer Communications (INFOCOM)", (Toronto, Canada, 2014).

Zhu, K. and L. Ying, "Information source detection in the SIR model: A sample path based approach", IEEE/ACM Trans. Netw. (2015a).

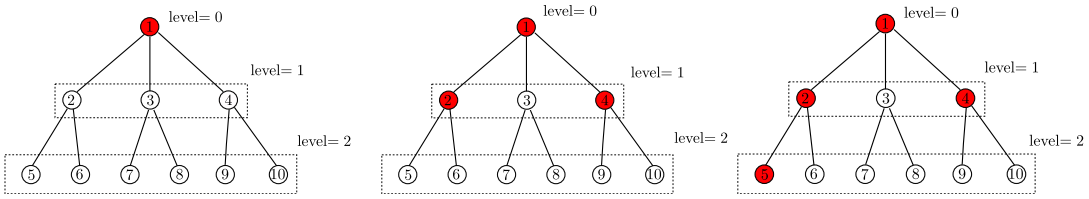
Zhu, K. and L. Ying, "Source localization in networks: Trees and beyond", arXiv:1510.01814 (2015b).

Zhu, K. and L. Ying, "Information source detection in networks: Possibility and impossibility results", in "Proc. IEEE Int. Conf. Computer Communications (INFOCOM)", (San Francisco, CA, 2016).

Zong, B., Y. Wu, A. K. Singh and X. Yan, "Inferring the underlying structure of information cascades", in "IEEE 12th Int. Conf. on Data Mining (ICDM)", pp. 1218–1223 (Brussels, Belgium, 2012).

APPENDIX A

PROOF OF CHAPTER 2



(a) Time $t = 0$. (b) Time $t = 1$. (c) Time $t = 2$.

Figure A.1: An example of one-time-slot branching process starting from node 1. For simplicity, we assume the time when node 1 gets infected is 0. From Figure A.1b, at $t = 1$, the current level is 0 and there are two nodes, 2 and 4, from level 1 infected by node 1 from level 0. Figure A.1c shows that at $t = 2$, node 5 from next level 2 is infected by node 2 from the current level 1. Therefore, at each time slot, there is at least one node infected at level, $l + 1$,by infected node from level l.

A.1 Proof of Theorem 2.1

Consider a $(g+1)$-regular tree $G(\mathcal{V}, \mathcal{E})$ with S different information sources, named $\zeta_1, \zeta_2, ..., \zeta_S$. These S original sources and the paths between each pair form a tree, named $G_s = (\mathcal{V}_{G_s}, \mathcal{E}_{G_s})$. Define event $\mathcal{A} = \bigcap_{i=1}^{S} \mathcal{A}_{\zeta_i}$, where $\mathcal{A}_{\zeta_i} (i = 1, ..., S)$ is the event that includes the following cases:

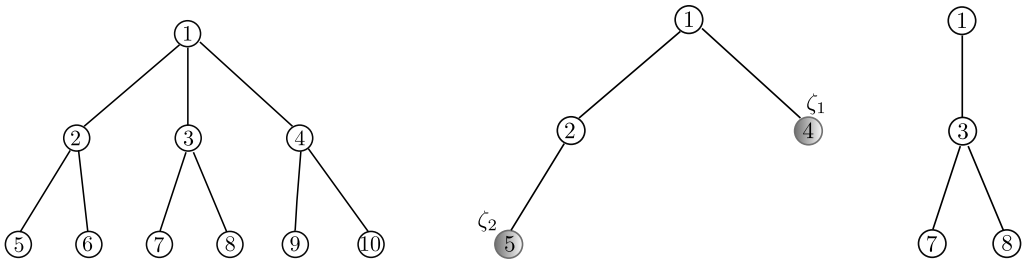
- Case 1: At least $(S+1)$ one-time-slot branching processes from source ζ_i survive after time t_0, where these $(S + 1)$ one-time-slot branching processes do not overlap.
- Case 2: The infection spreading tree starting from ζ_i terminates at or before time t_0. We describe this as the infection process from source ζ_i dies out at time t_0 on tree G.

We remark that t_0 is a constant. The one-time-slot branching process is defined to be the process that starting from an infected node, at each time slot, at least one node at the the next level, $l + 1$, gets infected by an infected node from the current level l. The level of a node is defined to be the distance between that node and the starting node of the process, while the current level is defined to be the largest level among all infected nodes associated with the starting node of the one-time-slot branching process at the beginning of the current time slot. In Figure A.1, an example is used to explain the definition of the one-time-slot branching process.

Then, we define a node set

$$\mathcal{V}_\alpha = \{\alpha |\ \alpha \in \mathcal{V}_{G_s} \text{ and } \min_{i=1,...,S} d(\alpha, \zeta_i) \leqslant C_1\},$$

in which each node is the on tree G_s and within C_1 ($C_1 > 0$ and $C_1 \in \mathbb{N}$) distance from at least one source, and $\mathcal{V}_\beta = \mathcal{V}_{G_s} \backslash \mathcal{V}_\alpha$, in which each node is on the tree G_s and in a distance larger than C_1 from any source. Define set $\mathcal{V}_S = \{\zeta_1, \zeta_2, ..., \zeta_S\}$ to be the set of original sources and we have $\mathcal{V}_S \subset \mathcal{V}_\alpha$. After that, define $m = |\mathcal{V}_\alpha|$



(a) G. (b) G_s. (c) T_1.

Figure A.2: An example of tree G, tree G_s and tree T_1. In this example, the original tree is the tree in Figure A.2a. Assume the original sources are $\zeta_1 = 4$ and $\zeta_2 = 5$. Then tree G_s formed by the orginal sources and paths between each pair of them is shown in Figure A.2b. And Tree T_1, which starts from root 1 without edges on G_s, is shown in Figure A.2c.

and $n = |\mathcal{V}_\beta|$. Without loss of generality, we assume $\mathcal{V}_\alpha = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ and define $\mathcal{V}_\beta = \{\beta_1, \beta_2, ..., \beta_n\}$.

Define T_a to be a subtree on G, which starts from root a without edges on G_s. To better understand the definition of tree G_s and T_a, an example is provided in Figure A.2. We further define $\mathcal{A}_a$ to be the event that on the tree T_a there are at least $(S+1)$ one-time-slot branching processes survived after time t_0 or all infection processes die out at time t_0. Then we have

$$\begin{aligned}
& P(\mathcal{A}) \\
&= P(\bigcap_{i=1}^{S} \mathcal{A}_{\zeta_i}) \\
&\geq P(\mathcal{A}_{\alpha_1}...\mathcal{A}_{\alpha_m}\mathcal{A}_{\beta_1}...\mathcal{A}_{\beta_n}) \\
&= \sum_{\mathbf{t}^I, \mathbf{t}^R} P(\mathcal{A}_{\alpha_1}...\mathcal{A}_{\alpha_m}\mathcal{A}_{\beta_1}...\mathcal{A}_{\beta_n}|\mathbf{t}^I, \mathbf{t}^R) P(\mathbf{t}^I, \mathbf{t}^R) \\
&> \sum_{(\mathbf{t}^I, \mathbf{t}^R) \in M_1} P(\mathcal{A}_{\alpha_1}...\mathcal{A}_{\alpha_m}\mathcal{A}_{\beta_1}...\mathcal{A}_{\beta_n}|\mathbf{t}^I, \mathbf{t}^R) P(\mathbf{t}^I, \mathbf{t}^R),
\end{aligned} \tag{A.1}$$

where

$$\mathbf{t}^I = (t^I_{\alpha_1}, ..., t^I_{\alpha_m}, t^I_{\beta_1}, ..., t^I_{\beta_n}),$$
$$\mathbf{t}^R = (t^R_{\alpha_1}, ..., t^R_{\alpha_m}, t^R_{\beta_1}, ..., t^R_{\beta_n})$$

and $M_1 = \{(\mathbf{t}^I, \mathbf{t}^R) |\ \forall a \in \mathcal{V}_{G_s}, t^I_a \leq C_1 \text{ or } t^I_a = \infty\}$. Define event

$$\tilde{\mathcal{A}} = \bigcap_{a \in V_{G_s}} \mathcal{A}_a \bigcap \{(\mathbf{t}^I, \mathbf{t}^R) \in M_1\},$$

which is the event of $\mathcal{A}$ restricted to M_1.

Lemma A.1. *For any $\epsilon > 0$, there exist some constants C_1 and t_0 such that $P(\tilde{\mathcal{A}}) > 1 - \epsilon$, if the distance between any two sources is larger than $2C_1$.*

Proof. According to the definition of $\tilde{\mathcal{A}}$, we have

$$P(\tilde{\mathcal{A}}) = \sum_{(\mathbf{t}^I, \mathbf{t}^R) \in M_1} P(\mathcal{A}_{\alpha_1} ... \mathcal{A}_{\alpha_m} \mathcal{A}_{\beta_1} ... \mathcal{A}_{\beta_n} | \mathbf{t}^I, \mathbf{t}^R) P(\mathbf{t}^I, \mathbf{t}^R). \tag{A.2}$$

To obtain a lower bound on $P(\tilde{\mathcal{A}})$, we need to analyze $P(\mathcal{A}_{\alpha_1} ... \mathcal{A}_{\alpha_m} \mathcal{A}_{\beta_1} ... \mathcal{A}_{\beta_n} | \mathbf{t}^I, \mathbf{t}^R)$ when $(\mathbf{t}^I, \mathbf{t}^R) \in M_1$ and $P((\mathbf{t}^I, \mathbf{t}^R) \in M_1)$ separately.

For $P((\mathbf{t}^I, \mathbf{t}^R) \in M_1)$, we have

$$P((\mathbf{t}^I, \mathbf{t}^R) \in M_1) = 1 - P((\mathbf{t}^I, \mathbf{t}^R) \in M_1^c), \tag{A.3}$$

where M_1^c is the complementary set of M_1. Define event $\mathcal{C} = \{(\mathbf{t}^I, \mathbf{t}^R) | \ \exists \beta \in \mathcal{V}_\beta, t_\beta^I > 0 \text{ and } t_\beta^I \neq \infty\}$. Since the probability of event $\mathcal{M}$ is difficult to analyze directly, we define $\mathcal{M}_1 = M_1^c \cap \mathcal{C}$ and $\mathcal{M}_2 = M_1^c \cap \mathcal{C}^c$, i.e.,

$$P(M_1^c) = P(\mathcal{M}_1) + P(\mathcal{M}_2) \tag{A.4}$$

Before discussing $P(M_1^c)$, we analyze when the infection process on a path is going to stop. From Figure A.3, we know that during each time slot the infection process on a path stops only when the recently infected node is recovered before the node next to it becomes infected. The probability for this event to happen is $p(1-q)$. Use $(\zeta_j \to a)$ to represent the event that node a is associated with ζ_j and $a_{\{t=k,\zeta_j\}}$ to represent the event that infection process from source ζ_j to a does not stop at time slot k. Define $p_s = 1 - p(1-q)$ and we have

$$\begin{aligned}
& P(t_a^I > C_1 \text{ and } t_a^I \neq \infty | \zeta_j \to a) \\
\leq\ & P(a_{\{t=1,\zeta_j\}} \cap a_{\{t=2,\zeta_j\}} \cap \cdots \cap a_{\{t=C_1,\zeta_j\}} | \zeta_j \to a) \\
=\ & P(a_{\{t=1,\zeta_j\}} | \zeta_j \to a) P(a_{\{t=2,\zeta_j\}} | a_{\{t=1,\zeta_j\}}, \zeta_j \to a) \dots \\
& P(a_{\{t=C_1,\zeta_j\}} | a_{\{t=C_1-1,\zeta_j\}}, \zeta_j \to a) \\
=\ & (p_s)^{C_1}.
\end{aligned} \tag{A.5}$$

Therefore, we have

$$\begin{aligned}
& P(t_a^I > C_1 \text{ and } t_a^I \neq \infty) \\
=\ & \sum_{i=1}^{S} P(t_a^I > C_1 \text{ and } t_a^I \neq \infty | \zeta_i \to a) P(\zeta_i \to a) \\
\leq\ & \sum_{i=1}^{S} (p_s)^{C_1} P(\zeta_i \to a) \\
\leq\ & (p_s)^{C_1}.
\end{aligned} \tag{A.6}$$

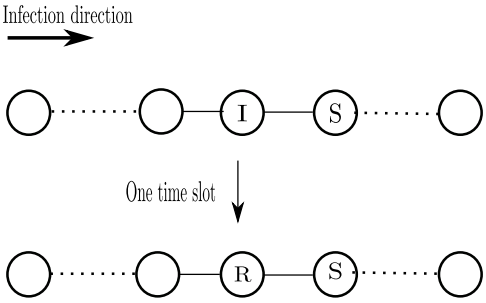


Figure A.3: The situation of the stop of infection process.

Next, we analyze $P(\mathcal{M}_1)$. Define set

$$\mathcal{V}_b = \{b | b \in \mathcal{V}_\beta \text{ and } \exists j, \text{ s.t. } d(\zeta_j, b) = C_1 + 1\},$$

and for any $\beta \in \mathcal{V}_\beta$, since $d(\beta, \zeta_i) > C_1$ for any $i = 1, ..., S$, we have $t_\beta^I > C_1$ as long as $t_\beta^I \neq \infty$. Thus, $\mathcal{C} \subset M_1^c$, and we have

$$\begin{aligned} P(\mathcal{M}_1) &= P(M_1^c \cap \mathcal{C}) \\ &= P(\mathcal{C}) \\ &\overset{\text{(a)}}{=} P(\bigcup_{b \in \mathcal{V}_b} \{t_b^I > C_1 \text{ and } t_b^I \neq \infty\}) \\ &\leq \sum_{b \in \mathcal{V}_b} P(t_b^I > C_1 \text{ and } t_b^I \neq \infty) \\ &\leq |\mathcal{V}_b|(p_s)^{C_1} \\ &\leq S(S-1)(p_s)^{C_1}, \end{aligned} \tag{A.7}$$

where (a) holds because $\beta \in \mathcal{V}_\beta$ infected by the information from source ζ_i along the path that contains node b and satisfies $d(b, \zeta_i) = C_1 + 1$.

For $P(\mathcal{M}_2)$, we have

$$\begin{aligned} P(\mathcal{M}_2) &= P(M_1^c \cap \mathcal{C}^c) \\ &< P(\{(\mathbf{t}^I, \mathbf{t}^R) | \exists \alpha \in \mathcal{V}_\alpha, \ t_\alpha^I > C_1 \text{ and } t_\alpha^I \neq \infty\}) \\ &= P(\bigcup_{\alpha \in \mathcal{V}_\alpha} \{t_\alpha^I > C_1 \text{ and } t_\alpha^I \neq \infty\}) \\ &\leq \sum_{\alpha \in \mathcal{V}_\alpha} P(t_\alpha^I > C_1 \text{ and } t_\alpha^I \neq \infty) \\ &\overset{\text{(a)}}{<} \sum_{\alpha \in \mathcal{V}_\alpha} (p_s)^{C_1} \\ &= m(p_s)^{C_1}, \end{aligned} \tag{A.8}$$

where (a) comes from (A.6). Based on inequalities (A.7) and (A.8), we conclude

$$\begin{aligned}
&P((\mathbf{t}^I, \mathbf{t}^R) \in M_1) \\
&= 1 - P((\mathbf{t}^I, \mathbf{t}^R) \in M_1^c) \\
&= 1 - P(\mathcal{M}_1) - P(\mathcal{M}_2) \\
&> 1 - m(p_s)^{C_1} - S(S-1)(p_s)^{C_1} \\
&\overset{(a)}{>} 1 - S(S-1)C_1(p_s)^{C_1} - S(S-1)(p_s)^{C_1} \\
&= 1 - S(S-1)(C_1+1)(p_s)^{C_1}.
\end{aligned} \tag{A.9}$$

where (a) holds because of $m \leq S(S-1)C_1$. Then we can choose a constant C_1 such that for any $\epsilon_1 > 0$,

$$P((\mathbf{t}^I, \mathbf{t}^R) \in M_1) > 1 - \epsilon_1. \tag{A.10}$$

Now, we need to discuss $P(\mathcal{A}_{\alpha_1}...\mathcal{A}_{\alpha_m}\mathcal{A}_{\beta_1}...\mathcal{A}_{\beta_n}|\mathbf{t}^I, \mathbf{t}^R)$ when $(\mathbf{t}^I, \mathbf{t}^R) \in M_1$. In this case, we have

$$\begin{aligned}
&P(\mathcal{A}_{\alpha_1}...\mathcal{A}_{\alpha_m}\mathcal{A}_{\beta_1}...\mathcal{A}_{\beta_n}|\mathbf{t}^I, \mathbf{t}^R) \\
&= \prod_{\alpha \in \mathcal{V}_\alpha} P(\mathcal{A}_\alpha | t_\alpha^I, t_\alpha^R) \prod_{\beta \in V_\beta} P(\mathcal{A}_\beta | t_\beta^I, t_\beta^R).
\end{aligned} \tag{A.11}$$

According to the definition of $\mathcal{V}_\beta$, we have for any $\beta \in V_\beta$ and $1 \leq i \leq S$, $d(\beta, \zeta_i) > C_1$, which implies that for any $(\mathbf{t}^I, \mathbf{t}^R) \in M_1$ and any $\beta \in \mathcal{V}_\beta$, we have $t_\beta^I = \infty$. So $P(\mathcal{A}_\beta | t_\beta^I, t_\beta^R) = 1$.

For $P(\mathcal{A}_\alpha | t_\alpha^I, t_\alpha^R)$ when $\alpha \in \mathcal{V}_\alpha$, we define $u_1, ..., u_k$ to be the child nodes of root α on the tree T_α when $\alpha \in \mathcal{V}_\alpha$ and $\mathcal{A}_{u_i}$ $(i = 1, ..., k)$ to be the event that on the tree $T_{u_i}^{-\alpha}$ ($T_{u_i}^{-\alpha}$ is a subtree of T_α rooted at u_i but without the branch from α), there are at least $(S+1)$ one-time-slot branching processes survived after time t_0 or all infection processes die out before or at time t_0. On a $(g+1)$-regular tree, we have $k \leqslant g$. Then we have

$$\begin{aligned}
P(\mathcal{A}_\alpha | t_\alpha^I, t_\alpha^R) &> P(\mathcal{A}_{u_1}...\mathcal{A}_{u_k} | t_\alpha^I, t_\alpha^R) \\
&= \sum_{\mathbf{t}_u^I} P(\mathcal{A}_{u_1}...\mathcal{A}_{u_k} | \mathbf{t}_u^I, t_\alpha^I, t_a^R) P(\mathbf{t}_u^I | t_\alpha^I, t_\alpha^R) \\
&> \sum_{\mathbf{t}_u^I \in M_2} P(\mathcal{A}_{u_1}...\mathcal{A}_{u_k} | \mathbf{t}_u^I) P(\mathbf{t}_u^I | t_\alpha^I, t_\alpha^R),
\end{aligned} \tag{A.12}$$

where $\mathbf{t}_u^I = \{t_{u_1}^I, ..., t_{u_k}^I\}$ and $M_2 = \{\vec{t_u^I} |$ for $i = 1, ..., k$, $t_{u_i}^I - t_\alpha^I \leq C_2$, or $t_{u_i}^I = \infty\}$, and $C_2 \in \mathbb{N}$ is a constant.

To compute $P(\mathbf{t}_u^I \in M_2 | t_\alpha^I, t_\alpha^R)$, we consider the following three cases:

1. When $t_\alpha^I = \infty$, we have $t_{u_i}^I = \infty$ for $i = 1, ..., k$, which means $P(\mathbf{t}_u^I \in M_2 | t_\alpha^I, t_\alpha^R) = 1$.

2. When $t_\alpha^R - t_\alpha^I \le C_2$ and $t_\alpha^I \neq \infty$, we always have $t_{u_i}^I - t_\alpha^I \le C_2$ or $t_{u_i}^I = \infty$ for $i = 1, .., k$, which means $\mathbf{t}_u^I \in M_2$. Thus, we have $P(\mathbf{t}_u^I \in M_2 | t_\alpha^I, t_\alpha^R) = 1$.

3. When $t_\alpha^R - t_\alpha^I > C_2$ and $t_\alpha^I \neq \infty$, we have

$$\begin{aligned}
&P(\mathbf{t}_u^I \in M_2 | t_\alpha^I, t_\alpha^R) \\
&= \prod_{i=1}^{k} P(t_{u_i}^I - t_\alpha^I \le C_2 \text{ or } t_{u_i}^I = \infty | t_\alpha^I, t_\alpha^R) \\
&= \prod_{i=1}^{k} (\sum_{t_{u_i}^I = 1 + t_\alpha^I}^{C_2 + t_\alpha^I} q(1-q)^{t_{u_i}^I - t_\alpha^I - 1} + (1-q)^{t_\alpha^R - t_\alpha^I}) \\
&= \prod_{i=1}^{k} (1 - (1-q)^{C_2} + (1-q)^{t_\alpha^R - t_\alpha^I}) \\
&> \prod_{i=1}^{k} (1 - (1-q)^{C_2}) \\
&= (1 - (1-q)^{C_2})^k.
\end{aligned} \tag{A.13}$$

In summary, we always have

$$P(\mathbf{t}_u^I \in M_2 | t_\alpha^I, t_\alpha^R) > (1 - (1-q)^{C_2})^k. \tag{A.14}$$

We also have

$$P(\mathcal{A}_{u_1} ... \mathcal{A}_{u_k} | \mathbf{t}_u^I, t_\alpha^I, t_\alpha^R) = \prod_{i=1}^{k} P(\mathcal{A}_{u_i} | t_{u_i}^I) \overset{(a)}{>} (1 - \epsilon_2)^k, \tag{A.15}$$

where (a) can be proved by following the proof of Lemma 6 in Zhu and Ying (2013).

By substituting (A.14) and (A.15) into (A.12), we have

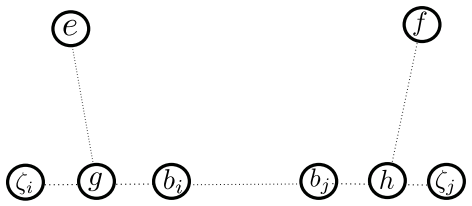
$$\begin{aligned}
&P(\mathcal{A}_\alpha | t_\alpha^I, t_\alpha^R) \\
&> \sum_{\mathbf{t}_u^I \in M_2} P(\mathcal{A}_{u_1} ... \mathcal{A}_{u_k} | \mathbf{t}_u^I, t_\alpha^I, t_\alpha^R) P(\mathbf{t}_u^I | t_\alpha^I, t_\alpha^R) \\
&> (1 - \epsilon_2)^k (1 - (1-q)^{C_2})^k.
\end{aligned} \tag{A.16}$$

Since $k \le g$, we can choose a constant C_2 such that for any $\epsilon_3 > 0$, $(1 - (1-q)^{C_2})^k > 1 - \epsilon_3$. Then we have

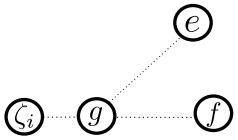
$$P(\mathcal{A}_\alpha | t_\alpha^I, t_\alpha^R) > (1 - \epsilon_2)^k (1 - \epsilon_3),$$

and

$$\begin{aligned}
P(\mathcal{A}_{\alpha_1} ... \mathcal{A}_{\alpha_m} \mathcal{A}_{\beta_1} ... \mathcal{A}_{\beta_n} | \mathbf{t}^I, \mathbf{t}^R) &> \prod_{\alpha \in \mathcal{V}_\alpha} (1 - \epsilon_2)^k (1 - \epsilon_3) \\
&= ((1 - \epsilon_2)^k (1 - \epsilon_3))^m.
\end{aligned} \tag{A.17}$$



(a) e and f are infected by two sources.



(b) e and f are infected by the same source.

Figure A.4: Different cases of distances between two infected nodes. In Figure A.4a, we consider the infection process of two sources, ζ_i and ζ_j. Assume b_i and b_j are the nodes on path (ζ_i, ζ_j) that satisfy $d(b_i, \zeta_i) = C_1$ and $d(b_j, \zeta_j) = C_1$. Under event $\tilde{\mathcal{A}}$, on path (ζ_i, ζ_j) only nodes on (ζ_i, b_i) and (ζ_j, b_j) can be infected. If e and f are endpoints of survived one-time-slot branching processes, we have $d(\zeta_i, e) \geq t - t_0$ and $d(\zeta_j, f) \geq t - t_0$. Therefore, $d(e, f) \geq d(\zeta_i, \zeta_j) - d(\zeta_i, b_i) - d(\zeta_j, b_j) + d(\zeta_i, e) - d(\zeta_i, b_i) + d(\zeta_j, f) - d(\zeta_j, b_j)$, which means $d(e, f) \geq d(\zeta_i, \zeta_j) + 2(t - t_0) - 4C_1$. In Figure A.4b, we consider the situation that e and f are infected by the same source, ζ_i. If e and f are endpoints of survived one-time-slot branching process, we have $d(e, f) = d(\zeta_i, e) + d(\zeta_i, f) - 2d(\zeta_i, g)$. Since the survived one-time-slot branching processes do not overlap after t_0, we have $d(\zeta_i, g) \leq t_0$. Therefore, we have $d(e, f) \geq 2(t - t_0) - 2t_0 = 2t - 4t_0$

Substituting (A.17) and (A.10) into (A.1), we have

$$\begin{aligned} P(\mathcal{A}) &> P(\tilde{\mathcal{A}}) \\ &= \sum_{\mathbf{t}^I, \mathbf{t}^R \in M_1} P(\mathcal{A}_{\alpha_1} ... \mathcal{A}_{\alpha_m} \mathcal{A}_{\beta_1} ... \mathcal{A}_{\beta_n} | \mathbf{t}^I, \mathbf{t}^R) P(\mathbf{t}^I, \mathbf{t}^R) \\ &> ((1 - \epsilon_2)^k (1 - \epsilon_3))^m (1 - \epsilon_1) \\ &> 1 - \epsilon. \end{aligned} \tag{A.18}$$

□

Assuming that the time when the snapshot is taken, t, satisfies $t \gg t_0$ and $t \gg d_{i,j}$ for any $i, j = 1, ..., S$, where $d_{i,j}$ is defined to be $d(\zeta_i, \zeta_j)$, and $d > 3St_0 + 4SC_1$, where $d = \min_{1 \leq i,j \leq S} d_{i,j}$. We need to prove that the distance between each estimator and its closest original source is bounded by some constants under event $\tilde{\mathcal{A}}$. We say a source ζ_i has infected nodes if there are some infected nodes associated with ζ_i when the snapshot was taken. Since in the following proof, we will use the distance between two infected nodes in the graph frequently, here in Figure A.4, we discuss some cases of distances between two infected nodes commonly used later.

We divide event $\tilde{\mathcal{A}}$ into three subevents:

1. Infection processes from all original sources die out at time t_0.

   There are two cases under this condition:

- The number of infected nodes is less than or equal to S. All those infected nodes will be treated as our estimators according to Algorithm 2.1. Because there is no node getting infected after time t_0, each of these nodes is within the distance of t_0 from the source that infects it. Therefore, the distance between each estimator and its closest source is bounded by t_0, which is a constant.
- The number of infected nodes is greater than S.

  Claim A.1. *After step 2, the set $\mathcal{B}$ contains infected nodes associated with all sources who have infected nodes when the snapshot is taken.*

  Proof of Claim A.1: Assume set $\mathcal{B}$ does not contain any infected node associated with some source ζ_k that has infected nodes observed in the snapshot, which means $\mathcal{B}$ contains at least two nodes associated with same source. Assume $a, b \in \mathcal{B}$ are associated with source ζ_j and $c \notin \mathcal{B}$ to be an infected node associated with ζ_k.
  Then we have $d(a, b) \leq 2t_0$. For any $e \in \mathcal{B}$, we have

$$\begin{aligned} d(e, c) &\geq d - 2C_1 \\ &> 3St_0 + 4SC_1 - 2C_1 \\ &> 2t_0 \\ &> d(a, b), \end{aligned} \tag{A.19}$$

  which contradicts the step 2 in Algorithm 2.1.

  Claim A.2. *After step 3, in each set $\mathcal{V}_I^{(i)}$ $(i = 1, ..., S)$, the nodes are infected by the same source.*

  Proof of Claim A.2: Without loss of generality, we assume $\mathcal{B} = \{e_1, ..., e_S\}$ and $e_i \in \mathcal{V}_I^{(i)}$. Assume node a in set $\mathcal{V}_I^{(i)}$ is associated with a source which is different from the source that e_i is associated with. Then we have

$$\begin{aligned} d(a, e_i) &\overset{(a)}{\geq} d - 2C_1 \\ &> 3St_0 + (4S - 2)C_1, \end{aligned} \tag{A.20}$$

  where (a) is true because of the definition of event $\tilde{\mathcal{A}}$ and set M_1. According to Claim A.1, set $\mathcal{B}$ contains nodes associated with the sources that have infected nodes associated with them in the snapshot. Therefore, we can assume in another set $\mathcal{V}_I^{(j)}$, node e_j is associated with the same source as node a. Then we have $d(e_j, a) < 2t_0$. Therefore, we have $d(e_j, a) < d(e_i, a)$, which is in contradiction with step 3. Thus, Claim A.2 holds.

  According to Claim A.2, in each set $\mathcal{V}_I^{(i)}$, all the nodes are infected by the same source. Therefore, the maximum infection radius $r_{\max}$ after step 4 should satisfy that $r_{\max} \leq t_0$. Assume γ_i to be the estimator generated by $\mathcal{V}_I^{(i)}$ that contains e_i and ζ_i to be the actual source who infected the nodes in $\mathcal{V}_I^{(i)}$. Then we have $d(\gamma_i, e_i) = r_{max}$ and $d(e_i, \zeta_i) \leq t_0$. Therefore,

$d(\gamma_i, \zeta_i) \leq 2t_0$, which means the distance between each estimator and its closest source is bounded by $2t_0$, which is a constant.

2. There are at least two sources surviving $(S+1)$ one-time-slot branching processes after time t_0.

   Assume the number of sources who survive at least $(S+1)$ one-time-slot branching processes is n_0. And we have $n_0 \geq 2$. We will then prove the following conclusions:

   (a) $e_1,\ e_2, ...,\ e_{n_0}$ are infected by different sources that survive at least $(S+1)$ one-time-slot branching processes after t_0.

   (b) For any $i \geq 2$,
   $$d(e_i, \hat{\zeta}_i) \geq t - it_0 - 4(i-1)C_1, \tag{A.21}$$
   and when $i = 1$,
   $$d(e_1, \hat{\zeta}_1) \geq t - 2t_0 - 4C_1, \tag{A.22}$$
   where $\hat{\zeta}_i$ represents the source that e_i is associated with.

   (c) For any $1 \leq i, j \leq n_0$, we have
   $$d(e_i, e_j) > 2t. \tag{A.23}$$

   Proof of the conclusions:

   - $e_1, e_2 (i = 1, 2)$: Assume η_1 and η_2 are two leaf-nodes of one-time-slot processes associated with two different sources. Since for any $a, b \in \mathcal{V}_I$, $d(e_1, e_2) \geq d(a, b)$, we have
   $$d(e_1, e_2) \geq d(\eta_1, \eta_2) \overset{\text{(a)}}{\geq} d + 2(t - t_0) - 4C_1 > 2t,$$
   where (a) is true because of Figure A.4a. Thus, e_1 and e_2 have to be infected by two different sources. Without loss of generality, we assume ζ_1 and ζ_2 are the two sources who infect e_1 and e_2. Then we have
   $$\begin{aligned} d(e_1, \zeta_1) + d_{1,2} + d(e_2, \zeta_2) &\geq d(e_1, e_2) \\ &\geq d_{1,2} + 2(t - t_0) - 4C_1, \end{aligned} \tag{A.24}$$
   where $d_{i,j} = d(\zeta_i, \zeta_j)$ for any $i, j = 1, ..., S$ and because $d(e_1, \zeta_1) \leq t$ and $d(e_2, \zeta_2) \leq t$, we have $d(e_1, \zeta_1) \geq t - 2t_0 - 4C_1$ and $d(e_2, \zeta_2) \geq t - 2t_0 - 4C_1$. Because $t \gg t_0$ and $t \gg C_1$, we may consider $d(e_1, \zeta_1) > t_0$ and $d(e_2, \zeta_2) > t_0$, which implies ζ_1 and ζ_2 must be two surviving sources. Therefore, e_1 and e_2 satisfy these three conclusions.

   - $e_k (2 \leq k \leq n_0 - 1)$: Assume that e_k satisfies the three conclusions, which means that e_k is infected by another source ζ_k, who survives at least $(S+1)$ one-time-slot branching processes at time t_0, $d(e_i, \zeta_k) \geq t - kt_0 - 4(k-1)C_1$, and for any $1 \leq i, j \leq k$, $d(e_i, e_j) > 2t$.

- e_{k+1}: If e_{k+1} is infected by $\zeta_1, ..., \zeta_k$ or any source that dies out at time t_0, we have $\min_{i=1}^{k} d(e_i, e_{k+1}) \leq 2t$. If e_{k+1} is infected by another source ζ_{k+1}, who survives at least $(S+1)$ one-time-slot branching processes at time t_0, assume η_{k+1} is the leaf-node of one survived one-time-slot branching process infected by ζ_{k+1} and we have

$$\begin{aligned} d(\eta_{k+1}, e_1) &\geq (t - 2t_0 - 4C_1) + d_{1,k+1} + t - t_0 - 4C_1 \\ &\overset{(a)}{>} 2t, \end{aligned} \tag{A.25}$$

where (a) holds according to $d_{1,k+1} \geq d > 3St_0 + 4SC_1$, and

$$\begin{aligned} d(\eta_{k+1}, e_i) \geq\ & (t - it_0 - 4(i-1)C_1) \\ & + d_{i,k+1} + t - t_0 - 4C_1 \\ >\ & 2t, \end{aligned} \tag{A.26}$$

for $i = 2, ..., k$. Thus, we have

$$\begin{aligned} d(e_{k+1}, e_i) &\geq \min_{i=1}^{k} d(\eta_k, e_i) \\ &> 2t. \end{aligned} \tag{A.27}$$

Therefore, we know that when e_{k+1} is infected by ζ_{k+1}, we have

$$\min_{i=1}^{k} d(e_{k+1}, e_i) > 2t.$$

Then, e_{k+1} has to be infected by another source who survives at least $(S+1)$ one-time-slot branching processes at time t_0 according to Algorithm 2.1. Assuming $j = \arg \min_{i \in \{1,...,k\}} d(\eta_{k+1}, e_i)$, we have

$$\begin{aligned} & d(e_{k+1}, \zeta_{k+1}) + d_{j,k+1} + d(e_j, \zeta_j) \\ \geq\ & d(e_{k+1}, e_j) \\ \geq\ & d(\eta_{k+1}, e_j) \\ \geq\ & d(e_j, \zeta_j) + d_{j,k+1} + t - t_0 - 4C_1 \\ \geq\ & t - kt_0 - 4(k-1)C_1 + d_{j,k+1} - 4C_1 + t - t_0 \\ =\ & 2t - (k+1)t_0 - 4kC_1 + d_{j,k+1}. \end{aligned} \tag{A.28}$$

Because $d(e_j, \zeta_j) \leq t$, we have $d(e_{k+1}, \zeta_{k+1}) \geq t - (k+1)_0 - 4kC_1$. Therefore, e_{k+1} satisfies the three conclusions mentioned before.

Then we complete the proof the conclusions.

Since $e_1, \ldots, e_{n_0}$ are infected by different sources who survive at least $(S+1)$ one-time-slot branching processes, we can assume $e_i (i = 1, ..., n_0)$ is infected by ζ_i, while ζ_i survives at least $(S+1)$ one-time-slot branching processes after time t_0. According to inequalities (A.21), (A.22) and (A.23), we have

$$d(e_i, \zeta_i) \geq t - n_0 t_0 - 4(n_0 - 1)C_1 \tag{A.29}$$

and for any $1 \leq i, j \leq n_0$, $d(e_i, e_j) > 2t$.

Define $\hat{\zeta}_i$ to be the source whom e_i $(i = n_0 + 1, ..., S)$ is associated with. Then we will prove the following conclusion:

(a) For $e_i (i = n_0 + 1, ..., S)$, we have

$$d(e_i, \hat{\zeta}_i) \geq t - (3i - 2n_0)t_0 - 4(n_0 - 1)C_1, \tag{A.30}$$

where $\hat{\zeta}_i$ is one of $\zeta_1, ..., \zeta_{n_0}$.

(b) For $j = 1, ..., i-1$, we have

$$d(e_i, e_j) \geq 2t - (3i - 2n_0)t_0 - 4(n_0 - 1)C_1. \tag{A.31}$$

Proof of the conclusions:

- e_{n_0+1}: If e_{n_0+1} is infected by a source ζ_{n_0+1}, whose infection process dies out at time t_0, for any $i = 1, ..., n_0$, we have $d(e_{n_0+1}, e_i) \leq d_{n_0+1,i} + t + t_0$, which means

$$\min_{i \in \{1,\ldots,n_0\}} d(e_{n_0+1}, e_i) \leq d' + t + t_0, \tag{A.32}$$

where $d' = \min_{i \in \{1,\ldots,n_0\}} d_{n_0+1,i}$.

If e_{n_0+1} is infected by ζ_j, where $j = 1, 2, ..., n_0$, we have

$$\begin{aligned} & d(e_{n_0+1}, e_i) \\ \geq\ & d(e_i, \zeta_i) + d_{i,j} - 4C_1 + d(e_{n_0+1}, \zeta_j) \\ \geq\ & t - n_0 t_0 + 4(n_0 - 1)C_1 + d_{i,j} - 4C_1 + d(e_{n_0+1}, \zeta_j) \\ =\ & t - n_0 t_0 - 4n_0 C_1 + d_{i,j} + d(e_{n_0+1}, \zeta_j) \\ >\ & t + d(e_{n_0+1}, \zeta_j), \end{aligned} \tag{A.33}$$

where $i \neq j$, $i = 1, ..., n_0$, and

$$d(e_{n_0+1}, e_j) \leq t + d(e_{n_0+1}, \zeta_j). \tag{A.34}$$

Therefore, assuming η_{n_0+1} is the leaf-node of a survived one-time-slot branching process generated by ζ_j, we have

$$\begin{aligned} \min_{i \in \{1,\ldots,n_0\}} d(e_{n_0+1}, e_i) &= d(e_{n_0+1}, e_j) \\ &\geq d(\eta_{n_0+1}, e_j). \end{aligned} \tag{A.35}$$

Since every surviving source has at least $(S+1)$ one-time-slot branching processes survived after time t_0, we can always find a leaf-node ζ_{n_0+1}, whose one-time-slot branching process doesn't overlap with path (e_{n_0+1}, ζ_j). Therefore, we have

$$\begin{aligned} d(e_{n_0+1}, e_j) &\geq d(\eta_{n_0+1}, e_j) \\ &\geq d(e_j, \zeta_j) + d(\eta_{n_0+1}, \zeta_j) - 2t_0 \\ &\geq t - n_0 t_0 - 4(n_0-1)C_1 + t - t_0 - 2t_0 \\ &= 2t - (n_0+3)t_0 - 4(n_0-1), \end{aligned} \tag{A.36}$$

which is bigger than $d' + t + t_0$ of (A.32) according to $t \gg t_0$ and $t \gg d_{i,j}$ for any $i, j = 1, ..., S$. This means e_{n_0+1} has to be infected by a source that survives at least $(S+1)$ one-time-slot branching processes at time t_0. According to (A.33), (A.34) and (A.36), we have $d(e_{n_0+1}, e_i) \geq 2t - (n_0 + 3)t_0 - 4(n_0-1)$, where $i = 1, ..., n_0$. Because $d(e_{n_0+1}, \zeta_j) + d(e_j, \zeta_j) \geq d(e_{n_0+1}, e_j)$ and $d(e_j, \zeta_j) \leq t$, we have

$$d(e_{n_0+1}, \zeta_j) \geq t - (n_0+3)t_0 - 4(n_0-1)C_1.$$

Therefore, the conclusions holds for e_{n_0+1}.

- $e_k (n_0 + 1 \leq k < S)$: Assume that it's infected by one of $\zeta_i (i = 1, ..., n_0)$ and we have

$$d(e_k, \hat{\zeta}_k) \geq t - (3k - 2n_0)t_0 - 4(n_0-1)C_1 \tag{A.37}$$

and for $i = 1, ..., k-1$,

$$d(e_k, e_i) \geq 2t - (3k - 2n_0)t_0 - 4(n_0-1)C_1. \tag{A.38}$$

- e_{k+1}: If e_{k+1} is infected by a source ζ_{k+1}, whose infection process dies out at time t_0, for any $i = 1, ..., k$, we have $d(e_{k+1}, e_i) \leq d(\zeta_{k+1}, \hat{\zeta}_i) + t + t_0$, which means

$$\min_{i \in \{1,...,k\}} d(e_{k+1}, e_i) \leq d'' + t + t_0, \tag{A.39}$$

where $d'' = \min_{i=\{1,...,k\}} d(\zeta_{k+1}, \hat{\zeta}_i)$.

According to the assumption for e_k and inequality (A.29), we have

$$d(e_i, \hat{\zeta}_i) \geq t - (3k - 2n_0)t_0 - 4(n_0-1)C_1, \tag{A.40}$$

for $i \in \{1, ..., k\}$.

If e_{k+1} is associated with ζ_j, where $j \in \{1, ..., n_0\}$, for $\zeta_j \neq \hat{\zeta}_i$, we have

$$\begin{aligned} d(e_{k+1}, e_i) &\geq d(e_{k+1}, \zeta_j) + d(\hat{\zeta}_i, e_i) + d(\zeta_j, \hat{\zeta}_i) - 4C_1 \\ &\geq t - (3k - 2n_0)t_0 - 4(n_0-1)C_1 \\ &\quad + d(\zeta_j, \hat{\zeta}_i) + d(e_{k+1}, \zeta_j) - 4C_1 \\ &> t + d(e_{k+1}, \zeta_j). \end{aligned} \tag{A.41}$$

When $\zeta_j = \hat{\zeta}_i$, we have $d(e_{k+1}, e_i) \leq t + d(e_{k+1}, \zeta_j)$. Assume

$$e' = \underset{\{e_1,...,e_k\}}{\operatorname{argmin}} d(e_{k+1}, e_i),$$

where e' is one of $e_1, ..., e_k$ and is associated with ζ_j, and η_{k+1} is the leaf-node of a survived one-time-slot branching process infected by ζ_j. Similarly, we assume $e'' = \arg\min_{e_i} d(\eta_{k+1}, e_i)$, where e'' is one of $e_1, ..., e_k$ and is associated with ζ_j.
According to Algorithm 2.1, we have

$$\begin{aligned} \min_{i=1}^{k} d(e_{k+1}, e_i) &= d(e_{k+1}, e') \\ &\geq d(\eta_{k+1}, e''). \end{aligned} \tag{A.42}$$

Since every survived source has at least $(S+1)$ one-time-slot branching processes survived at time t_0, we can always find an leaf-node η_{k+1}, whose one-time-slot branching process doesn't overlap with (e_{k+1}, ζ_j). Therefore, we have

$$\begin{aligned} & d(e_{k+1}, e') \\ \geq\ & d(\eta_{k+1}, e'') \\ \geq\ & d(\eta_{k+1}, \zeta_j) + d(e'', \zeta_j) - 2t_0 \\ \geq\ & t - t_0 + t - (3k - 2n_0)t_0 - 4(n_0 - 1)C_1 - 2t_0 \\ =\ & 2t - (3k + 3 - 2n_0)t_0 - 4(n_0 - 1)C_1, \end{aligned} \tag{A.43}$$

which is bigger than $d'' + t + t_0$ of (A.39) because of $t \gg t_0$ and $t \gg d_{i,j}$ for any $i, j = 1, ..., k+1$. This means e_{k+1} has to be infected by a surviving source. Since for $i = 1, ..., k$, the inequalities

$$d(e_{k+1}, e_i) \geq 2t - (3k + 3 - 2n_0)t_0 - 4(n_0 - 1)C_1,$$

$$d(e_{k+1}, \zeta_j) + d(e', \zeta_j) \geq d(e_{k+1}, e'),$$

and $d(e', \zeta_j) \leq t$, hold, we have

$$d(e_{k+1}, \zeta_j) \geq t - (3k + 3 - 2n_0)n_0 t_0 - 4(n_0 - 1)C_1, \tag{A.44}$$

which satisfies the condition $d(e_i, \hat{\zeta}_i) \geq t - (3k + 3 - 2n_0)t_0 - 4(n_0 - 1)C_1$, when $i = k + 1$.

Then we complete the proof of the conclusions.

According to inequalities (A.29) and (A.40), we have that for $i = 1, ..., n_0$, $d(e_i, \hat{\zeta}_i) \geq t - it_0 - 4(i-1)C_1$ and for $i = n_0 + 1, ..., S$, $d(e_i, \hat{\zeta}_i) \geq t - (3i - 2n_0)t_0 - 4(n_0 - 1)C_1$ $(i = n_0 + 1, ..., S)$. Thus, when $n_0 \geq 2$, if we set $C_3 = (3S - 2n_0)t_0 + 4(n_0 - 1)C_1$, then we have $d(e_i, \hat{\zeta}_i) \geq t - C_3$. And for $1 \leq i, j \leq S$, we have

$$d(e_i, e_j) \geq 2t - (3S - 2n_0)t_0 - 4(n_0 - 1)C_1, \tag{A.45}$$

which means $d(e_i, e_j) \geq 2t - C_3$.

Claim A.3. *If $d > 3St_0+4SC_1$ and there are at least two sources having $(S+1)$ one-time-slot branching processes survived, the leaf-nodes of all survived one-time-slot branching processes in the same set $\mathcal{V}_I^{(i)}$ $(i = 1, ..., S)$ are associated with the same source that e_i is associated with.*

Proof of Claim A.3: Assume $e_i \in \mathcal{V}_I^{(i)}$ is associated with source ζ_i and $a \in \mathcal{V}_I^{(i)}$ is the leaf-node of an survived one-time-slot branching process generated by another source ζ_j, which means in set $\mathcal{V}_I^{(i)}$, e_i and a are associated with different sources.

Assume $e_j \in \mathcal{V}_I^{(i)}$ is associated with source ζ_j. Then we have

$$\begin{aligned} d(a, e_i) &\geq d(e_i, \zeta_i) + d(a, \zeta_j) + d_{i,j} - 4C_1 \\ &\geq (t - C_3) + (t - t_0) + d_{i,j} - 4C_1. \\ &= 2t + d_{i,j} - C_3 - t_0 - 4C_1 \end{aligned} \tag{A.46}$$

Because $C_3 = (3S - 2n_0)t_0 + 4(n_0 - 1)C_1$ and $d_{i,j} > 3St_0 + 4SC_1$, we have

$$\begin{aligned} & d_{i,j} - C_3 - t_0 - 4C_1 \\ &\geq (2n_0 - 1)t_0 + 4(S - n_0)C_1 \\ &> 0 \end{aligned} \tag{A.47}$$

which means $d(a, e_i) > 2t$. However, we have $d(a, e_j) \leq 2t$, which means $d(a, e_i) > d(a, e_j)$, which is in contradiction to step 3. This completes the proof Claim A.3.

Next, we need to prove that the distance between each estimator and its closest source is bounded by a constant. Since there are S clusters and each surviving source has at least $(S + 1)$ one-time-slot branching processes, there is at least one cluster containing two leaf-nodes of one-time-slot branching processes from one source. Then in step 4, the infection radius r_i of set $\mathcal{V}_I^{(i)}$ that contains at least two leaf-nodes of survived one-time-slot branching processes from a single source satisfies that $t - t_0 \leqslant r_i \leqslant t$. Then $r_{\max} = \max\{r_i\}$ in step 4 should satisfy that $t - t_0 \leqslant r_{\max} \leqslant t$.

Then, we need to prove that, in step 5, on the tree $\mathcal{T}$, the node, which is in distance $r_{\max}$ from e_i, is in a constant distance to the surviving original source with which e_i is associated. Define node h_i to be the node on path $(\hat{\zeta}_i, e_i)$ that satisfies $d(\hat{\zeta}_i, h_i) = C_3$. Then we the following claim.

Claim A.4. *Node h_i is on the tree $\mathcal{T}$ for all $i = 1, ..., S$.*

Proof of Claim A.4: There are two cases:

(a) $e_i, e_j \in \mathcal{B}$ are associated with the same source:
Without loss of generality, we assume e_i and e_j are associated with ζ_i. Then we know that $K_{e_i,e_j}^{\zeta_i}$ (the definition is in Table 2.1) is on the path (e_i, e_j) and we have

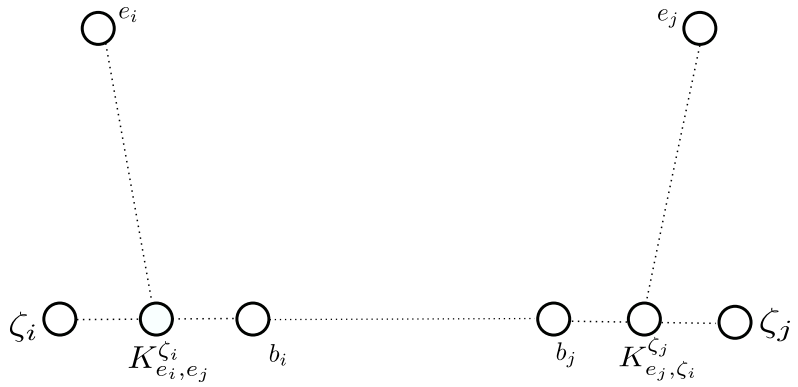


Figure A.5: The positions of ζ_i, ζ_j, e_i and e_j when e_i and e_j are associated with different sources, where $d(b_i, \zeta_i) = C_1$ and $d(b_j, \zeta_j) = C_1$.

$$d(e_i, e_j) = d(e_i, \zeta_i) + d(e_j, \zeta_i) - 2d(\zeta_i, K^{\zeta_i}_{e_i,e_j}) \tag{A.48}$$

and

$$d(e_i, e_j) \geq 2t - C_3, \tag{A.49}$$

which means

$$\begin{aligned} 2d(\zeta_i, K^{\zeta_i}_{e_i,e_j}) &\leq d(e_i, \zeta_i) + d(e_j, \zeta_i) - 2t + C_3 \\ &\leq C_3, \end{aligned} \tag{A.50}$$

Therefore, we have

$$d(\zeta_i, K^{\zeta_i}_{e_i,e_j}) \leq C_3/2. \tag{A.51}$$

(b) $e_i, e_j \in \mathcal{B}$ are associated with different sources:
Without loss of generality, we assume e_i is associated with ζ_i, while e_j is associated with ζ_j. Figure A.5 is the description of the relations among nodes e_i, e_j, ζ_i and ζ_j. According to the definition, both $K^{\zeta_i}_{e_i,e_j}$ and $(K^{\zeta_j}_{e_j,\zeta_i}, e_j)$ are on the path (ζ_i, e_j). More precisely, $K^{\zeta_i}_{e_i,e_j}$ is on the path $(\zeta_i, K^{\zeta_j}_{e_j,\zeta_i})$. Since path $(\zeta_i, K^{\zeta_j}_{e_j,\zeta_i})$ is part of path (ζ_i, ζ_j), which means $K^{\zeta_i}_{e_i,e_j}$ is on path (ζ_i, ζ_j). Because $K^{\zeta_i}_{e_i,e_j}$ is associated with ζ_i, under event $\tilde{\mathcal{A}}$, we have $d(\zeta_i, K^{\zeta_i}_{e_i,e_j}) \leq C_1$

Therefore, for all $e_i, e_j \in \mathcal{B}$, we have

$$\begin{aligned} d(\hat{\zeta}_i, K^{\hat{\zeta}_i}_{e_i,e_j}) &\leq \max\{C_3/2, C_1\} \\ &= C_3/2. \end{aligned} \tag{A.52}$$

Then for each e_i, if we choose h_i which satisfies $h_i \in (\hat{\zeta}_i, e_i)$ and $d(h_i, \hat{\zeta}_i) = C_3$. We have $h_i \in (K^{\hat{\zeta}_i}_{e_i,e_j}, e_i)$. Since path $(K^{\hat{\zeta}_i}_{e_i,e_j}, e_i)$ is on the tree $\mathcal{T}$, h_i is on the tree $\mathcal{T}$. This completes the proof of Claim A.4.

Therefore, we have $d(e_i, \hat{\zeta}_i) = d(e_i, h_i) + d(h_i, \hat{\zeta}_i)$ and $d(\hat{\zeta}_i, h_i) = C_3$. Assume γ_i to be the estimator to $\hat{\zeta}_i$ generated by e_i. Because $r_{\max} \geq t - t_0$, $(e_i, h_i) \subset (e_i, e_j)$ for all $e_j \in \mathcal{B}$ and

$$\begin{aligned} d(e_i, h_i) &= d(e_i, \hat{\zeta}_i) - d(h_i, \hat{\zeta}_i) \\ &\leq t - C_3 \\ &< t - t_0 \\ &\leq r_{\max}, \end{aligned} \tag{A.53}$$

we have $r_{\max} = d(e_i, \gamma_i) = d(e_i, h_i) + d(h_i, \gamma_i)$.

Therefore,

$$\begin{aligned} d(\gamma_i, \hat{\zeta}_i) &\leqslant d(\gamma_i, h_i) + d(\hat{\zeta}_i, h_i) \\ &= d(e_i, \gamma_i) - d(e_i, h_i) + d(\hat{\zeta}_i, h_i) \\ &= d(e_i, \gamma_i) - (d(e_i, \hat{\zeta}_i) - d(h_i, \hat{\zeta}_i)) + d(\hat{\zeta}_i, h_i) \\ &\leq t - (t - C_3 - C_3) + C_3 \\ &\leq 3C_3. \end{aligned} \tag{A.54}$$

Finally, we have $d(\gamma_i, \hat{\zeta}_i) \leqslant 3C_3$, which means the estimator we find is in a constant distance with the surviving original source.

3. There is only one source surviving $(S+1)$ one-time-slot branching processes after time t_0.

   Assuming $\hat{\zeta}_i$ to be the source that infects e_i, we will prove the following conclusions:

   (a) For $i = 1, ..., S$, $\hat{\zeta}_i$ must be the same source that survives at least $(S+1)$ one-time-slot branching processes at time t_0.

   (b) For e_1, we have
   $$d(e_1, \hat{\zeta}_1) \geq t - 4t_0 \tag{A.55}$$

   (c) For $e_i (i = 2, ..., S)$, we have
   $$d(e_i, \hat{\zeta}_i) \geq t - (3i - 2)t_0. \tag{A.56}$$

   (d) For any $i = 1, ..., S$ and $j = 1, ..., i-1$, we have
   $$d(e_i, e_j) \geq 2t - (3i - 2)t_0. \tag{A.57}$$

   Proof of the conclusions: At first we assume that the only source who survives at least $(S+1)$ one-time-slot branching processes at time t_0 is ζ_j.

   - e_1, e_2: If e_1 and e_2 are associated with two other sources, let's say ζ_{i_1} and ζ_{i_2}, except ζ_j, we have $d(e_1, e_2) \leq 2t_0 + d_{i_1, i_2}$.
     If e_1 and e_2 are associated with the same source except ζ_j, we have $d(e_1, e_2) \leq 2t_0$.

If e_1 and e_2 are associated with ζ_j, assuming η_1 and η_2 are two leaf-nodes of survived one-time-slot branching processes of ζ_j, we have

$$\begin{aligned} d(e_1, e_2) &\geq d(\eta_1, \eta_2) \\ &\geq d(\eta_1, \zeta_j) + d(\eta_2, \zeta_j) - 2t_0 \\ &\geq t - t_0 + t - t_0 - 2t_0 \\ &= 2t - 4t_0. \end{aligned} \tag{A.58}$$

Because $t \gg t_0$, we know that when e_1 and e_2 are associated with the same source ζ_j, we can get $d(e_1, e_2)$ maximized. According to Algorithm 2.1, e_1 and e_2 are associated with ζ_j. Since $d(e_2, \zeta_j) \leq t$ and

$$\begin{aligned} d(e_1, \zeta_j) + d(e_2, \zeta_j) &\geq d(e_1, e_2) \\ &\geq 2t - 4t_0, \end{aligned} \tag{A.59}$$

we have $d(e_1, \zeta_j) \geq t - 4t_0$. In a similar way, we have $d(e_2, \zeta_j) \geq t - 4t_0$. Therefore, these conclusions hold for e_1 and e_2.

- $e_k (2 < k \leq S-1)$: Assume that e_k is infected by ζ_j and it satisfies $d(e_k, \zeta_j) \geq t-(3k-2)t_0$ and $d(e_k, e_i) \geq 2t-(3k-2)t_0$, where $i = 1, \ldots, k-1$.
- $e_{k+1} (2 \leq k \leq S-1)$: If e_{k+1} is associated with another source $\zeta_{i_{k+1}}$, we have $d(e_{k+1}, e_i) \leq d(\zeta_{i_{k+1}}, \zeta_j) + t + t_0$.
  If e_{k+1} is associated with ζ_j, assuming η_{k+1} is the leaf-node of a survived one-time-slot branching process of ζ_j who dose not overlap with (ζ_j, e_i) for $i = 1, ..., k$ after time t_0, we have

$$\begin{aligned} d(e_i, \eta_{k+1}) &\geq d(e_i, \zeta_j) + d(\eta_{k+1}, \zeta_j) - 2t_0 \\ &\geq t - (3i-2)t_0 + t - t_0 - 2t_0 \\ &\geq t - (3k-2)t_0 + t - t_0 - 2t_0 \\ &= 2t - (3k+1)t_0, \end{aligned} \tag{A.60}$$

  which means for any $i = 1, ..., k$, we have

$$\begin{aligned} d(e_i, e_{k+1}) &\geq \min_{i=1}^{k} d(e_i, \eta_{k+1}) \\ &\geq 2t - (3k+1)t_0. \end{aligned} \tag{A.61}$$

  Since we hypothesize $t >> t_0$ and $t >> d_{i,j}$ for any $i, j = 1, ..., k+1$, we have $2t - (3k+1)t_0 > d(\zeta_{i_{S+1}}, \zeta_j) + t + t_0$, which means e_{k+1} has to be associated with ζ_j.
  Assuming

$$e' = \arg \min_{e_i, i=1,\ldots k} d(e_i, e_{S+1}),$$

  because $d(e', \zeta_j) \leq t$ and

$$\begin{aligned} d(e', \zeta_j) + d(e_{k+1}, \zeta_j) &\geq d(e', e_{k+1}) \\ &\geq 2t - (3k+1)t_0, \end{aligned} \tag{A.62}$$

  we have $d(e_{k+1}, \zeta_j) \geq t - (3k+1)t_0$.

The we have finished the proof of these conclusions.

Therefore, according to inequalities (A.55), (A.56) and (A.57), for any $i = 1, ..., S$, we have

$$d(e_i, \hat{\zeta}_i) \geq t - C_4, \tag{A.63}$$

$$d(e_i, e_j) \geq 2t - C_4, \forall 1 \leq i, j \leq S, \tag{A.64}$$

where we define $C_4 = (3S - 2)t_0$, and $e_1, ..., e_S$ are infected by the only source who survives at least $(S + 1)$ one-time-slot branching processes at time t_0.

Because there are S clusters and each surviving source has at least $(S + 1)$ one-time-slot branching processes, we have at least one cluster containing two leaf-nodes of one-time-slot branching processes. Then in step 4, the infection radius r_i of the set $\mathcal{V}_I^{(i)}$, which contains at least two leaf-nodes of survived one-time-slot branching processes from a single source satisfies that $t - t_0 \leqslant r_i \leqslant t$. Then $r_{\max} = \max\{r_i\}$ in step 4 should also satisfy that $t - t_0 \leqslant r_{\max} \leqslant t$.

Next, we need to consider the tree $\mathcal{T}$, which is formed by nodes in $\mathcal{B}$ and paths between every two nodes in $\mathcal{B}$. According to the conclusions we have proved before, the nodes in $\mathcal{B}$ are associated with the same source. Without loss of generality, we assume that nodes in set $\mathcal{B}$ are infected by the source ζ_1. Then for any two nodes e_i, e_j in $\mathcal{B}$ we have

$$\begin{aligned} & d(e_i, e_j) \\ &= d(e_i, \zeta_1) + d(e_j, \zeta_1) - 2d(\zeta_1, K_{e_i,e_j}^{\zeta_1}) \\ &\geq 2t - C_4, \end{aligned} \tag{A.65}$$

which means

$$\begin{aligned} & 2d(\zeta_1, K_{e_i,e_j}^{\zeta_1}) \\ &\leq d(e_i, \zeta_1) + d(e_j, \zeta_1) - 2t + C_4 \\ &\leq C_4. \end{aligned} \tag{A.66}$$

Therefore, we have $d(\zeta_1, K_{e_i,e_j}^{\zeta_1}) \leq C_4/2$.

According to the definition of $\mathcal{T}$, we know that path $(e_i, K_{e_i,e_j}^{\zeta_1})$ is on the tree $\mathcal{T}$. Therefore, for node e_i, if we define another node h_i, which is on the path (ζ_1, e_i) and satisfies $d(h_i, \zeta_1) = C_4$, h_i is on the tree $\mathcal{T}$. In a similar way, we can define h_j.

Thus, for each $e_i \in \mathcal{B}(i = 1, \ldots, S)$, we can define another node h_i which is on the path (ζ_1, e_i) and satisfies $d(\zeta_1, h_i) = C_4$ so that it is on the tree $\mathcal{T}$. And we have $d(e_i, \zeta_1) = d(e_i, h_i) + d(h_i, \zeta_1)$ and $d(\zeta_1, h_i) = C_4$.

Assume γ_i to be the estimator to ζ_1 generated by e_i. Because $r_{\max} \geq t - t_0$, we have

$$r_{\max} = d(e_i, \gamma_i) = d(e_i, h_i) + d(h_i, \gamma_i).$$

We have

$$\begin{aligned} & d(\gamma_i, \zeta_1) \\ \leqslant & d(\gamma_i, h_i) + d(\zeta_1, h_i) \\ = & d(e_i, \gamma_i) - d(e_i, h_i) + d(\zeta_1, h_i) \\ = & d(e_i, \gamma_i) - (d(e_i, \zeta_1) - d(h_i, \zeta_j)) + d(\zeta_1, h_i) \\ \leq & t - (t - C_4 - C_4) + C_4 \\ \leq & 3C_4. \end{aligned} \tag{A.67}$$

Finally, we have $d(\gamma_i, \zeta_1) \leq 3C_4$, which means the estimator we find is in a constant distance to the surviving original source.

A.2 Reverse Infection Algorithm

The key idea of the reverse infection algorithm in Zhu and Ying (2013) is to let each observed infected node to broadcast its identity (ID) to its neighbors. The set of nodes who first receive all IDs of the infected nodes are declared to be Jordan infection centers. The pseudocode is described in Algorithm A.1.

Algorithm A.1 Reverse Infection Algorithm

for $i \in \mathcal{V}_I$ **do**
  i sends its ID w_i to its neighbors.
end for
while $t \geq 1$ and STOP=0 **do**
  for $u \in \mathcal{V}$ **do**
    if u receives w_i for the first time **then**
      set $t_{ui} = t$, where t is the current time slot, and then broadcast the message w_i to its neighbors.
      if there exists a node who received $\mathcal{V}_I$ distinct messages **then**
        set STOP=1.
      end if
    end if
  end for
end while
return $u^\star = \arg\min_{s \in \mathcal{S}} \sum_{i \in \mathcal{V}_I} t_{ui}$, where $\mathcal{S}$ is the set of nodes who receive $\mathcal{V}_I$ distinct messages when the algorithm terminates. Ties are broken at random.

APPENDIX B

PROOF OF CHAPTER 3

B.1 Proof of E_1 and E_2

Lemma B.1. *Assume the conditions in Theorem 2 hold, for any $\epsilon > 0$, we have*

$$\Pr(E_1) \geq 1 - \epsilon,$$

and

$$\Pr(E_2|E_1) \geq 1 - \epsilon,$$

for suffciently large n.

The proof follows directly from the proof of Lemma 3 and 4 in Zhu and Ying (2015b).

B.2 Proof of E_3

Lemma B.2. *Assume the conditions in Theorem 2 hold, for any $\epsilon > 0$, we have*

$$\Pr(E_3|E_1) \geq 1 - \epsilon,$$

for suffciently large n.

Proof. Since all sources are within D hops from s_1 and the snapshot is taken at time t, all the infected nodes are within $t + D$ hops from s_1. To prove the conclusion, we only need to show that any node on level $t+D+1$ does not have more than $(1-\delta)^3\mu q\theta$ neighbors. Since all infected nodes are within level $t + D$, instead of considering the observed infected neighbors, we only need to show that any node on level $t + D + 1$ does not have more than $(1 - \delta)^3\mu q\theta$ neighbors in level $t + D$.Therefore, in this proof, we consider a more restrictive event which is only a topological feature of the ER random and does not depend on the infection process.

Based on E_1, there are at most $[(1 + \delta)\mu]^{t+D}$ nodes in level $t + D$ and at most $[(1+\delta)\mu]^{t+D+1}$ nodes in level $t+D+1$. For any node v in level $t+D+1$, the neighbors of node v in $t+D$ are either the node which introduce node v into level $t+D+1$ (i.e., the parent of v in the BFS tree) or the collision edges between node v and nodes in level $t + D$. The total number of possible collision edges depends on the order that the parent of node v is introduced to the BFS tree.

In general, if the parent of node v is the ith node, the number of possible neighbors on level $t + D$ follows $\mathrm{Bi}([(1 + \delta)\mu]^{D+t} - i, \mu/n) + 1$. As a summary, for any node on level $t + D + 1$ the number of neighbors in level $t + D$ is stochastically upper bounded by $\mathrm{Bi}([(1 + \delta)\mu]^{D+t}, \mu n) + 1$. Define

$$X \triangleq (1 - \delta)^3\mu q\theta - 1.$$

Define δ' to be

$$\delta' \triangleq \frac{Xn}{[(1 + \delta)\mu]^t\mu} - 1.$$

Denote by N_v the number of neighbors in level $t+D$ for one node v on level $t+D+1$.

$$\begin{aligned}
&\Pr(N_v \geq X+1|E_1) && \text{(B.1)}\\
\leq &\exp\left(-\frac{\delta'^2[(1+\delta)\mu]^{t+D}\mu/n}{2+\delta'}\right) && \text{(B.2)}\\
= &\exp\left(-\frac{\delta'}{2+\delta'}\times\delta'[(1+\delta)\mu]^{t+D}\mu/n\right) && \text{(B.3)}\\
= &\exp\left(-\frac{\delta'}{2+\delta'}\times\left(\frac{Xn}{[(1+\delta)\mu]^{t+D}\mu}-1\right)[(1+\delta)\mu]^{t+D}\mu/n\right) && \text{(B.4)}\\
= &\exp\left(-\frac{\delta'}{2+\delta'}\times\left(\frac{X}{[(1+\delta)\mu]^{t+D}\mu}[(1+\delta)\mu]^{t+D}\mu-[(1+\delta)\mu]^{t+D}\mu/n\right)\right) && \text{(B.5)}\\
= &\exp\left(-\frac{\delta'}{2+\delta'}\times\left(X-[(1+\delta)\mu]^{t+D}\mu/n\right)\right) && \text{(B.6)}\\
= &\exp\left(-\frac{\delta'}{2+\delta'}\times\left((1-\delta)^3\mu q\theta-1-[(1+\delta)\mu]^{t+D}\mu/n\right)\right) && \text{(B.7)}\\
= &\exp\left(-\frac{\delta'\mu}{2+\delta'}\left((1-\delta)^3 q\theta-1/\mu-[(1+\delta)\mu]^{t+D}/n\right)\right) && \text{(B.8)}\\
\leq &\exp\left(-\frac{\delta'(1-\delta)^3 q\theta\mu}{2(2+\delta')}\right) && \text{(B.9)}\\
\leq &\exp\left(-\frac{(1-\delta)^3 q\theta}{6}\mu\right) && \text{(B.10)}
\end{aligned}$$

Since $t+D<\frac{\log n}{(1+\alpha)\log\mu}$, we have

$$t+D\leq\frac{\log n}{\log\mu}\times\frac{1-\frac{\log\mu}{\log n}}{1+\frac{\log(1+\delta)}{\log\mu}}. \tag{B.11}$$

Hence,

$$\frac{[(1+\delta)\mu]^{t+D}}{n}\leq\frac{1}{\mu} \tag{B.12}$$

Hence, Inequality (B.9) is based on Inequality (B.12) and Inequality (B.10) is based on $\delta'\geq 1$ for sufficiently large n. For any node in level $t+D+1$, we have

$$\begin{aligned}
&\Pr\left(\cap_v N_v<X+1|E_1\right)\\
=&1-\Pr\left(\cup_v N_v\geq X+1|E_1\right)\\
\geq&1-\sum_v\Pr\left(N_v\geq X+1|E_1\right)\\
\geq&1-[(1+\delta)\mu]^{t+D+1}\exp\left(-\Omega(\mu)\right)\\
\geq&1-\exp\left((t+D+1)\log[(1+\delta)\mu]-\frac{(1-\delta)^3 q\theta}{6}\mu\right)
\end{aligned}$$

Note we have $t + D \leq \frac{\log n}{(1+\alpha)\log \mu}$. Therefore,

$$\begin{aligned}
&\Pr\left(\cap_v N_v < X + 1 | E_1\right) \\
\geq & 1 - \exp\left(((t+D)\log[(1+\delta)] + (t+D)\log\mu + \log[(1+\delta)\mu]) - \frac{(1-\delta)^3 q\theta}{6}\mu\right) \\
\geq & 1 - \exp\left(\frac{\log n}{(1+\alpha)\log\mu}\log[(1+\delta)] + \frac{\log n}{(1+\alpha)} + \log[(1+\delta)\mu] - \frac{(1-\delta)^3 q\theta}{6}\mu\right)
\end{aligned}$$

Let $C \leq \frac{6}{(1-\delta)^3(1+\alpha)}$ and since $\mu > \frac{1}{Cq\theta}\log n$, we have

$$\Pr\left(\cap_v N_v < X + 1\right) \geq 1 - \exp\left(-\Omega(\mu)\right)$$

for sufficiently large n. By substituting X, we proved the lemma. □

B.3 Proof of E_4 and E_5

B.3.1 Neighboring structure of all sources

To prove E_4 and E_5 happen with a high probability, we first analyze the neighborhood of all sources in the ER random graph. In this section, we derived upper and lower bounds of the t neighborhood of all sources. Define $\mathcal{L}_l^i$ the set of nodes from level 0 to level l of the BFS tree rooted in source s_i. In addition, define $\phi_i'(v)$ the number of offsprings of node v on the BFS tree rooted in source s_i. Define

$$E_1^i = \{\forall v \in \mathcal{L}_{t-1}^i, \phi_i'(v) \in ((1-\delta)\mu, (1+\delta)\mu)$$

Denote by the event

$$\tilde{E} = \cap_{i=2}^m E_1^i \cap E_1.$$

We have the following lemma.

Lemma B.3. *If the conditions in Theorem 2 hold, for any $\epsilon > 0$,*

$$\Pr(\tilde{E}) \geq 1 - \epsilon$$

for sufficiently large n.

Proof. For each infection source s_i, follow the similar argument of Lemma 3 in Zhu and Ying (2015b), we have

$$\Pr(E_1^i) \geq \exp\left(-8\exp\left(-\frac{\delta^2\mu}{2+\delta} + (t-1)\log[(1+\delta)\mu]\right)\right) \geq 1 - \frac{8\left(\mu\left(1+\delta\right)\right)^{t-1}}{\exp\left(\frac{\delta^2\mu}{2+\delta}\right)}$$

Hence, we have

$$\Pr(\bar{E}_1^i) \leq \frac{8\left(\mu\left(1+\delta\right)\right)^{t-1}}{\exp\left(\frac{\delta^2\mu}{2+\delta}\right)}$$

Therefore, with an union bound, we have

$$\begin{aligned}\Pr(\cap_{i=2}^{m} E_1^i) &\geq 1 - \sum_{i=1}^{m} \Pr(\bar{E}_1^i) \\ &\geq 1 - \frac{8(m-1)\left(\mu\left(1+\delta\right)\right)^{t-1}}{\exp\left(\frac{\delta^2\mu}{2+\delta}\right)} \\ &\geq 1 - \exp\left(\log 8(m-1) + (t-1)\log\left(\mu(1+\delta)\right) - \frac{\delta^2\mu}{2+\delta}\right)\end{aligned}$$

To make the probability larger than $1-\epsilon$, we have

$$t \leq \frac{\log \frac{\epsilon}{8(m-1)} + \frac{\delta^2\mu}{2+\delta}}{\log(\mu(1+\delta))} + 1$$

Again, we have $t + D < \frac{\log n}{(1+\alpha)\log \mu}$ and $\mu > \frac{1}{Cq\theta}\log n > 3\log n$ which guarantees the probability goes to 1 asymptotically.

Note the events $\cap_{i=2}^{m} E_1^i$ do not contain the neighborhood of source s_1. Based on Lemma B.1, with a union bound, it is straightforward to show that

$$\Pr\left(\cap_{i=2}^{m} E_1^i \cap E_1\right) > 1 - \epsilon.$$

for sufficiently large n. Hence the lemma is proved.

□

B.3.2 Proof of E_4 and E_5

Lemma B.4. *If the conditions in Theorem 2 hold, for any $\epsilon > 0$,*

$$\Pr(E_4, E_5) \geq 1 - \epsilon$$

for sufficiently large n.

Proof. We still consider the BFS tree rooted at source s_1 and we can rewrite the events E_4 and E_5 in a combined fashion

$$E_4 \cap E_5 = \{\forall v \in \cup_{i=0}^{t-1} \mathcal{Z}_i, \psi'(v) \geq (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta\}.$$

Define

$$\mathcal{F}_i = \{\mathcal{Z}_i | \forall v \in \mathcal{Z}_i, \psi'(v) \geq (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta\}$$

Then, we have

$$\begin{aligned}&\Pr(E_4, E_5) \\ \geq &\Pr(E_4, E_5|\tilde{E})\Pr(\tilde{E})\end{aligned}$$

$$
\begin{aligned}
&\Pr(E_4, E_5|\tilde{E}) \\
=&\Pr(\forall v \in \cup_{i=0}^{t-1}\mathcal{Z}_i, \psi'(v) \geq (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta|\tilde{E}) \\
=&\sum_{\mathcal{Z}_1\in\mathcal{F}_1}\cdots\sum_{\mathcal{Z}_{t-1}\in\mathcal{F}_{t-1}}\Pr(\forall v \in \mathcal{Z}_{t-1}, \psi'(v) > (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta \\
&|\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E})\Pr(\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1|\tilde{E})
\end{aligned}
$$

We have

$$
\begin{aligned}
&\Pr(\forall v \in \mathcal{Z}_{t-1}, \psi'(v) > (1-\delta)^2\mu q|\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \\
\geq&1 - \sum_{v\in\mathcal{Z}_{t-1}}\Pr(\psi'(v) \leq (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta|\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E})
\end{aligned}
$$

Note conditioned on $\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1$, consider an offspring u of node v on the BFS tree rooted at source s_1. Node u has two possible states: infected or susceptible. If u is not infected, v will infect node u with probability q in the next time slot. On the other hand, if u is infected, it counts as an infected offspring of node v deterministically. Therefore, $\psi'(v)$ is stochastically lower bounded by binomial distribution $B((1-\delta)\mu, q)$. Therefore, with Chernoff bound in Zhu and Ying (2015b), we have

$$
\Pr(\psi'(v) \leq (1-\delta)^2\mu q|\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \leq \exp\left(-\frac{\delta^2(1-\delta)\mu q}{2}\right)
$$

Each infected nodes are observed with probability θ independently. Conditioned on $\psi'(v) \geq (1-\delta)^2\mu q$, $\psi''(v)$ is stochastically lower bounded by $B((1-\delta)^2\mu q, \theta)$. Therefore,

$$
\begin{aligned}
&\Pr(\psi''(v) \leq (1-\delta)^3\mu q\theta|\psi'(v) \geq (1-\delta)^2\mu q, \mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \\
\leq&\exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)
\end{aligned} \tag{B.13}
$$

Therefore, we have

$$
\begin{aligned}
&\Pr(\psi''(v) \geq (1-\delta)^3\mu q\theta, \psi'(v) \geq (1-\delta)^2\mu q|\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \\
=&\left(1-\exp\left(-\frac{\delta^2(1-\delta)\mu q}{2}\right)\right)\left(1-\exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \\
\geq&1-\exp\left(-\frac{\delta^2(1-\delta)\mu q}{2}\right)-\exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right) \\
\geq&1-2\exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)
\end{aligned}
$$

Again with union bound, we have

$$
\begin{aligned}
&\Pr(\forall v \in \mathcal{Z}_{t-1}, \psi'(v) > (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta|\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \\
\geq&1-2|\mathcal{Z}_{t-1}|\exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)
\end{aligned}
$$

Note, based on event $\tilde{E}$, we have

$$|\mathcal{Z}_{t-1}| \leq m \sum_{i=0}^{t-1} [(1+\delta)\mu]^i \leq 2m[(1+\delta)\mu]^{t-1}$$

Hence, we have

$$\begin{aligned}
&\Pr(\forall v \in \mathcal{Z}_{t-1}, \psi'(v) > (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta | \mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \\
\geq & 1 - 4m[(1+\delta)\mu]^{t-1} \exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
&\Pr(E_4, E_5|\tilde{E}) \\
=& \sum_{\mathcal{Z}_1 \in \mathcal{F}_1} \cdots \sum_{\mathcal{Z}_{t-1} \in \mathcal{F}_{t-1}} \Pr(\forall v \in \mathcal{Z}_{t-1}, \psi'(v) > (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta \\
& |\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1, \tilde{E}) \Pr(\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1|\tilde{E}) \\
\geq & \sum_{\mathcal{Z}_1 \in \mathcal{F}_1} \cdots \sum_{\mathcal{Z}_{t-1} \in \mathcal{F}_{t-1}} \left(1 - 4m[(1+\delta)\mu]^{t-1} \exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \\
& \times \Pr(\mathcal{Z}_{t-1}, \mathcal{Z}_{t-2}, \cdots, \mathcal{Z}_1|\tilde{E}) \\
=& \left(1 - 4m[(1+\delta)\mu]^{t-1} \exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \\
& \times \sum_{\mathcal{Z}_1 \in \mathcal{F}_1} \cdots \sum_{\mathcal{Z}_{t-2} \in \mathcal{F}_{t-2}} \Pr(\forall v \in \mathcal{Z}_{t-2}, \psi'(v) > (1-\delta)^2\mu q, \psi''(v) \geq (1-\delta)^3\mu q\theta \\
& |\mathcal{Z}_{t-2}, \mathcal{Z}_{t-3}, \cdots, \mathcal{Z}_1, \tilde{E}) \Pr(\mathcal{Z}_{t-2}, \mathcal{Z}_{t-3}, \cdots, \mathcal{Z}_1|\tilde{E})
\end{aligned}$$

Then, iteratively apply the similar arguments, we obtain,

$$\Pr(E_4, E_5|\tilde{E}) \tag{B.14}$$

$$\geq \prod_{i=2}^{t} \left(1 - 4m[(1+\delta)\mu]^{i-1} \exp\left(-\frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \tag{B.15}$$

$$= \prod_{i=2}^{t} \left(1 - \exp\left(\log 4m + (i-1)\log[(1+\delta)\mu] - \frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \tag{B.16}$$

$$\geq \prod_{i=2}^{t} \exp\left(-2\exp\left(\log 4m + (i-1)\log[(1+\delta)\mu] - \frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \tag{B.17}$$

$$= \exp\left(-2\sum_{i=2}^{t} \exp\left(\log 4m + (i-1)\log[(1+\delta)\mu] - \frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\right) \tag{B.18}$$

$$= \exp\left(-2\exp\left(\log 4m - \frac{\delta^2(1-\delta)^2\mu q\theta}{2}\right)\sum_{i=1}^{t-1} \exp\left(i\log[(1+\delta)\mu]\right)\right) \tag{B.19}$$

$$\geq \exp\left(-4\exp\left(\log 4m - \frac{\delta^2(1-\delta)^2\mu q\theta}{2} + (t-1)\log[(1+\delta)\mu]\right)\right) \tag{B.20}$$

To make the probability greater than $1-\epsilon$, we need,

$$t \leq 1 + \frac{\log\log(1-\epsilon)^{-1/4} - \log 2m + \frac{\delta^2(1-\delta)^2\mu q\theta}{2}}{\log((1+\delta)\mu)} \tag{B.21}$$

Since we have $t + D < \frac{\log n}{(1+\alpha)\log\mu}$ and $\mu > \frac{1}{Cq\theta}\log n > \frac{2}{\delta^2(1-\delta)^2 q\theta}\log n$, Inequality B.21 is satisfied when n is large enough.

Therefore, based on Lemma B.3 and Inequality B.20, we showed that

$$\Pr(E_4, E_5) \geq \Pr(E_4, E_5|\tilde{E})\Pr(\tilde{E}) \geq 1 - \epsilon$$

for any $\epsilon > 0$ when n is sufficiently large. □

B.4 Proof of E_6

Lemma B.5. *If the conditions in Theorem 2 hold, for any $\epsilon > 0$,*

$$\Pr(E_6|E_1) \geq 1 - \epsilon$$

for sufficiently large n.

Proof. Define

$$E_7 = \{\tilde{Z}_1^1 \geq (1-\delta)^2\mu q\} \cap \{\forall v \in \tilde{Z}_1^1, \cap_{i=2}^{t} \tilde{Z}_i^i(v) \geq (1-\delta)^2\mu q \tilde{Z}_{i-1}^{i-1}(v)\}$$

Following the similar arguments in Lemma 5 in Zhu and Ying (2015b), we have

$$\Pr(E_7|E_1) \geq 1 - \epsilon.$$

Note, for each node $v \in \tilde{Z}_1^1$, based on event E_7, we have $\tilde{Z}_t^t(v) \geq [(1-\delta)^2\mu q]^{t-1}$. Recall each infected node report its status independently. Therefore, $\tilde{Z}_t'^t(v)$ is stochastically lower bounded by $\mathrm{Bi}([(1-\delta)^2\mu q]^{t-1}, \theta)$. By Chernoff bound, we have

$$\Pr(\tilde{Z}_t'^t(v) \geq [(1-\delta)^2\mu q]^{t-1}(1-\delta)\theta | E_7, E_1) \leq \exp\left(-\frac{\delta^2[(1-\delta)^2\mu q]^{t-1}\theta}{2}\right)$$

Note $\tilde{Z}_1^1 \leq (1+\delta)\mu$ based on event E_1, with a union bound, we have

$$\begin{aligned}
&\Pr(\forall v \in \tilde{\mathcal{Z}}_1^1, \tilde{Z}_t'^t(v) \geq [(1-\delta)^2\mu q]^{t-1}(1-\delta)\theta | E_7, E_1) \\
\geq & 1 - (1+\delta)\mu \exp\left(-\frac{\delta^2[(1-\delta)^2\mu q]^{t-1}\theta}{2}\right) \\
\geq & 1 - \exp\left(\log[(1+\delta)\mu] - \frac{\delta^2[(1-\delta)^2\mu q]^{t-1}\theta}{2}\right) \\
\geq & 1 - \epsilon,
\end{aligned}$$

for sufficiently large n since $\mu > \frac{1}{Cq\theta}\log n$.

Therefore, we have

$$\Pr(E_6|E_1) \geq \Pr(E_6|E_7, E_1)\Pr(E_7|E_1) \geq 1 - \epsilon.$$

The lemma is proved. □

APPENDIX C

PROOF OF CHAPTER 5

C.1 Proof of Theorem 5.1

Proof. We prove this theorem by considering a special case of the MDP problem with $p = 1$, which means the attack stops at time slot 0 and only the initial attack source gets attacked. Thus, we have

$$\mathbb{E}\sum_{t=0}^{T} f(s_t, a_t(n_t)) = f(s_0, a_0) \tag{C.1}$$

We define this as the single step MDP problem. If we can show the single step MDP problem is NP-hard, the MDP problem is also NP-hard. The NP-hardness is shown by reducing the maximum coverage problem to the single step MDP problem. The decision versions of the maximum coverage problem and the single step MDP problem are as follows:

Problem 2 (The maximum coverage problem decision version)**.**
INSTANCE: *A collection of sets* $\mathcal{U} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\}$, *and integers* $k,\ h$.
QUESTION: *Is there a subset* $\mathcal{U}' \subseteq \mathcal{U}$ *such that* $|\mathcal{U}'| \leq k$ *and* $|\cup_{\mathcal{S}_i \in \mathcal{U}'} \mathcal{S}_i| \geq h$?

Problem 3. *(The single step MDP problem decision version)*
INSTANCE: $G_a(\mathcal{V}_a, \mathcal{E}_a),\ G_b(\mathcal{V}_b, \mathcal{E}_b), \mathcal{C}_0, \mathcal{F}_0,\ \mathcal{D}$, *and integers* $n_0,\ q$.
QUESTION: *Is there a rewiring such that* $f(s_0, a_0) \geq q$?

Given any instance of the maximum coverage problem $\langle \mathcal{U}, k, h \rangle$, where

$$\mathcal{U} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m\},$$

$\mathcal{S} = \cup_{\mathcal{S}_i \in \mathcal{U}} S_i$ and $n = |\mathcal{S}|$. We first build a graph $G_a(\mathcal{V}_a, \mathcal{E}_a)$ as follows:

1. For each $\mathcal{S}_i \in \mathcal{U}$, there is a node v_i on G_a.
2. For each $s_i \in \mathcal{S}$, there is a node u_i on G_a.
3. For any $s_i \in \mathcal{S}$, $\mathcal{S}_j \in \mathcal{U}$, there is an edge between u_i and v_j if $s_i \in \mathcal{S}_j$.
4. For any v_i, v_j $(1 \leq i, j \leq m)$, add edge between v_i and v_j.
5. Add node z_{ij} $(1 \leq i \leq m,\ 1 \leq j \leq n)$ to graph G_a and add edge between z_{ij} and z_{ik} for any $1 \leq i \leq m$ and $1 \leq j, k \leq n$.
6. For each node v_i $(1 \leq i \leq m)$, add nodes x_{ij} $(1 \leq j \leq nm - 1)$ and add edge between v_i and x_{ij} for $1 \leq j \leq mn - 1$.
7. Add node r and add edge between r and v_i for $1 \leq i \leq m$.

An example of G_a can be found in Figure C.1. We then construct a graph $G_b(\mathcal{V}_b, \mathcal{E}_b)$ as a graph of isolated nodes, with $|\mathcal{V}_b| = |\mathcal{V}_a|$. The set of dependency links is a one-to-one mapping between nodes from G_a and G_b. Then, we set $\mathcal{C}_0 = \{r\}$, $n_0 = k$ and $q = kmn + kn + h + 1$. Since G_b is a graph of isolated nodes in our construction, there is no connected component on G_b with size greater than 1 no matter how we rewire G_a.

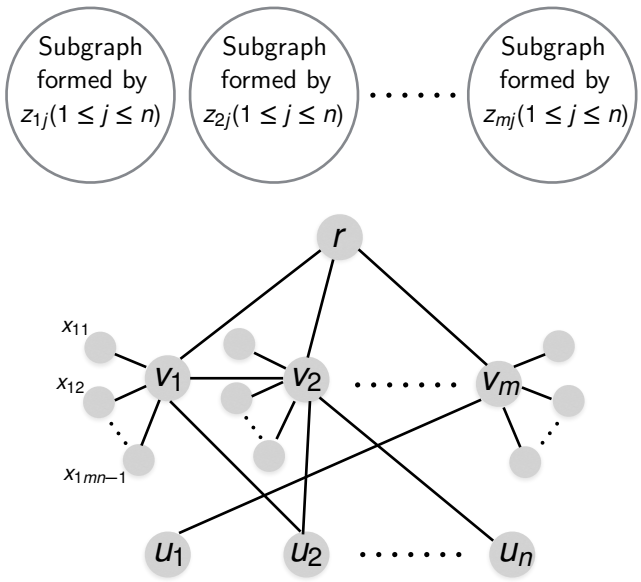


Figure C.1: An example of G_a.

If we can find a subset $\mathcal{U}'$ with $|\mathcal{U}'| = k$ such that $|\cup_{\mathcal{S}_i \in \mathcal{U}'} \mathcal{S}_i| \geq h$, then for any $\mathcal{S}_i \in \mathcal{U}'$, we remove the edge between v_i and r and add an edge between v_i and z_{i1}. Since $n_0 = k$, this is a rewiring of n_0 edges on graph G_a. According to the construction of G_a, it is easy to find out $f(s_0, a_0) \geq kmn + kn + h + 1$.

Suppose we can find a rewiring of k links on G_a such that

$$f(s_0, a_0) \geq kmn + kn + h + 1. \tag{C.2}$$

Since the contribution of graph G_b to $f(s_0, a_0)$ is 1, the contribution of graph G_a to $f(s_0, a_0)$ has to be greater than or equal to $kmn + kn + h$. We can divide the rewiring of an edge into two steps:

- Remove an edge between two nodes;
- Reattach one endpoint of the previous removed edge to another node in the graph.

The removed edges have to be among the edges (r, v_i) $(1 \leq i \leq m)$. Otherwise, nodes x_{ij} $(1 \leq j \leq n)$ connected to v_i will not contribute to the size of the largest connected component, $f(s_0, a_0(n_0))$, which means the term kmn in inequality (C.2) can not be satisfied. Define $\mathcal{V}'$ to be the set of node v_i with link (r, v_i) removed. To satisfy term kn in inequality (C.2), after removing k links between r and v_i, for each $v_i \in \mathcal{V}'$, we have to add another edge between v_i to any node z_{ij} $(1 \leq j \leq mn)$. Then according to inequality (C.2), we have

$$|\{u_j | (u_j, v_i) \in \mathcal{E}_a \ \forall 1 \leq j \leq n, \ v_i \in \mathcal{V}'\}| \geq h, \tag{C.3}$$

which means

$$|\cup_{\mathcal{S}_i \in \mathcal{U}'} \mathcal{S}_i| \geq h, \tag{C.4}$$

where $\mathcal{U}' = \{\mathcal{S}_i | v_i \in \mathcal{V}'\}$.

Therefore, we have the maximum coverage problem is polynomial-time reducible to our problem, which means the single step MDP problem is NP-hard. Thus, the MDP problem is NP-hard. □