Learning Diffusions without Timestamps

Hao Huang[1], **Qian Yan**[1], **Ting Gan**[1], **Di Niu**[2], **Wei Lu**[3]✉, **Yunjun Gao**[4]
[1] School of Computer Science, Wuhan University, China, {haohuang, qy, ganting}@whu.edu.cn
[2] Department of Electrical and Computer Engineering, University of Alberta, Canada, dniu@ualberta.ca
[3] School of Information and DEKE, MOE, Renmin University of China, China, lu-wei@ruc.edu.cn
[4] College of Computer Science and Technology, Zhejiang University, China, gaoyj@zju.edu.cn

Abstract

To learn the underlying parent-child influence relationships between nodes in a diffusion network, most existing approaches require timestamps that pinpoint the exact time when node infections occur in historical diffusion processes. In many real-world diffusion processes like the spread of epidemics, monitoring such infection temporal information is often expensive and difficult. In this work, we study how to carry out diffusion network inference without infection timestamps, using only the final infection statuses of nodes in each historical diffusion process, which are more readily accessible in practice. Our main result is a probabilistic model that can find for each node an appropriate number of most probable parent nodes, who are most likely to have generated the historical infection results of the node. Extensive experiments on both synthetic and real-world networks are conducted, and the results verify the effectiveness and efficiency of our approach.

Introduction

In real life, many underlying influence relationships among people form various diffusion networks, spreading different contents such as information, viewpoints, or even viruses. Diffusion network inference aims to uncover these influence relationships based on the spread results observed in historical diffusion processes. Therefore, this problem is fundamental in many applications such as information propagation (He et al. 2015), viral marketing (Leskovec, Adamic, and Huberman 2007), and epidemic prevention (Wallinga and Teunis 2004), in which the inferred influence relationships can intuitively illustrate the latent diffusion paths, and help users to better predict, promote or prevent future diffusion events.

Most existing approaches to diffusion network inference are based on the following basic ideas: nodes that are infected sequentially within a time interval are assumed to have influence relationships, and the previously infected ones are regarded as potential parent nodes of the subsequently infected ones. Hence, these approaches require the observed spread results used by them (known as cascades) to include the exact infection timestamps of the infected nodes in each diffusion process. To infer diffusion networks with given cascades, a major method is adopting a convex optimization framework to find influence relationships that maximize the likelihood of the cascades, and utilizing techniques, such as sequential quadratic programming (Myers and Leskovec 2010; Gomez-Rodriguez, Balduzzi, and Schölkopf 2011), block coordinate descent (Du et al. 2012), stochastic and proximal gradient methods (Gomez-Rodriguez, Leskovec, and Schölkopf 2013b; Daneshmand et al. 2014), survival theory (Gomez-Rodriguez, Leskovec, and Schölkopf 2013a), EM algorithm (Wang et al. 2014; Rong, Zhu, and Cheng 2016), and sparse recovery (Pouget-Abadie and Horel 2015), to approximate the optimal solution. While several other approaches adopt non-convex optimization frameworks, each of them finally decouples the non-convex problem into multiple smaller convex problems (Netrapalli and Sanghavi 2012; Narasimhan, Parkes, and Singer 2015; Kalimeris et al. 2018). Another effective method is transforming the problem of diffusion network inference into that of submodular optimization (Gomez-Rodriguez, Leskovec, and Krause 2010; Gomez-Rodriguez and Schölkopf 2012) by constructing a likelihood function of cascades with the property of submodularity. Then, a near-optimal solution of the submodular optimization problem can be achieved by applying greedy algorithm. It has been validated that with adequate amount of complete and correct cascades, the influence relationships can be accurately recovered, even using some simple inference approaches (Abrahao, Chierichetti, and Kleinberg 2013). In addition, some more sophisticated approaches are proposed to handle the case that cascades have partial incorrect infection timestamps (Sefer and Kingsford 2015) or miss partial snapshots of the network (He et al. 2016; Lokhov 2016).

Although the cascade-based approaches have shown their efficacy on diffusion network inference, acquiring the cascades are often expensive in many real-world diffusion processes, such as the spread of epidemics and the viral marketing campaigns. This is because, unlike tracing the infection timestamps of nodes in online social networks (e.g., Facebook and Weibo), monitoring the infections of nodes to obtain the cascades through these real-world diffusion processes is labor/resource demanding and time consuming. Furthermore, due to a few unavoidable objective factors such as a long incubation period (i.e., the time from infection to illness), the observed cascades may not reflect the exact

occurrence time of infections.

To avoid the limitation of cascade-based approaches, new techniques are required to learn diffusions without the infection timestamps. To the best of our knowledge, only two existing works have partially addressed this problem by learning either from path traces (referred to as PATH henceforth) or from the seed and resulting sets of infected nodes (referred to as S2R henceforth). In PATH (Gripon and Rabbat 2013), the learner is assumed to have all path-connected triples, i.e., three nodes that are activated along a diffusion path through the network. Although PATH has a solid mathematical foundation, the triples are not always naturally observable in practice. Even if complete and correct cascades are available, inferring exact triples is still challenging. S2R (Amin, Heidari, and Kearns 2014) calculates the lifting effect of each seed node u to another infected node v, which measures the increase in the probability of v's infection on the condition that u is previously infected. Then, S2R adds a directed edge (i.e., an influence relationship) from u to v, if u has the greatest current lifting effect to v. It's worth noting that if there is no priori knowledge on the number of edges in the network, S2R will keep adding edges until all nodes are connected.

Aiming at a more general solution to diffusion learning, in this paper, we study the problem of how to infer diffusion networks with only the final infection statuses of the nodes in historical diffusion processes, which are more readily accessible in practice. We propose an effective and efficient algorithm called TWIND (**D**iffusion **N**etwork **I**nference **W**ithout **T**imestamps) for this problem. TWIND reveals potential parent-child influence relationships by finding for each node a set of most probable parent nodes, who are most likely to have generated the observed infection results. To this end, we present a probabilistic model to quantify the possibility of inferred influence relationships given the observed final infection statuses. Based on the model, we can also theoretically derive an upper limit on the amount of most probable parent nodes for each node in the network, which helps TWIND to prevent its inferred diffusion network from containing too many low-probability influence relationships not in the original diffusion network. Furthermore, to reduce redundant computation during the execution of TWIND, we disqualify the insignificant candidate parent nodes whose infections have rather weak correlations with the infections of the corresponding child nodes, and exclude them from the selection of most probable parent nodes.

In summary, our key contributions include the following: (1) We propose a new infection timestamp-free approach for diffusion network inference. To execute this approach, there is no need to monitor the infection timestamps of nodes through each diffusion process, or to worry about the correctness of observed infection timestamps. Except for the final infection statuses of nodes, the approach does not need any other information of infections or priori knowledge on the network. (2) We theoretically guarantee that TWIND will find for each node an limited number of most probable parent nodes, avoiding an overly complex inferred network.

In what follows, we first present our problem statement, and then elaborate our proposed TWIND algorithm, followed by reporting experimental results and our findings before concluding the paper.

Problem Statement

A diffusion network can be represented as a directed graph $G = \{V, E\}$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of n nodes in the network, and E is the set of m directed edges (i.e., influence relationships) between the nodes. A directed edge from a parent node v_i to a child node v_j indicates that when v_i is infected and v_j is uninfected, v_i will successfully infect v_j with a certain probability (which can be regarded as the edge weight of this directed edge). As a few existing approaches have presented how to calculate the edge weight based on observed infection status results for a specified edge (Yan et al. 2017), in this paper, we focus on inferring the unknown directed edge set of the objective network. Our problem can be formulated as follows.

***Given*:** a set $S = \{S^1, ..., S^\beta\}$ of infection status results observed on a diffusion network G in β diffusion processes, where $S^\ell = \{s_1^\ell, ..., s_n^\ell\}$ is a n-dimensional vector that records the final infection status $s_i^\ell \in \{0, 1\}$ (1 for infected status and 0 for uninfected status) of each node $v_i \in V$ observed in the ℓ-th diffusion process ($\ell \in \{1, \ldots, \beta\}$).

***Infer*:** the edge set E of the diffusion network G.

The TWIND Algorithm

In this section, we first introduce how to identify the most probable parent nodes for each node in the network, followed by a theoretical analysis of the upper limit on the amount of most probable parent nodes, and then we present how to reduce redundant computation during the identification of most probable parent nodes before giving the detailed steps of our TWIND algorithm. We conclude this section with a complexity analysis on our approach.

Identification of Most Probable Parent Nodes

Let matrix $A \in \mathbb{R}^{n \times n}$ be a network structure variable, of which each element $A_{ij} \in \{0, 1\}$ ($i, j \in \{1, \ldots, n\}$) indicating whether there is a directed edge from node v_i to node v_j (1 for yes, 0 for no), diffusion network inference using infection status results S is equivalent to finding a optimal A that maximizes the following conditional probability:

$$P(A \mid S) = \frac{P(A, S)}{\sum_{A' \in Q} P(A', S)} \tag{1}$$

where set Q is the set of all possible matrices of A, and the value of $\sum_{A' \in Q} P(A', S)$ is a certain constant. Maximizing probability $P(A \mid S)$ is equivalent to maximizing the joint probability $P(A, S)$, which can be calculated as follows.

$$\begin{aligned} P(A, S) &= \int_B P(S|A, B) f(B|A) P(A) \mathrm{d}B \\ &= P(A) \int_B P(S|A, B) f(B|A) \mathrm{d}B \end{aligned} \tag{2}$$

where B is a $n \times n$ block matrix related to A. If $A_{ij} = 0$, then B_{ij} is a 2×2 zero matrix; if $A_{ij} = 1$, then B_{ij} is a 2×2 nonnegative matrix, of which each element refers to a

conditional probability $P(X_j \mid X_i) \geqslant 0$, where $X_i \in \{0,1\}$ and $X_j \in \{0,1\}$ are the infection status variables of nodes v_i and v_j, respectively.

Since each historical diffusion process is independent to each other, each S^ℓ is generated independently. Moreover, since the infection of each node can be only affected by its parent nodes during each diffusion process, the relationship $P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i \mid X_{F_i})$ holds, where F_i is the parent node set of node v_i, and X_{F_i} is the infection status variables of the parent nodes of v_i. Then, we can reformulate the probability $P(S|A,B)$ in Eq. (2) as follows.

$$\begin{aligned} P(S|A,B) &= \prod\nolimits_{\ell=1}^{\beta} P(S^\ell \mid A,B) \\ &= \prod\nolimits_{\ell=1}^{\beta} P(X_1 = s_1^\ell,\ldots,X_n = s_n^\ell \mid A,B) \\ &= \prod\nolimits_{\ell=1}^{\beta} \prod\nolimits_{i=1}^{n} P(X_i = s_i^\ell \mid X_{F_i} = \pi_i^\ell, B) \\ &= \prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} \prod_{k=1}^{2} P(X_i = s_k \mid X_{F_i} = \pi_{ij}, B)^{N_{ijk}} \\ &= \prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} \prod_{k=1}^{2} \theta_{ijk}^{N_{ijk}} \end{aligned} \tag{3}$$

where π_i^ℓ refers to the infection statuses of v_i's parent nodes in the ℓ-th diffusion process, $s_k \in \{0,1\}$ is the k-th possible infection status of a node (without loss of generality, $s_1 = 0, s_2 = 1$), $2^{|F_i|}$ refers to the number of all possible combinations of the infection statuses of v_i's parent nodes, π_{ij} is the corresponding j-th possible combination, N_{ijk} is the number of times situation $X_i = s_k \wedge X_{F_i} = \pi_{ij}$ appears in S, θ_{ijk} is equal to $P(X_i = s_k \mid X_{F_i} = \pi_{ij}, B)$, and $\forall v_i, \sum_{j=1}^{2^{|F_i|}} \sum_{k=1}^{2} N_{ijk} = \beta, \theta_{ij1} + \theta_{ij2} = 1$.

Let $f(\theta_{ij1}, \theta_{ij2})$ denote the probability density function of $(\theta_{ij1}, \theta_{ij2})$. Since there is no correlation between the influences of different parent nodes to a same child node (or a same parent node to different child nodes), relationship $f(B|A) = \prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} f(\theta_{ij1}, \theta_{ij2})$ holds. Combining it with Eq. (3), we can reformulate the calculation of probability $P(A,S)$ as follows.

$$\begin{aligned} &P(A,S) \\ &= P(A) \int_B \left(\begin{array}{l} \left(\prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} \prod_{k=1}^{2} \theta_{ijk}^{N_{ijk}} \right) \\ \times \left(\prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} f(\theta_{ij1}, \theta_{ij2}) \right) \end{array} \right) \mathrm{d}B \\ &= P(A) \prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} \iint_{\theta_{ij1},\theta_{ij2}} \left(\begin{array}{l} \prod_{k=1}^{2} \theta_{ijk}^{N_{ijk}} \\ \times f(\theta_{ij1}, \theta_{ij2}) \end{array} \right) \mathrm{d}\theta_{ij1} \mathrm{d}\theta_{ij2} \end{aligned} \tag{4}$$

As there is no prior knowledge on the value of θ_{ijk}, we are indifferent to regard every possible value of θ_{ijk}. In other words, we assume that distribution $f(\theta_{ij1}, \theta_{ij2})$ is uniform, for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant 2^{|F_i|}$. Then, the value of $f(\theta_{ij1}, \theta_{ij2})$, denoted as c_{ij}, is a constant. According to the property of probability density function, we have

$$\begin{aligned} &\iint_{\theta_{ij1},\theta_{ij2}} f(\theta_{ij1}, \theta_{ij2}) \mathrm{d}\theta_{ij1} \mathrm{d}\theta_{ij2} \\ &= \iint_{\theta_{ij1},\theta_{ij2}} c_{ij} \mathrm{d}\theta_{ij1} \mathrm{d}\theta_{ij2} = 1 \end{aligned} \tag{5}$$

According to Dirichlet's integral, we have

$$\iint_{\theta_{ij1},\theta_{ij2}} \mathrm{d}\theta_{ij1} \mathrm{d}\theta_{ij2} = \frac{1}{(2-1)!} = 1 \tag{6}$$

$$\begin{aligned} &\iint_{\theta_{ij1},\theta_{ij2}} \left(\prod_{k=1}^{2} \theta_{ijk}^{N_{ijk}} \right) \mathrm{d}\theta_{ij1} \mathrm{d}\theta_{ij2} \\ &= \frac{N_{ij1}! \cdot N_{ij2}!}{(N_{ij1} + N_{ij2} + 2 - 1)!} \end{aligned} \tag{7}$$

Combining Eqs. (5) & (6), we have $f(\theta_{ij1}, \theta_{ij2}) = c_{ij} = 1$. Combining this conclusion with Eq. (7), we can finally formulate the calculation of probability $P(A,S)$ as follows.

$$P(A,S) = P(A) \prod_{i=1}^{n} \prod_{j=1}^{2^{|F_i|}} \frac{N_{ij1}! \cdot N_{ij2}!}{(N_{ij1} + N_{ij2} + 1)!} \tag{8}$$

To estimate $P(A,S)$, we need a prior probability $P(A)$ for each possible network structure, and the numbers N_{ij1} and N_{ij2} determined by each node v_i and its parent nodes F_i in corresponding network structure. N_{ij1} and N_{ij2} can be counted from the observed infection status results S, while there is no prior knowledge on the network structure to estimate $P(A)$. Furthermore, as there are $2^{n(n-1)}$ possible network structures, it is not feasible to apply Eq. (8) for each possible network structure when n is large. Therefore, we equally treat each possible network structure by assuming equal priors on A, i.e., the prior probability $P(A)$ is equal to a certain constant. Then, to maximize $P(A,S)$, what we need to do is finding for each node v_i a optimal parent node set F_i that maximizes the following scoring function.

$$\begin{aligned} g(v_i, F_i) &= \log \prod_{j=1}^{2^{|F_i|}} \frac{N_{ij1}! \cdot N_{ij2}!}{(N_{ij1} + N_{ij2} + 1)!} \\ &= \sum_{j=1}^{2^{|F_i|}} \left(\begin{array}{l} \log N_{ij1}! + \log N_{ij2}! \\ - \log(N_{ij1} + N_{ij2} + 1)! \end{array} \right) \end{aligned} \tag{9}$$

where the base of $\log$ is 2. Then, the nodes in this optimal F_i are regarded as the most probable parent nodes of v_i.

Given the above scoring function, we can utilize greedy search to find the most probable parent nodes for v_i. It starts from an empty parent node set F_i, and expands the set F_i by incrementally adding a node combination (i.e., a subset of $V \backslash \{v_i\}$) that can most increase the value of current $g(v_i, F_i)$ until this value dose not increase. In this way, we can efficiently achieve a local optimal solution of F_i. A similar greedy-search strategy is commonly used in many other applications, such as influence maximization (Tang, Xiao, and Shi 2014) and classification (Huang et al. 2014), due to its high efficiency and nice search performance.

Upper Limit on Amount of Parent Nodes

At the beginning of the greedy search for most probable parent nodes of a node v_i, $F_i = \emptyset$ and current $g(v_i, F_i)$ can be calculated as follows.

$$g(v_i, \emptyset) = \sum_{k=1}^{2} \log N_{ik}! - \log(\beta + 1)! \tag{10}$$

where N_{ik} is the number of times situation $X_i = s_k$ appears in S, and $\sum_{k=1}^{2} N_{ik} = \beta$. Since $(\frac{N}{e})^N < N! < e(\frac{N}{2})^N$ always holds for any positive integer N, we can deduce a lower bound on the value of $g(v_i, \emptyset)$ as follows.

$$\begin{aligned} g(v_i, \emptyset) &> \sum_{k=1}^{2} \log \left(\frac{N_{ik}}{e}\right)^{N_{ik}} - \log e \left(\frac{\beta+1}{2}\right)^{\beta+1} \\ &= \sum_{k=1}^{2} N_{ik} \log \frac{N_{ik}}{e} - \log e - (\beta+1) \log \frac{\beta+1}{2} \\ &= \sum_{k=1}^{2} N_{ik} \log \frac{\beta}{e} + \beta \sum_{k=1}^{2} \frac{N_{ik}}{\beta} \log \frac{N_{ik}}{\beta} - \log e \\ &\quad - (\beta+1) \log \frac{\beta+1}{2} \\ &= \beta \log \frac{\beta}{e} - \beta H(X_i) - \log e - (\beta+1) \log \frac{\beta+1}{2} \end{aligned} \tag{11}$$

where $H(X_i)$ is the entropy of variable X_i.

When the greedy search method adds a few nodes into set F_i (i.e., $F_i \neq \emptyset$), the following inequality should hold.

$$g(v_i, F_i) > g(v_i, \emptyset) \tag{12}$$

Moreover, although there are $2^{|F_i|}$ possible combinations of the infection statuses of nodes in F_i, some combinations may not have instances in the observed infection status results S. We denote the number of these non-existent combinations as δ_i. It can be obtained by checking how many of the $2^{|F_i|}$ possible combinations have instances in S. As each of these existent combinations has at least one instance in S, i.e., $N_{ij1}+N_{ij2} \geqslant 1$, we can have relationship $\sum_{j=1}^{2^{|F_i|}} \log(N_{ij1} + N_{ij2} + 1) \geqslant \sum_{j=1}^{2^{|F_i|}-\delta_i} \log(1+1)$. In addition, given the fact that $N! \leqslant N^N$, we can deduce a upper bound on the value of $g(v_i, F_i)$ as follows.

$$\begin{aligned} g(v_i, F_i) &\leqslant \sum_{j=1}^{2^{|F_i|}} \left(\sum_{k=1}^{2} \log {N_{ijk}}^{N_{ijk}} \right) - \sum_{j=1}^{2^{|F_i|}-\delta_i} \log 2 \\ &= \sum_{j=1}^{2^{|F_i|}} \sum_{k=1}^{2} N_{ijk} \log \left(\beta \frac{N_{ijk}}{\beta} \right) - (2^{|F_i|} - \delta_i) \\ &= \sum_{j=1}^{2^{|F_i|}} \sum_{k=1}^{2} N_{ijk} \log \beta - 2^{|F_i|} + \delta_i \\ &\quad - \left(-\beta \sum_{j=1}^{2^{|F_i|}} \sum_{k=1}^{2} \frac{N_{ijk}}{\beta} \log \frac{N_{ijk}}{\beta} \right) \\ &= \beta \log \beta - 2^{|F_i|} + \delta_i - \beta H(X_i, X_{F_i}) \end{aligned} \tag{13}$$

where $H(X_i, X_{F_i})$ is the entropy of variables X_i and X_{F_i}.

Combining Eqs. (11)–(13), we have relationship

$$\begin{aligned} &\beta \log \beta - \beta H(X_i, X_{F_i}) - 2^{|F_i|} + \delta_i \\ &> \beta \log \frac{\beta}{e} - \beta H(X_i) - \log e - (\beta+1) \log \frac{\beta+1}{2} \end{aligned} \tag{14}$$

which can be translated as

$$2^{|F_i|} < (\beta+1) \log \left(e \frac{\beta+1}{2} \right) + \delta_i - \beta H(X_{F_i}|X_i) \tag{15}$$

where $H(X_{F_i}|X_i) = H(X_i, X_{F_i}) - H(X_i)$ is the entropy of X_{F_i} conditioned on X_i . As relationship $H(X_{F_i}|X_i) \geqslant 0$ always holds, we have

$$|F_i| < \log \left((\beta+1) \log \left(e \frac{\beta+1}{2} \right) + \delta_i \right) \tag{16}$$

Therefore, by using the greedy search method to find a set F_i of most probable parent nodes for a node v_i, the upper limit η for the set size of F_i is $\log \left((\beta+1) \log \left(e \frac{\beta+1}{2} \right) + \delta_i \right)$. In practice, if there is enough historical data logged in S, i.e., $\beta \gg \delta_i$, we can adopt a fast estimation on the value of η by using $\eta = \left\lceil \log \left((\beta+1) \log \left(e \frac{\beta+1}{2} \right) \right) \right\rceil$.

Based on this η, each possible node combination that has the chance to be added into current F_i should satisfy the condition that when this node combination is added into current F_i, the size of new F_i will not be greater than η.

Pruning of Candidate Parent Nodes

In a given diffusion network with a node set V, each node $v_j \in V \backslash \{v_i\}$ could be a candidate parent node for node $v_i \in V$, resulting in $\sum_{i=1}^{\eta} \binom{i}{n-1}$ possible parent node combinations for v_i, where n is the number of nodes in V. To avoid redundant computation, we should prune the candidate parent nodes to reduce the number of possible parent node combinations for each node v_i in the network.

Given the fact that the infections of nodes are affected by their parent nodes, the infection statuses of the parent nodes and corresponding child nodes should have correlation. In other words, if the infection statuses of two nodes are independent or have extremely low correlation to each other, there is a very low probability that these nodes have influence relationship between them. To quantify the correlation between two variables, mutual information (abbreviated as MI) is a commonly used criterion and can be calculated as

$$MI(X_i, X_j) = P(X_i, X_j) \log \frac{P(X_i, X_j)}{P(X_i) P(X_j)}. \tag{17}$$

A greater value of $MI(X_i, X_j)$ indicates a stronger correlation between the infection statuses of nodes v_i and v_j. Furthermore, in a real-world diffusion network, each node v_i often has a finite number of parent nodes. Many other nodes in this network do not have influence relationships to v_i, and their infection statuses often have no (or very low) correlations to the infection status of v_i, resulting in very small MI values (close to 0). These very small MI values form a compact cluster with a very small mean (close to 0).

Algorithm 1: The TWIND Algorithm

Input : Node set $V = \{v_1, \ldots, v_n\}$, infection status results $S = \{S^1, \ldots, S^\beta\}$ observed on V.
Output: The diffusion network $G = \{V, E\}$.

1 $E = \emptyset$; // the set of directed edge
2 $\forall v_i \in V, v_j \in V$, calculate $MI(X_i, X_j)$ by Eq. (17);
3 Partition all MI values into two groups by K-means (with $K = 2$ and one mean fixed to 0), and set τ to the greatest MI value in the group with mean closer to 0;
4 $\eta = \left\lceil \log\left((\beta+1)\log\left(e^{\frac{\beta+1}{2}}\right)\right)\right\rceil$; //$|F_i|$'s upper limit
5 **for** each $v_i \in V$ **do**
6 $F_i = \emptyset$; // inferred parent node set of v_i
7 $P_i = \emptyset$; // candidate parent node set of v_i
8 $C_i = \emptyset$;//set of possible parent node combinations
9 **for** each $v_j \in V (j \neq i)$ **do**
10 **if** $MI(X_i, X_j) > \tau$ **then**
11 $P_i = P_i \bigcup \{v_j\}$; // merge v_j into P_i
12 **for** each $W \subseteq P_i$, $|W| \leqslant \eta$ **do**
13 Calculate $g(v_i, W)$ by Eq. (9);
14 $C_i = \{C_i, W\}$; // add a new element W to C_i
15 **while** $C_i \neq \emptyset$ **do**
16 $W^* = \arg\max_{W \in C_i} g(v_i, W)$;
17 **if** $|F_i \bigcup W^*| \leqslant \eta$ **then**
18 $F_i = F_i \bigcup W^*$;
19 $C_i = C_i \backslash W^*$;
20 $E = \{(v_j, v_i) | v_j \in F_i\} \bigcup E$; // (v_j, v_i) is directed

Inspired by this kind of situations, we can carry out a heuristic pruning method to screen out insignificant candidate parent nodes for each node. As the very small MI values form a compact cluster with a mean close to 0, we can execute K-means with $K = 2$ and fix one of the two means to 0 through all the iterations of K-means. This modified K-means algorithm can efficiently partition all MI values into two groups, in which one group has a mean very close to 0. Let τ be the greatest MI value in the group with a mean closer to 0, then for each $MI(X_i, X_j) \leqslant \tau$, we regard the corresponding node v_j as an insignificant candidate parent node for v_i, and exclude it from the candidate parent node set of v_i, since the very small MI value means that there is a high probability that v_j has no influence relationship to v_i.

Algorithm

To infer a diffusion network with observed infection status results S, we introduce how to identify the most probable parent nodes for each node in the network, deduce an upper limit on the amount of most probable parent nodes, and present a heuristic pruning method to help avoid redundant computation during the identification of most probable parent nodes. Based on these preparing work, we propose an algorithm called TWIND, which is outlined in Algorithm 1.

TWIND takes as inputs node set V of the objective diffusion network G and a set S of infection status results observed on V in β diffusion processes, and consists of two main phases, i.e., (1) the phase of pruning candidate parent nodes, which calculates the MI value for each node pair by Eq. (17) (lines 2), and performs K-means to select candidate parent nodes with greater MI values (lines 3, 9–11), and (2) the phase of greedy search for the parent node set F_i of each node v_i (lines 12–20), which first traverses all possible parent node combinations and calculates corresponding scores by scoring function proposed in Eq. (9) (lines 12–14), and then continuously expands the parent node set F_i with the highest scored parent node combinations until the size of F_i is equal to the upper bound η or there is no more possible parent node combination (lines 15–19). Finally, a directed edge from each node in F_i to v_i will be added into the inferred edge set E of the objective diffusion network G (line 20).

Complexity Analysis

In TWIND, the most computationally expensive process consists of the following two parts. (1) In the phase of pruning candidate parent nodes, calculating MI values requires $O(\beta n^2)$ time, and performing K-means on these MI values takes $O(tn^2)$ time, where n is the number of nodes in the network, β is the number of historical diffusion processes, and t is the number of iterations of K-means ($t \ll n$). (2) Since there are at most $\sum_{i=1}^{\eta} \binom{i}{\kappa} \leqslant \eta\kappa^\eta$ possible parent node combinations for each node, scoring each possible parent node combination takes at most $O(\beta\eta)$ time, scoring all of them takes at most $O(\eta^2\kappa^\eta n\beta)$ time, where $\eta \ll n$ is the upper-bound of parent node set size, and κ is the number of candidate parent nodes for each node. As the candidate parent nodes are pruned by our proposed heuristic method, κ is usually much less than n, i.e., $\kappa \ll n$.

In summary, the overall time complexity of TWIND is about $O(\beta n^2 + tn^2 + \eta^2\kappa^\eta n\beta)$, where $t \ll n$, $\eta \ll n$, and $\kappa \ll n$. Therefore, the runtime of TWIND mainly depends on the network size and the number of diffusion processes.

Experimental Evaluation

In this section, we first introduce the experimental setup, and then verify the effectiveness and efficiency of our TWIND algorithm on synthetic and real-world networks. To this end, we investigate the effects of diffusion network size, diffusion network's average degree, initial infection ratio, and the amount of diffusion processes on the accuracy performance and runtime of TWIND. All algorithms in the experiments are implemented in Java, running on a desktop PC with Intel Core i3-6100 CPU at 3.70GHz and 8GB RAM.

Experimental Setup

Network. We adopt LFR benchmark graphs (Lancichinetti, Fortunato, and Radicchi 2008) as the synthetic networks. By setting different generation parameters, such as the number of nodes and the average degree of each node, we generate two series of LFR benchmark graphs with properties summarized in Table 1. In addition, we adopt two real-world networks, i.e., NetSci (Newman 2006) which is a coauthorship network containing 379 scientists and

Table 1: Properties of LFR benchmark graphs

Graphs	Number of Nodes	Average Degree
LFR1-5	100,150,200,250,300	4
LFR6-10	200	2,3,4,5,6

1602 coauthorships, and DUNF (Wang et al. 2014) which is a microblogging network containing 750 users and 2974 following relationships.

Infection Data. The infection status results S can be obtained by simulating β times of diffusion processes on each network with randomly selected initially infected nodes in each simulation (α denotes the initial infection ratio). Corresponding cascades are also recorded for cascade-based tested algorithms in the experiments. In each diffusion process, each infected node tries to infect its uninfected child nodes with a transmission rate, which subjects to a Gaussian distribution with a mean of 0.3 and a standard deviation of 0.05, to make about 95% of transmission rate values are within a range from 0.2 to 0.4.

Performance Criterion. To evaluate the accuracy performance of TWIND algorithm, we report the F-score (i.e., the harmonic mean of precision and recall) of its inferred directed edges, which can be calculated as follows.

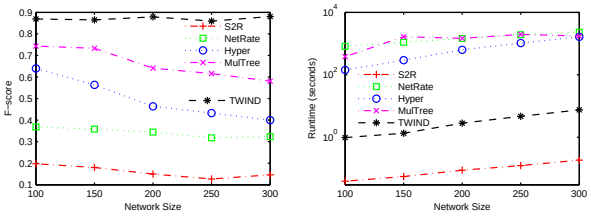
$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}}, \quad Recall = \frac{N_{TP}}{N_{TP} + N_{FN}},$$
$$F\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall},$$

where N_{TP} refers to the number of true positives, i.e., the true edges which are correctly inferred by the algorithm; N_{FP} refers to the number of false positives, i.e., the wrong inferred edges which are not in the real network; and N_{FN} refers to the number of false negatives, i.e., the true edges which are not correctly inferred by the algorithm.

Benchmark Algorithms. We compare our algorithm with a classical convex programming-based approach NetRate (Gomez-Rodriguez, Balduzzi, and Schölkopf 2011), a state-of-the-art non-convex programming-based approach using hyper-parameters (referred to as Hyper henceforth) (Kalimeris et al. 2018), a high performance submodularity-based approach MulTree (Gomez-Rodriguez and Schölkopf 2012), and an efficient infection timestamp-free approach S2R (Amin, Heidari, and Kearns 2014) for performance comparison. Since NetRate infers the transmission rate between each two node in the network, we give NetRate a privilege in accuracy performance comparison, i.e., by calculating the F-score of edges whose transmission rates are greater than a threshold, we use different thresholds to find a highest F-score and report this F-score as the final accuracy performance of NetRate. Moreover, since MulTree and S2R need users to specify the number of edges to be inferred, we use the real number m of edges in the network as an input of these two algorithms.

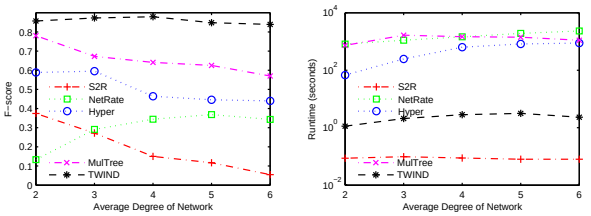
Effect of Diffusion Network Size

To study the effect of diffusion network size on algorithm performance, we adopt five synthetic networks, i.e., LFR1–5, of which the sizes vary from 100 to 300. We simulate 150 times of diffusion processes on each network (i.e., $\beta = 150$). In each simulation, $0.15n$ nodes are randomly selected as the initial infected nodes (i.e., $\alpha = 0.15$).



(a) *F-score* (b) *Runtime*

Figure 1: Effect of diffusion network size



(a) *F-score* (b) *Runtime*

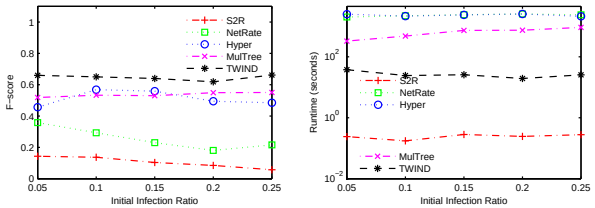
Figure 2: Effect of average degree of diffusion network

Fig. 1 illustrates the F-score and runtime of each tested algorithm, from which we can observe that (1) a greater diffusion network size tends to degrade the accuracy performance of NetRate, Hyper, MulTree and S2R, while the accuracy performance of TWIND is reasonably insensitive to diffusion network size and outperforms that of the others. (2) The runtime of each tested algorithm increases with the growth of diffusion network size. S2R executes the fastest (but with a low accuracy performance), and TWIND is reasonably more efficient than NetRate, Hyper and MulTree.

Effect of Average Degree of Diffusion Network

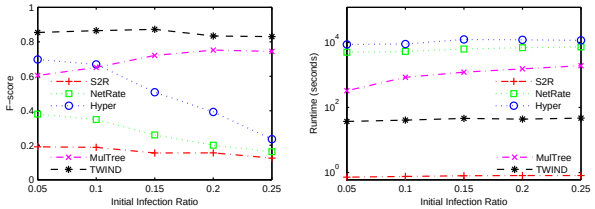
To study the effect of diffusion network's average degree on algorithm performance, we test the algorithms on five synthetic networks, i.e., LFR6–10, of which the average degrees vary from 2 to 6. We simulate 150 times of diffusion processes on each network (i.e., $\beta = 150$). In each simulation, $0.15n$ nodes are randomly selected as the initial infected nodes (i.e., $\alpha = 0.15$).

Fig. 2 illustrates the F-score and runtime of each algorithm, from which we can observe that (1) diffusion networks with greater average degrees degrade the accuracy performance of Hyper, MulTree, S2R and TWIND. The accuracy performance of NetRate increases when the average degree increases from 2 to 5, and decreases after the average degree exceeds 5. Compared with the other tested algorithms, our TWIND algorithm has a reasonably better accuracy performance. (2) The runtimes of NetRate, Hyper, MulTree, S2R and TWIND increase with the growth of average degree,



(a) *F-score* (b) *Runtime*

Figure 3: Effect of initial infection ratio on NetSci



(a) *F-score* (b) *Runtime*

Figure 4: Effect of initial infection ratio on DUNF

and TWIND shows a significant advantage on efficiency performance over NetRate, Hyper and MulTree.

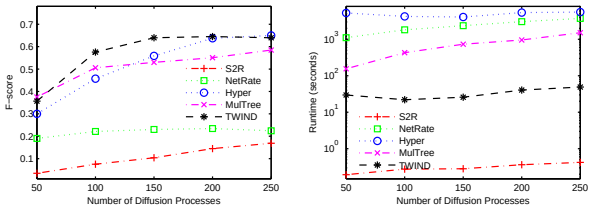
Effect of Initial Infection Ratio

The ratio of initially infected nodes may affect the number of final infected nodes. To study the effect of initial infection ratio on algorithm performance, we test the algorithms on two real-world networks NetSci and DUNF with different initial infection ratios α (vary from 0.05 to 0.25). For each initial infection ratio, we simulate 150 times of diffusion processes on each network (i.e., $\beta = 150$).

Figs. 3 & 4 illustrate the F-score and runtime of each algorithm on NetSci and DUNF, respectively. From the figures we can observe that (1) a greater initial infection ratio tends to improve the accuracy performance of MulTree, while degrading the accuracy performance of NetRate, Hyper and S2R. TWIND is reasonably insensitive to initial infection ratio and has better accuracy performance. (2) The increase of initial infection ratio has little effect on the runtime of NetRate, Hyper, S2R and TWIND, but results in more runtime for MulTree. Similar results can also be observed on synthetic networks LRF1–10.

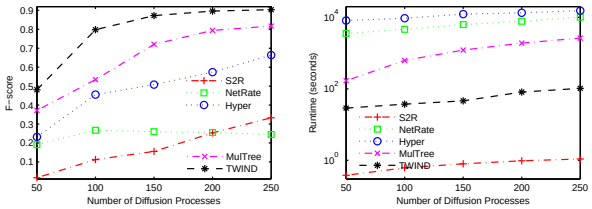
Effect of Amount of Diffusion Processes

The inference of diffusion network is based on the observed diffusion results of diffusion processes. Hence, the amount of diffusion processes may affect the accuracy performance of diffusion network inference. Generally, more diffusion processes will contain more information about diffusion network, and help the diffusion network inference algorithms to achieve more accurate inference results. To study the effect of the amount of diffusion processes on algorithm performance, we test the algorithms on two real-world networks NetSci and DUNF with different number β of diffusion processes (β varies from 50 to 250). In each diffusion process, we randomly selected $0.15n$ nodes as the initial infection nodes ($\alpha = 0.15$).



(a) *F-score* (b) *Runtime*

Figure 5: Effect of number of diffusion processes on NetSci



(a) *F-score* (b) *Runtime*

Figure 6: Effect of number of diffusion processes on DUNF

Figs. 5 & 6 illustrate the F-score and runtime of each algorithm on NetSci and DUNF, respectively. From the figures we can observe that (1) a greater amount of diffusion processes often helps the tested algorithms to achieve more accurate results on diffusion network structure inference. TWIND often has a better accuracy performance compared with the other tested algorithms. (2) To analyze the infection status results observed from more diffusion processes, the tested algorithms often require more runtime. Compared with NetRate, Hyper and MulTree, TWIND shows a significant advantage on efficiency performance. Similar results can also be observed on synthetic networks LRF1–10.

Conclusion

In this paper, we have investigated the problem of how to infer diffusion networks using only the infection statuses of nodes observed in historical diffusion processes. Towards this, we have proposed a probabilistic model to identify most probable parent nodes for each node in the objective network, and theoretically driven a upper limit on the amount of each node's most probable parent nodes. Furthermore, we have also presented a heuristic pruning method for candidate parent nodes to reduce redundant computation during the identification of the most probable parent nodes. Extensive experiments on both synthetic and real-world networks have been conducted, and the results have verified the effectiveness and efficiency of our approach.

Acknowledgments

This research was supported partly by the NSFC Grants 61502347, 61502504 and 61522208, the Technological Innovation Major Projects of Hubei Province under Grant No. 2017AAA125, Beijing Municipal Science and Technology Projects under Grant No. Z171100005117002, and the Science and Technology Program of Wuhan City under Grant No. 2018010401011288, in which Di's work was supported in part by the NSERC Canada under the grant CRDPJ 479555 Niu. Wei Lu is the corresponding author.

References

Abrahao, B.; Chierichetti, F.; and Kleinberg, R. 2013. Trace complexity of network inference. In *KDD 2013*, 491–499.

Amin, K.; Heidari, H.; and Kearns, M. 2014. Learning from contagion(without timestamps). In *ICML 2014*, 1845–1853.

Daneshmand, H.; Gomez-Rodriguez, M.; Song, L.; and Schölkopf, B. 2014. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *ICML 2014*, 793–801.

Du, N.; Song, L.; Smola, A.; and Yuan, M. 2012. Learning networks of heterogeneous influence. In *NIPS 2012*, 2780–2788.

Gomez-Rodriguez, M., and Schölkopf, B. 2012. Submodular inference of diffusion networks from multiple trees. In *ICML 2012*, 489–496.

Gomez-Rodriguez, M.; Balduzzi, D.; and Schölkopf, B. 2011. Uncovering the temporal dynamics of diffusion networks. In *ICML 2011*, 561–568.

Gomez-Rodriguez, M.; Leskovec, J.; and Krause, A. 2010. Inferring networks of diffusion and influence. In *KDD 2010*, 1019–1028.

Gomez-Rodriguez, M.; Leskovec, J.; and Schölkopf, B. 2013a. Modeling information propagation with survival theory. In *ICML 2013*, 666–674.

Gomez-Rodriguez, M.; Leskovec, J.; and Schölkopf, B. 2013b. Structure and dynamics of information pathways in online media. In *WSDM 2013*, 23–32.

Gripon, V., and Rabbat, M. 2013. Reconstructing a graph from path traces. In *ISIT 2013*, 2488–2492.

He, X.; Rekatsinas, T.; Foulds, J.; Getoor, L.; and Liu, Y. 2015. Hawkestopic: A joint model for network inference and topic modeling from text-based cascades. In *ICML 2015*, 871–880.

He, X.; Xu, K.; Kempe, D.; and Liu, Y. 2016. Learning influence functions from incomplete observations. In *NIPS 2016*, 2065–2073.

Huang, H.; Chiew, K.; Gao, Y.; Q, H.; and Li, Q. 2014. Rare category exploration. *Expert Systems with Applications* 41(9):4197–4210.

Kalimeris, D.; Singer, Y.; Subbian, K.; and Weinsberg, U. 2018. Learning diffusion using hyperparameters. In *ICML 2018*, 2420–2428.

Lancichinetti, A.; Fortunato, S.; and Radicchi, F. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E* 78(4).

Leskovec, J.; Adamic, L. A.; and Huberman, B. A. 2007. The dynamics of viral marketing. *ACM Transactions on the Web* 1(1):5.

Lokhov, A. 2016. Reconstructing parameters of spreading models from partial observations. In *NIPS 2016*, 3467–3475.

Myers, S., and Leskovec, J. 2010. On the convexity of latent social network inference. In *NIPS 2010*, 1741–1749.

Narasimhan, H.; Parkes, D. C.; and Singer, Y. 2015. Learnability of influence in networks. In *NIPS 2015*, 3186–3194.

Netrapalli, P., and Sanghavi, S. 2012. Learning the graph of epidemic cascades. In *SIGMETRICS 2012*, 211–222.

Newman, M. E. J. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* 74(3):036104.

Pouget-Abadie, J., and Horel, T. 2015. Inferring graphs from cascades: A sparse recovery framework. In *ICML 2015*, 977–986.

Rong, Y.; Zhu, Q.; and Cheng, H. 2016. A model-free approach to infer the diffusion network from event cascade. In *CIKM 2016*, 1653–1662.

Sefer, E., and Kingsford, C. 2015. Convex risk minimization to infer networks from probabilistic diffusion data at multiple scales. In *ICDE 2015*, 663–674.

Tang, Y.; Xiao, X.; and Shi, Y. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD 2014*, 75–86.

Wallinga, J., and Teunis, P. 2004. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology* 160(6):509–516.

Wang, S.; Hu, X.; Yu, P.; and Li, Z. 2014. MMRate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *KDD 2014*, 1246–1255.

Yan, Q.; Huang, H.; Gao, Y.; Lu, W.; and He, Q. 2017. Group-level influence maximization with budget constraint. In *DASFAA 2017*, 625–641.